ユーザーマニュアル

ALTOVA®
XMLSpy® 2017
POWERED BY RAPTORXML
Copyright ©2017 Altova GmbH. All rights reserved. Use of this software is governed by an Altova license agreement. XMLSpy, MapForce, StyleVision, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, MissionKit, FlowForce, RaptorXML, MobileTogether, and Altova as well as their respective logos are either registered trademarks or trademarks of Altova GmbH. Protected by U.S. Patents 7,739,292, 7,200,816, 9,501,456 and other pending patents. This software contains third party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html.

Altova XMLSpy 2017 Enterprise Edition
ユーザーマニュアル

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/ or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

発行日 :2017

(C) 2017 Altova GmbH

目次

Altova XMLSpy 2017 Enterprise Edition 3

2017 の新規機能 6

1 バージョン 2015 7

2 バージョン 2016 8

3 バージョン 2014 9

4 バージョン 2013 10

5 バージョン 2012 11

6 バージョン 2011 12

7 バージョン 2010 14

RaptorXML Server 16

XMLSpy チュートリアル 18

1 XMLSpy インターフェイス 19

1.1 表示 21
1.2 ウィンドウ 23
1.3 メニューとツールバー 25
1.4 テキストビューの設定 27

2 XML スキーマ :基本 30

2.1 新しい XML スキーマファイルの作成 31
2.2 名前空間の定義 33
2.3 コンテンツモデルの定義 34
2.4 ドラッグアンドドロップによる要素の追加 38
2.5 コンテンツモデルビューの構成 39
2.6 基本的なスキーマの完成 41

3 XML スキーマ :高度な機能 44

3.1 複合型と単純型を扱う 45
3.2 グローバル要素の参照 51
3.3 属性と列挙型 53

4 XML スキーマ :XMLSpy の機能 55

4.1 スキーマナビゲーション 56

5 スキーマドキュメンテーション 58

6 XML ドキュメント 62

6.1 新規 XML ファイルの作成 63
6.2 要素の型の指定 65
6.3 グリッドビューを使ったデータの入力 67
6.4 テキストビューを使ったデータの入力 68
6.5 ドキュメントの検証 71

6.6 要素と属性の追加 75
6.7 データベース /テーブルビューでの編集 77
6.8 スキーマの修正 81

7 XSLT 変換 83

7.1 XSLT ファイルの割り当て 84
7.2 XMLファイルの変換 85
7.3 XSLファイルの修正 86

8 プロジェクト管理 87

8.1 プロジェクトの利点 88
8.2 プロジェクトの作成 89

9 おしまい 91

ユーザーマニュアル 94

1 インターフェイスと環境 96

1.1 グラフィカルユーザインターフェース (GUI) 97
1.1.1 メイン ウィンドウ 98
1.1.2 プロジェクトウィンドウ 99
1.1.3 情報ウィンドウ 101
1.1.4 入力ヘルパー 101
1.1.5 出力ウィンドウ :メッセージ 102
1.1.6 出力ウィンドウ :XPath/ XQuery 103
・評価モード 105
・デバッグモード 109
・XPath とXQuery 仕様情報 113
1.1.7 出力ウィンドウ :XSL アウトライン 114
1.1.8 出力ウィンドウ :ファイルの中を検索 114
1.1.9 出力ウィンドウ :スキーマの中を検索 116
1.1.10 出力ウィンドウ :XBRL 内を検索 116
1.1.11 出力ウィンドウ :チャート 117
1.1.12 メニューバー、ツールバー、ステータスバー 118
1.2 アプリケーション環境 120
1.2.1 設定、カスタマイズ 120

1.2.2 チュートリアル、プロジェクト、サンプル 122
1.2.3 XMLSpy 製品機能と Altova 製品 122

2 編集ビュー 124

2.1 テキストビュー 125
2.1.1 テキストビューでのフォーマット 125
2.1.2 ドキュメントの表示 127
2.1.3 テキストビューでの編集 130
2.1.4 ドキュメント内の移動 132
2.1.5 検索および置換 135
2.1.6 XML ドキュメントの検証 141
2.1.7 テキストビューでの入力ヘルパー 141
2.1.8 テキストビューショートカット 142
2.2 グリッドビュー 145
2.2.1 グリッドビューでの編集 146
2.2.2 グリッドビューテーブル 147
2.2.3 グリッドビューでの入力ヘルパー 150
2.2.4 グリッドビューショートカット 152
2.3 スキーマ ビュー 154
2.3.1 XSD モード:XSD 1.0 と 1.1 156
2.3.2 スキーマ概要 159
・GUI メカニズム 160
・グローバル コンポーネント 164
2.3.3 コンテンツ モデル ビュー 170
・コンテンツ モデル オブジェクト 172
・コンテンツ モデル ビューでの編集 177
・条件型割り当て 182
・オープン コンテンツ モデル 185
2.3.4 属性、アサーション、同一性制約 187
・属性、属性グループ、属性ワイルドカード 189
・アサーション 192
・同一性制約 195
2.3.5 スキーマ ビューにおける入力ヘルパー 202
・コンポーネント 202
・詳細 205
・ファセット 206
2.3.6 検証とスマートフィックス 210
2.3.7 アサーションメッセージ 211
2.3.8 基底型の修正 213
2.3.9 スマート制約 214
2.3.10 xml:base、xml:id、xml:lang、xml:space 218
2.3.11 戻ると進む :位置の移動 219
2.4 WSDL ビュー 221

2.4.1 メインウィンドウ 222
2.4.2 概要入力ヘルパー 225
2.4.3 詳細入力ヘルパー 231
2.5 XBRL ビュー 232
2.5.1 メインウィンドウ :要素タブ 232
2.5.2 メインウィンドウ :定義、プレゼンテーション、計算 タブ 236
2.5.3 XBRL ビューの入力ヘルパー 238
2.6 Authentic ビュー 243
2.7 ブラウザービュー 244
2.8 アーカイブビュー 245

3 XML 247

3.1 XML ドキュメントを作成、開く、保存する 248
3.2 スキーマの割り当てと検証 250
3.3 テキストビューでの XML 編集 252
3.4 グリッドビューでの XML 編集 254
3.5 Authentic ビューでの XML 編集 256
3.6 XML ドキュメントの入力ヘルパー 258
3.7 XSLT と XQuery による処理 259
3.8 PDF フォント 261
3.9 チャート 263
3.9.1 チャートの作成 265
3.9.2 ソース XPath 269
3.9.3 X- 軸の選択 272
3.9.4 Y- 軸の選択 276
3.9.5 チャートのデータ 279
3.9.6 オーバーレイ 281
3.9.7 チャートの設定 :クイックリファレンス 281
3.9.8 チャートの設定と外観 283
・チャートの基本設定 284
・チャートの詳細設定 288
一般 291
各チャートの機能 293
色 296
X- 軸 298
Y- 軸 300
Z- 軸 301
3D アングル 301
サイズ 302
フォント 303
3.9.9 エクスポート 303
3.9.10 チャートの例 :基本 304
3.9.11 チャートの例 :詳細 305

3.9.12 チャートの例 :ローソク足チャート 311
3.10 XML 署名 314
3.10.1 XML 署名の作成 316
3.10.2 XML 署名の検証 317
3.10.3 証明書を使用する 319
3.11 その他の機能 323

4 DTD と XML スキーマ 325

4.1 DTD 326
4.2 XML スキーマ 327
4.3 スキーマサブセット 328
4.4 スキーマルール 330
4.4.1 ルールセットの管理 330
4.4.2 ルールセットの定義 331
4.5 XMLSpy のカタログ 336
4.6 SchemaAgent との連携 341
4.6.1 SchemaAgent Server へ接続 342
4.6.2 検索パス上のスキーマを開く 344
4.6.3 IIR の使用 344
4.6.4 SchemaAgent でスキーマを閲覧 348
4.6.5 SchemaAgent 検証 348
4.7 スキーマの中を検索 351
4.7.1 検索語 352
4.7.2 コンポーネント 353
4.7.3 プロパティ 355
4.7.4 スコープ 357
4.7.5 検索と置き換えコマンド 358
4.7.6 結果と情報 359
4.7.7 グローバルコンポーネントの検索と名前変更 360

5 XSLT 362

5.1 XSLT ドキュメント 363
5.2 XSLT 処理 365
5.3 XSL アウトライン 367
5.3.1 XSL アウトラインウィンドウ 368
5.3.2 情報ウィンドウ 370
5.4 XSL スピードオプティマイザー 373

6 XQuery 375

6.1 XQuery ドキュメントの編集 377
6.1.1 XQuery ドキュメント 377

6.1.2 XQuery 入力ヘルパー ... 378
6.1.3 XQuery の構文による色分け ... 379
6.1.4 XQuery インテリジェント編集 ... 380
6.2 XQuery 評価 ... 384
6.3 XQuery 検証 ... 385
6.4 XQuery/ Update 実行 ... 386
6.5 XQuery Update Facility ... 389
6.5.1 アップデートのプレビューと適用 ... 389
6.5.2 Update オペレーションと構文 ... 391
・ノードの削除 ... 392
・ノードの挿入 ... 392
・ノードの名前の変更 ... 393
・ノードの置換 ... 393
・ノードの値の置換 ... 394
・fn:put 関数 ... 394
6.6 XQuery と XML データベース ... 395

7 XSLT/ XQuery デバッガー とプロファイラー 398

7.1 XSLT と XQuery デバッガー ... 399
7.1.1 メカニズムとインターフェース ... 400
7.1.2 コマンドとツールバーアイコン ... 401
7.1.3 デバッガー設定 ... 403
7.1.4 デバッグセッションの開始 ... 405
7.1.5 情報ウィンドウ ... 406
・コンテキストウィンドウ ... 407
・変数ウィンドウ ... 407
・XPath- Watch ウィンドウ ... 408
・コールスタックウィンドウ ... 408
・メッセージウィンドウ ... 409
・テンプレートウィンドウ ... 409
・情報ウィンドウ ... 410
・トレースウィンドウ ... 410
・情報ウィンドウの配置 ... 410
7.1.6 ブレークポイント ... 412
7.1.7 トレースポイント ... 414
7.1.8 デバッガーショートカット ... 419
7.2 XSLT と XQuery プロファイラー ... 420
7.2.1 XSLT プロファイリング ... 424
7.2.2 XQuery プロファイリング ... 426
7.2.3 プロファイリングの結果：エクスポートとチャート ... 430

8 Authentic 432

8.1 Authentic View チュートリアル 434
8.1.1 Authentic View にて XML ドキュメントを開く 435
8.1.2 Authentic View インターフェース 436
8.1.3 ノードオペレーション 438
8.1.4 Authentic View におけるデータの入力 440
8.1.5 属性値の入力 442
8.1.6 エンティティの追加 442
8.1.7 ドキュメントの印刷 443
8.2 Authentic View インターフェイス 445
8.2.1 GUI の概要 445
8.2.2 Authentic View ツールバーアイコン 446
8.2.3 Authentic View メインウィンドウ 448
8.2.4 Authentic View 入力ヘルパー 450
8.2.5 Authentic View コンテキストメニュー 454
8.3 Authentic View における編集 456
8.3.1 基本的な編集 456
8.3.2 Authentic View におけるテーブル 460
・SPS テーブル 460
・CALS/ HTML テーブル 461
・CALS/ HTML テーブル編集アイコン 464
8.3.3 DB の編集 466
・DB テーブルのナビゲーション 466
・DB クエリ 467
・DB テーブルの修正 470
8.3.4 日付の入力 471
・日付の選択 471
・テキストの入力 472
8.3.5 エンティティの定義 472
8.3.6 XML 署名 474
8.3.7 Authentic View におけるイメージ 475
8.3.8 Authentic View におけるキーストローク 476
8.4 Authentic スクリプト 477

9 HTML と CSS 479

9.1 HTML 480
9.2 CSS 482

10 JSON と JSON スキーマ 486

10.1 JSON データ 488
10.2 JSON スキーマ 491
10.3 テキストビュー内の JSON ドキュメント 493
10.4 グリッドビュー内の JSON ドキュメント 498

10.5 JSON ドキュメントの検証 502
10.6 JSON インスタンスからの JSON スキーマの生成 504
10.7 JSON インスタンスを JSON スキーマから生成する 507
10.8 JSON と XML の間の変換 508
10.9 JSON スキーマビュー 509
10.9.1 グローバル定義の追加 510
10.9.2 入力ヘルパー 概要、詳細、制約 511
10.9.3 グローバルとローカル定義 514
10.9.4 デザインビュー 515
10.9.5 オブジェクトとプロパティ 516
10.9.6 指定されていないプロパティ 519
10.9.7 オブジェクトと依存性 521
10.9.8 配列 524
10.9.9 アトミック型 525
10.9.10 型セレクター (Any、Multiple) 527
10.9.11 演算子 528
10.9.12 デザインビューの構成 529
10.9.13 JSON スキーマドキュメンテーションの生成 530

11 Avro、Avro スキーマ 533

11.1 Avro スキーマ 535
11.2 JSON フォーマットの Avro データ 538
11.3 Avro ビュー :Avro バイナリのグリッドビュー 539

12 WSDL と SOAP 540

12.1 WSDL チュートリアル 541
12.1.1 新規ドキュメントの作成 541
12.1.2 ポートタイプの作成 542
12.1.3 バインディングの作成 543
12.1.4 サービスとポートの作成 544
12.1.5 WSDL ドキュメントの検証 545
12.1.6 ウェブサービスへの接続とファイルを開く 545
12.1.7 WSDL ファイルから SOAP リクエストを送信 547
12.1.8 WSDL ドキュメンテーションの作成 547
12.1.9 WSDL 2.0 への変換 549
12.2 SOAP 550
12.2.1 SOAP 検証 550
12.2.2 SOAP デバッガー 551
・SOAP コミュニケーション処理 553
・SOAP デバッガーオプション 553
・デバッグ セッションの開始 554
・SOAP リクエストエントリー ポイント 557

・ブレークポイントのセット 558
・デバッグ 559
・結果とエラーの分析 560
・ブレークポイントに関する更なる情報 561

13 XBRL **565**

13.1 基本のプロシージャ 566
13.1.1 タクソノミ:新規と既存 566
13.1.2 タクソノミファイルの概要 567
13.1.3 新規タクソノミの作成 569
13.1.4 タクソノミ内の名前空間 573
13.1.5 タクソノミのインポート 574
13.1.6 タクソノミファイルの設定 576
13.1.7 タクソノミへ要素の追加 578
13.1.8 リレーションシップとリンクロール 581
13.1.9 リレーションシップの作成 :パート1 582
13.1.10 リレーションシップの作成 :パート2 585
13.2 XBRL フォーミュラエディター 588
13.2.1 フォーミュラ リンクベース およびリンクロール 588
13.2.2 フォーミュラ コンポーネント 590
・アサーション と アサーションセット 591
・フォーミュラ 593
・パラメーター 595
・変数 595
・フィルター 596
・事前条件 603
・関数 603
・等値定義 603
13.2.3 コンポーネントプロパティとコンテンツの編集 604
13.2.4 フォーミュラコンポーネントリレーション 605
13.2.5 フォーミュラ パラメーター 606
13.2.6 フォーミュラ コンポーネントの検索 608
13.3 XBRL テーブル定義エディター 609
13.3.1 テーブル リンクベース およびリンクロール 610
13.3.2 テーブル構造 611
・X とY 軸 613
・定義ノード 614
ルールノード 615
リレーションシップノード 618
アスペクトノード 621
・Z 軸 621
13.3.3 テーブルコンポーネント 623
・テーブル 625

・ブレークダウン 625
・定義ノード:ルール 625
・定義ノード:コンセプトリレーションシップ 625
・定義ノード:ディメンションリレーションシップ 626
・定義ノード:アスペクト 626
13.3.4 コンポーネントプロパティとコンテンツの編集 627
13.3.5 テーブルコンポーネントリレーションシップ 627
13.3.6 テーブルパラメーター 628
13.3.7 テーブル レイアウトプレビュー 630
・テーブル レイアウトプレビュー内でのフォーミュラの作成 632
13.3.8 テーブルコンポーネントの検索 633
13.4 XBRL 内を検索 634
13.4.1 検索語 634
13.4.2 コマンドの実行 636
13.4.3 結果と情報 637

14 Office Open XML、ZIP、EPUB 639

14.1 OOXML ファイルの編集 641
14.2 OOXML サンプルファイル 643
14.3 ZIP ファイル 644
14.4 EPUB ファイル 645

15 データベース 649

15.1 データベースへの接続 650
15.1.1 データベース接続ウィザードの開始 651
15.1.2 データベースドライバー概要 652
15.1.3 ADO 接続の設定 654
・既存の Microsoft Access データベースへの接続 656
・新規の Microsoft Access データベースの作成 656
・SQL Server データ リンク プロパティの設定 657
・Microsoft Access データ リンク プロパティの設定 658
15.1.4 ADO.NET 接続のセットアップ 659
・Visual Studio 内で接続文字列を作成する 660
・サンプル ADO.NET 接続文字列 662
・ADO.NET サポートに関するメモ 663
15.1.5 ODBC 接続の設定 664
・使用可能な ODBC ドライバーの表示 666
15.1.6 JDBC 接続の設定 667
・CLASSPATH の構成 668
15.1.7 PostgreSQL 接続のセットアップ 669
15.1.8 SQLite 接続の設定 670
・既存の SQLite データベースへの接続 670

・新規の SQLite データベースの作成 671
15.1.9 グローバルリソースからの接続の使用 671
15.1.10 データベース接続のサンプル 672
・Firebird (ODBC) への接続 672
・Firebird (JDBC) への接続 674
・IBM DB2 (ODBC) への接続 675
・IBM DB2 for i (ODBC) への接続 679
・IBM Informix (JDBC) への接続 682
・Microsoft Access (ADO) への接続 683
・Microsoft SQL Server (ADO) への接続 684
・Microsoft SQL Server (ODBC) への接続 687
・MySQL (ODBC) への接続 688
・Oracle (ODBC) への接続 690
・Oracle (JDBC) への接続 694
PostgreSQL (ODBC) への接続 695
・Progress OpenEdge (ODBC) への接続 697
・Progress OpenEdge (JDBC) への接続 699
・Sybase (JDBC) への接続 700
15.2 サポートされているデータベース 702

16 Altova グローバルリソース 703

16.1 グローバル リソースの定義 704
16.1.1 ファイル 706
16.1.2 フォルダー 709
16.1.3 データベース 711
16.2 グローバルリソースを使用する 713
16.2.1 ファイルとフォルダーの割り当て 713
16.2.2 データベースの割り当て 716
16.2.3 構成の切り替え 716

17 プロジェクト 718

17.1 プロジェクトの作成と編集 719
17.2 プロジェクトを使用する 723

18 RaptorXML Server 725

18.1 サーバーとサーバー構成の追加 726
18.2 RaptorXML Server を使用して検証 730

19 検証オプション 731

19.1 DTD に対して XML 732
19.2 DTD 733
19.3 W3C スキーマに対して XML 734
19.4 W3C スキーマ 735
19.5 XBRL インスタンス 736
19.6 XBRL タクソノミ 737
19.7 XSLT 738
19.8 XQuery 739
19.9 JSON 740
19.10 JSON スキーマ 741
19.11 EDGAR 742

20 RaptorXML Server使用しての XSLT と XQuery 744

21 ファイル /ディレクトリの比較 746

21.1 ファイルの比較 747
21.2 ディレクトリの比較 748

22 Source Control 749

22.1 Setting Up Source Control 751
22.2 Supported Source Control Systems 752
22.3 Local Workspace Folder 754
22.4 Application Project 755
22.5 Add to Source Control 757
22.6 Working with Source Control 759
22.6.1 Add to, Remove from Source Control 759
22.6.2 Check Out, Check In 760
22.6.3 Getting Files as Read-Only 762
22.6.4 Copying and Sharing from Source Control 763
22.6.5 Changing Source Control 765
22.7 Source Control with Git 767
22.7.1 Enabling Git Source Control with GIT SCC Plug-in 767
22.7.2 Adding a Project to Git Source Control 768
22.7.3 Cloning a Project from Git Source Control 769

23 Visual Studio 内の XMLSpy 772

23.1 XMLSpy プラグインのインストール 773
23.2 XMLSpy スタンドアロンとの違い 775
23.3 Visual Studio での XMLSpy デバッガー 777

24 Eclipse 内の XMLSpy 778

24.1 Eclipse 用の XMLSpy プラグインをインストール 779
24.2 Eclipse 内の XMLSpy エントリポイント 784
24.3 XMLSpy のデバッガーパースペクティブ 787

25 Code Generator 788

25.1 Introduction to code generator 789
25.2 What's new 791
25.3 Generating Code from XML Schemas or DTDs 793
25.3.1 About Schema Wrapper Libraries (C++) 796
25.3.2 About Schema Wrapper Libraries (C#) 798
25.3.3 About Schema Wrapper Libraries (Java) 800
25.3.4 Integrating Schema Wrapper Libraries 802
25.3.5 Example: Using the Schema Wrapper Libraries 805
• Example Schema 805
• Reading and Writing XML Documents (C++) 808
• Reading and Writing XML Documents (C#) 812
• Reading and Writing XML Documents (Java) 820
25.4 Reference to Generated Classes (C++) 829
25.4.1 altova::DateTime 829
25.4.2 altova::Duration 832
25.4.3 altova::DayTimeDuration 834
25.4.4 altova::YearMonthDuration 834
25.4.5 altova::meta::Attribute 835
25.4.6 altova::meta::ComplexType 836
25.4.7 altova::meta::Element 837
25.4.8 altova::meta::SimpleType 837
25.4.9 [YourSchema]::[CDoc] 838
25.4.10 [YourSchema]::MemberAttribute 841
25.4.11 [YourSchema]::MemberElement 841
25.5 Reference to Generated Classes (C#) 843
25.5.1 Altova.Types.DateTime 843
25.5.2 Altova.Types.DateTimeFormat 846
25.5.3 Altova.Types.Duration 847
25.5.4 Altova.Xml.Meta.Attribute 849
25.5.5 Altova.Xml.Meta.ComplexType 850
25.5.6 Altova.Xml.Meta.Element 851
25.5.7 Altova.Xml.Meta.SimpleType 851
25.5.8 [YourSchema].[Doc] 852
25.5.9 [YourSchemaType].MemberAttribute 854
25.5.10 [YourSchemaType].MemberElement 854

25.6 Reference to Generated Classes (Java) 856
25.6.1 com.altova.types.DateTime 856
25.6.2 com.altova.types.Duration 860
25.6.3 com.altova.xml.meta.Attribute 863
25.6.4 com.altova.xml.meta.ComplexType 864
25.6.5 com.altova.xml.meta.Element 864
25.6.6 com.altova.xml.meta.SimpleType 865
25.6.7 com.[YourSchema].[Doc] 866
25.6.8 com.[YourSchema].[YourSchemaType].MemberAttribute 867
25.6.9 com.[YourSchema].[YourSchemaType].MemberElement 868
25.7 Code Generation Tips 870
25.8 Code Generator Options 871
25.9 SPL (Spy Programming Language) 872
25.9.1 Basic SPL structure 872
25.9.2 Declarations 873
25.9.3 Variables 874
25.9.4 Predefined variables 875
25.9.5 Creating output files 876
25.9.6 Operators 877
25.9.7 Conditions 878
25.9.8 Collections and foreach 878
25.9.9 Subroutines 879
• Subroutine declaration 879
• Subroutine invocation 880
• Subroutine example 881
25.9.10 Built in Types 882
• Library 882
• Namespace 882
• Type 883
• Member 884
• NativeBinding 885
• Facets 885
25.10 Error Codes 887

26 Menu Commands 888

26.1 File Menu 889
26.1.1 New 889
26.1.2 Open 893
26.1.3 Reload 897
26.1.4 Encoding 897
26.1.5 Close, Close All, Close All But Active 898
26.1.6 Save, Save As, Save All 898
26.1.7 Send by Mail 903

26.1.8 Print 904
26.1.9 Print Preview, Print Setup 905
26.1.10 Recent Files, Exit 906
26.2 Edit Menu 907
26.2.1 Undo, Redo 907
26.2.2 Cut, Copy, Paste, Delete 908
26.2.3 Copy as XML Text 909
26.2.4 Copy as Structured Text 909
26.2.5 Copy XPath 912
26.2.6 Copy XPointer 912
26.2.7 Insert 913
26.2.8 Pretty-Print 916
26.2.9 Strip Whitespaces 917
26.2.10 Select All 917
26.2.11 Find, Find Next 917
26.2.12 Replace 920
26.2.13 Find in Files 921
26.2.14 Bookmark Commands 923
26.2.15 Comment In/Out 924
26.3 Project Menu 925
26.3.1 New Project 927
26.3.2 Open Project 927
26.3.3 Reload Project 927
26.3.4 Close Project 928
26.3.5 Save Project, Save Project As 928
26.3.6 Source Control 928
• Open from Source Control 929
• Enable Source Control 930
• Get Latest Version 930
• Get, Get Folders 930
• Check Out, Check In 931
• Undo Check Out 933
• Add to Source Control 934
• Remove from Source Control 934
• Share from Source Control 934
• Show History 936
• Show Differences 937
• Show Properties 938
• Refresh Status 939
• Source Control Manager 939
• Change Source Control 939
26.3.7 Add Files to Project 940
26.3.8 Add Global Resource to Project 940
26.3.9 Add URL to Project 940
26.3.10 Add Active File to Project 941

26.3.11 Add Active And Related Files to Project 941
26.3.12 Add Project Folder to Project 941
26.3.13 Add External Folder to Project 941
26.3.14 Add External Web Folder to Project 943
26.3.15 Script Settings 946
26.3.16 Properties 947
26.3.17 Most Recently Used Projects 950
26.4 XML Menu 951
26.4.1 Insert 951
• Insert Attribute 952
• Insert Element 952
• Insert Text 952
• Insert CDATA 952
• Insert Comment 953
• Insert XML 953
• Insert Processing Instruction 953
• Insert XInclude 953
• Insert DOCTYPE 955
• Insert ExternalID 956
• Insert ELEMENT 956
• Insert ATTLIST 957
• Insert ENTITY 957
• Insert NOTATION 957
• Insert Encoded External File 957
26.4.2 Append 958
• Append Attribute 959
• Append Element 959
• Append Text 959
• Append CDATA 959
• Append Comment 960
• Append XML 960
• Append Processing Instruction 960
• Append XInclude 960
• Append DOCTYPE 962
• Append ExternalID 963
• Append ELEMENT 963
• Append ATTLIST 964
• Append ENTITY 964
• Append NOTATION 964
• Append Encoded External File 964
26.4.3 Add Child 965
• Add Child Attribute 965
• Add Child Element 965
• Add Child Text 966
• Add Child CDATA 966

- Add Child Comment 966
- Add Child XML 966
- Add Child Processing Instruction 966
- Add Child XInclude 966
- Add Child DOCTYPE 968
- Add Child ExternalID 969
- Add Child ELEMENT 969
- Add Child ATTLIST 969
- Add Child ENTITY 969
- Add Child NOTATION 969
- Add Child Encoded External File 970

26.4.4 Convert To 970
- Convert To Attribute 971
- Convert To Element 971
- Convert To Text 971
- Convert To CDATA 971
- Convert To Comment 971
- Convert To XML 971
- Convert To Processing Instruction 971
- Convert To DOCTYPE 971
- Convert To ExternalID 971
- Convert To ELEMENT 972
- Convert To ATTLIST 972
- Convert To ENTITY 972
- Convert To NOTATION 972

26.4.5 Table 972
- Display as Table 972
- Insert Row 973
- Append Row 973
- Ascending Sort 973
- Descending Sort 974

26.4.6 Move Left 974
26.4.7 Move Right 974
26.4.8 Enclose in Element 975
26.4.9 Evaluate XPath 975
26.4.10 Check Well-Formedness 975
26.4.11 Validate XML 977
26.4.12 Validate XML on Server (high-performance) 981
26.4.13 Validating WSDL Files 982
26.4.14 Update Entry Helpers 983
26.4.15 Namespace Prefix 983
26.4.16 Create XML Signature 984
26.4.17 Verify XML Signature 986

26.5 DTD/Schema Menu 989
26.5.1 Assign DTD 989

26.5.2 Assign Schema 990
26.5.3 Include Another DTD 990
26.5.4 Go to DTD 990
26.5.5 Go to Schema 990
26.5.6 Go to Definition 990
26.5.7 Generate DTD/ Schema 991
26.5.8 Flatten DTD 992
26.5.9 Convert DTD to Schema 992
26.5.10 Flatten Schema 994
26.5.11 Convert Schema to DTD 994
26.5.12 Convert to UML 995
26.5.13 Generate XML from DB, Excel, EDI with MapForce 996
26.5.14 Design HTML/ PDF/ Word Output with StyleVision 996
26.5.15 Generate Sample XML/ JSON File 996
26.5.16 Generate Program Code 999
26.5.17 Flush Memory Cache 1000
26.6 Schema Design Menu 1001
26.6.1 Schema Settings 1001
26.6.2 Save Diagram 1003
26.6.3 Generate Documentation 1004
• Documentation Options 1005
• User-Defined Design 1007
26.6.4 Configure View 1009
26.6.5 Zoom 1012
26.6.6 Display All Globals 1012
26.6.7 Display Diagram 1013
26.6.8 Schema Extensions for Databases 1013
• Enable Oracle Schema Extensions 1013
• Oracle Schema Settings 1014
• Enable Microsoft SQL Server Schema Extensions 1014
• Named Schema Relationships 1015
• Unnamed Element Relationships 1015
26.6.9 Connect to SchemaAgent Server 1016
26.6.10 Disconnect from SchemaAgent Server 1017
26.6.11 Show in SchemaAgent 1017
26.6.12 SchemaAgent Validation 1017
26.6.13 Create Schema Subset 1017
26.6.14 Flatten Schema 1019
26.7 XSL/ XQuery Menu 1020
26.7.1 XSL Transformation 1021
26.7.2 XSL Speed Optimizer 1021
26.7.3 XSL-FO Transformation 1022
26.7.4 XSL Parameters / XQuery Variables 1023
26.7.5 XQuery/ Update Execution 1026
26.7.6 Enable Back-Mapping 1027

26.7.7 Enable XSLT/ XQuery Profiling ... 1029
26.7.8 Assign XSL ... 1029
26.7.9 Assign XSL- FO ... 1029
26.7.10 Assign Sample XML File ... 1030
26.7.11 Go to XSL ... 1030
26.7.12 Start Debugger / Go ... 1030
26.7.13 Stop Debugger ... 1030
26.7.14 Restart Debugger ... 1031
26.7.15 End Debugger Session ... 1031
26.7.16 Step Into ... 1031
26.7.17 Step Out ... 1031
26.7.18 Step Over ... 1031
26.7.19 Show Current Execution Node ... 1031
26.7.20 Insert/ Remove Breakpoint ... 1032
26.7.21 Insert/ Remove Tracepoint ... 1032
26.7.22 Enable/ Disable Breakpoint ... 1032
26.7.23 Enable/ Disable Tracepoint ... 1032
26.7.24 Breakpoints/ Tracepoints ... 1032
26.7.25 Debug Windows ... 1033
26.7.26 Debug Settings ... 1033
26.8 Authentic Menu ... 1034
26.8.1 New Document ... 1034
26.8.2 Edit Database Data ... 1035
26.8.3 Assign a StyleVision Stylesheet ... 1036
26.8.4 Edit StyleVision Stylesheet ... 1036
26.8.5 Select New Row with XML Data for Editing ... 1036
26.8.6 XML Signature ... 1037
26.8.7 Define XML Entities ... 1038
26.8.8 View Markup ... 1040
26.8.9 RichEdit ... 1040
26.8.10 Append/ Insert/ Duplicate/ Delete Row ... 1041
26.8.11 Move Row Up/ Down ... 1041
26.8.12 Generate HTML, RTF, PDF, Word 2007+ Document ... 1042
26.8.13 Trusted Locations ... 1042
26.9 DB Menu ... 1044
26.9.1 Query Database ... 1044
• Data Sources ... 1045
• Browser Pane: Viewing the DB Objects ... 1047
• Query Pane: Description and Features ... 1050
• Query Pane: Working with Queries ... 1053
• Results and Messages ... 1054
26.9.2 IBM DB2 ... 1056
• Manage XML Schemas ... 1056
• Assign XML Schema ... 1059
26.9.3 SQL Server ... 1061

• Manage XML Schemas 1061
26.9.4 Oracle XML DB 1062
• Manage XML Schemas 1063
• Browse Oracle XML documents 1064
26.10 Convert Menu 1066
26.10.1 Import Text File 1066
26.10.2 Import Database Data 1068
26.10.3 Import Microsoft Word Document 1072
26.10.4 Create XML Schema from DB Structure 1072
26.10.5 DB Import Based on XML Schema 1076
26.10.6 Create DB Structure from XML Schema 1077
26.10.7 Export to Text Files 1079
26.10.8 Export to a Database 1082
26.10.9 Convert XML Instance to/ from JSON 1084
26.10.10 Convert XML Schema to/ from JSON Schema 1087
26.11 View Menu 1090
26.11.1 Text View 1090
26.11.2 Enhanced Grid View 1090
26.11.3 Schema Design View 1091
26.11.4 WSDL Design View 1091
26.11.5 XBRL Taxonomy View 1091
26.11.6 Authentic View 1091
26.11.7 Browser View 1092
26.11.8 Expand 1092
26.11.9 Collapse 1092
26.11.10 Expand Fully 1092
26.11.11 Collapse Unselected 1092
26.11.12 Optimal Widths 1093
26.11.13 Word Wrap 1093
26.11.14 Go to Line/ Character 1093
26.11.15 Go to File 1093
26.11.16 Text View Settings 1094
26.12 Browser Menu 1096
26.12.1 Back 1096
26.12.2 Forward 1096
26.12.3 Stop 1096
26.12.4 Refresh 1096
26.12.5 Fonts 1097
26.12.6 Separate Window 1097
26.13 WSDL Menu 1098
26.13.1 WSDL 1.1 Components 1098
• Messages 1098
• Operations 1099
• PortType 1099
• Binding 1099

• Service 1100
26.13.2 WSDL 2.0 Components 1100
• Interface 1100
• Binding 1102
• Service 1102
26.13.3 Types, Save Diagram 1103
26.13.4 Generate Documentation 1104
• Documentation Options 1104
• User-Defined Design 1106
26.13.5 Reparse WSDL Document 1107
26.13.6 Convert to WSDL 2.0 1107
26.13.7 Generate WSDL Program Code with MapForce 1108
26.14 SOAP Menu 1109
26.14.1 Create New SOAP Request 1109
26.14.2 Send Request to Server 1111
26.14.3 SOAP Request Settings 1112
26.14.4 Soap Debugger Session 1115
26.14.5 Go 1116
26.14.6 Single Step 1116
26.14.7 Break on Next Request 1117
26.14.8 Break on Next Response 1117
26.14.9 Stop the Proxy Server 1117
26.14.10 SOAP Debugger Options 1117
26.15 XBRL Menu 1118
26.15.1 Arcroles 1118
26.15.2 Linkroles 1120
26.15.3 Namespace Prefixes 1121
26.15.4 Set Target Namespace 1122
26.15.5 Parameter Values 1122
26.15.6 Import/ Reference 1123
26.15.7 Find Component by ID 1124
26.15.8 Generate Documentation 1125
• Documentation Options 1125
• User-Defined Design 1127
26.15.9 View Settings 1128
26.15.10 Generate XBRL from DB, Excel, CSV with MapForce 1129
26.15.11 Present XBRL as HTML/ PDF/ Word with StyleVision 1130
26.15.12 Execute Formula (on Server) 1130
26.15.13 Generate Table (on Server) 1133
26.15.14 Transform Inline XBRL 1134
26.16 Tools Menu 1135
26.16.1 Spelling 1135
26.16.2 Spelling Options 1139
26.16.3 Scripting Editor 1142
26.16.4 Macros 1143

26.16.5 Comparisons 1143
• Compare Open File With 1143
• Compare Directories 1145
• Compare Options 1147
26.16.6 User-defined Tools 1150
26.16.7 Global Resources 1150
26.16.8 Active Configuration 1151
26.16.9 Manage Raptor Servers 1151
26.16.10 Raptor Servers and Configurations 1155
26.16.11 Customize 1155
• Commands 1155
• Toolbars 1157
• Tools 1158
• Keyboard 1160
• Menu 1163
• Macros 1164
• Plug-Ins 1166
• Options 1167
• Customize Context Menu 1167
26.16.12 Restore Toolbars and Windows 1169
26.16.13 Options 1170
• File 1170
• File Types 1173
• Editing 1175
• View 1176
• Grid Fonts 1178
• Schema Fonts 1179
• WSDL Fonts 1180
• XBRL Fonts 1181
• Text Fonts 1182
• Colors 1183
• Encoding 1185
• XSL 1186
• XSL Speed Optimizer 1188
• XQuery 1188
• Scripting 1189
• Taxonomy Packages 1190
• Source Control 1192
26.17 Window Menu 1194
26.17.1 Cascade 1194
26.17.2 Tile Horizontally 1194
26.17.3 Tile Vertically 1194
26.17.4 Project Window 1194
26.17.5 Info Window 1194
26.17.6 Entry Helpers 1195

26.17.7 Output Windows 1195
26.17.8 Project and Entry Helpers 1195
26.17.9 All On/ Off 1195
26.17.10 Currently Open Window List 1195
26.18 Help Menu 1196
26.18.1 Table of Contents, Index, Search 1196
26.18.2 Keyboard Map 1197
26.18.3 Activation, Order Form, Registration, Updates 1197
26.18.4 Other Commands 1200
26.19 Command Line 1202

プログラマーのレファレンス 1204

1 Scripting Editor 1206

1.1 Overview 1208
1.1.1 Scripting Projects in XMLSpy 1208
1.1.2 The Scripting Editor GUI 1210
1.1.3 Components of a Scripting Project 1213
1.2 Creating a Scripting Project 1215
1.3 Global Declarations 1217
1.4 Forms 1219
1.4.1 Creating a New Form 1219
1.4.2 Form Design and Form Objects 1220
1.4.3 Form Events 1222
1.5 Events 1225
1.6 Macros 1228
1.6.1 Creating and Editing a Macro 1228
1.6.2 Running a Macro 1230
1.6.3 Debugging a Macro 1232
1.7 Programming Points 1233
1.7.1 Built-in Commands 1234
• Form usage and commands 1240
1.8 Migrating to Scripting Editor 2010 and Later 1242

2 IDE Plugins 1245

2.1 Registration of IDE PlugIns 1246
2.2 ActiveX Controls 1247
2.3 Configuration XML 1248
2.4 ATL sample files 1251

2.4.1 Interface description (IDL) 1251
2.4.2 Class definition 1253
2.4.3 Implementation 1253
2.5 IXMLSpyPlugIn 1256
2.5.1 OnCommand 1256
2.5.2 OnUpdateCommand 1257
2.5.3 OnEvent 1257
2.5.4 GetUIModifications 1260
2.5.5 GetDescription 1260

3 Application API 1261

3.1 Overview 1263
3.1.1 Object Model 1263
3.1.2 Programming Languages 1264
• JScript 1265
Start Application 1266
Simple Document Access 1266
Iteration 1267
Error Handling 1267
Events 1268
Import and Export of Data 1269
Import from Database 1269
Export to Database 1270
Import from Text 1271
Export to Text 1272
• VBScript 1273
Events 1273
Example: Using Events 1274
• C# 1275
Add Reference to XMLSpy API 1279
Application Startup and Shutdown 1279
Opening Documents 1280
Iterating through Open Documents 1281
Errors and COM Output Parameters 1281
Events 1282
• Java 1283
Example Java Project 1284
Application Startup and Shutdown 1287
Simple Document Access 1288
Iterations 1288
Use of Out-Parameters 1289
Event Handlers 1289
3.1.3 The DOM and XMLData 1290

3.1.4 Obsolete: Authentic View Row operations 1292
3.1.5 Obsolete: Authentic View Editing operations 1293
3.2 Interfaces 1294
3.2.1 Application 1294
• Events 1296
OnBeforeOpenDocument 1296
OnBeforeOpenProject 1296
OnDocumentOpened 1297
OnProjectOpened 1297
• ActiveDocument 1298
• AddMacroMenuItem 1298
• AddXSLT_XQParameter 1298
• Application 1298
• ClearMacroMenu 1299
• CreateXMLSchemaFromDBStructure 1299
• CurrentProject 1299
• Dialogs 1300
• Documents 1300
• Edition 1300
• FindInFiles 1300
• GetDatabaseImportElementList 1301
• GetDatabaseSettings 1301
• GetDatabaseTables 1301
• GetExportSettings 1302
• GetTextImportElementList 1302
• GetTextImportExportSettings 1303
• GetXSLT_XQParameterCount 1303
• GetXSLT_XQParameterName 1304
• GetXSLT_XQParameterXPath 1304
• ImportFromDatabase 1304
• ImportFromSchema 1305
• ImportFromText 1306
• ImportFromWord 1306
• IsAPISupported 1307
• MajorVersion 1307
• MinorVersion 1307
• NewProject 1307
• OpenProject 1308
• Parent 1308
• Quit 1308
• ReloadSettings 1309
• RemoveXSLT_XQParameter 1309
• RunMacro 1309
• ScriptingEnvironment 1309
• ServicePackVersion 1310

- ShowApplication 1310
- ShowFindInFiles 1310
- ShowForm 1311
- Status 1311
- URLDelete 1311
- URLMakeDirectory 1311
- Visible 1312
- WarningNumber 1312
- WarningText 1312

3.2.2 AuthenticContextMenu 1312
- CountItems 1313
- DeleteItem 1313
- GetItemText 1313
- InsertItem 1313
- SetItemText 1313

3.2.3 AuthenticDataTransfer 1314
- dropEffect 1314
- getData 1315
- ownDrag 1315
- type 1315

3.2.4 AuthenticEventContext 1315
- EvaluateXPath 1316
- GetEventContextType 1316
- GetNormalizedTextValue 1316
- GetVariableValue 1317
- GetXMLNode 1317
- IsAvailable 1317
- SetVariableValue 1317

3.2.5 AuthenticRange 1318
- AppendRow 1319
- Application 1320
- CanPerformAction 1320
- CanPerformActionWith 1321
- Clone 1321
- CollapsToBegin 1321
- CollapsToEnd 1322
- Copy 1322
- Cut 1322
- Delete 1322
- DeleteRow 1323
- DuplicateRow 1323
- EvaluateXPath 1324
- ExpandTo 1324
- FirstTextPosition 1324
- FirstXMLData 1325

- FirstXMLDataOffset 1326
- GetElementAttributeNames 1327
- GetElementAttributeValue 1327
- GetElementHierarchy 1328
- GetEntityNames 1329
- GetVariableValue 1329
- Goto 1329
- GotoNext 1330
- GotoNextCursorPosition 1330
- GotoPrevious 1331
- GotoPreviousCursorPosition 1331
- HasElementAttribute 1332
- InsertEntity 1332
- InsertRow 1332
- IsCopyEnabled 1333
- IsCutEnabled 1333
- IsDeleteEnabled 1333
- IsEmpty 1334
- IsEqual 1334
- IsFirstRow 1334
- IsInDynamicTable 1334
- IsLastRow 1335
- IsPasteEnabled 1335
- IsSelected 1335
- IsTextStateApplied 1335
- LastTextPosition 1336
- LastXMLData 1336
- LastXMLDataOffset 1337
- MoveBegin 1338
- MoveEnd 1338
- MoveRowDown 1339
- MoveRowUp 1339
- Parent 1339
- Paste 1340
- PerformAction 1340
- Select 1341
- SelectNext 1341
- SelectPrevious 1342
- SetElementAttributeValue 1342
- SetFromRange 1343
- SetVariableValue 1344
- Text 1344

3.2.6 AuthenticView 1344

- Events 1345

OnBeforeCopy 1345

OnBeforeCut ... 1346
OnBeforeDelete ... 1346
OnBeforeDrop ... 1347
OnBeforePaste ... 1348
OnBeforeSave ... 1348
OnDragOver ... 1348
OnKeyboardEvent ... 1349
OnLoad ... 1350
OnMouseEvent ... 1350
OnSelectionChanged ... 1351
OnToolbarButtonClicked ... 1351
OnToolbarButtonExecuted ... 1353
OnUserAddedXMLNode ... 1353
- Application ... 1353
- AsXMLString ... 1354
- ContextMenu ... 1354
- CreateXMLNode ... 1354
- DisableAttributeEntryHelper ... 1354
- DisableElementEntryHelper ... 1355
- DisableEntityEntryHelper ... 1355
- DocumentBegin ... 1355
- DocumentEnd ... 1355
- DoNotPerformStandardAction ... 1355
- EvaluateXPath ... 1356
- Event ... 1356
- EventContext ... 1356
- GetToolbarButtonState ... 1356
- Goto ... 1357
- IsRedoEnabled ... 1358
- IsUndoEnabled ... 1358
- MarkupVisibility ... 1358
- Parent ... 1358
- Print ... 1359
- Redo ... 1359
- Selection ... 1359
- SetToolbarButtonState ... 1360
- Undo ... 1360
- UpdateXMLInstanceEntities ... 1360
- WholeDocument ... 1361
- XMLDataRoot ... 1361

3.2.7 CodeGeneratorDlg ... 1361
- Application ... 1362
- CompatibilityMode (obsolete) ... 1362
- CPPSettings_DOMType ... 1363
- CPPSettings_GenerateVC6ProjectFile ... 1363

- CPPSettings_GenerateGCCMakefile ... 1363
- CPPSettings_GenerateVSProjectFile ... 1363
- CPPSettings_LibraryType ... 1364
- CPPSettings_UseMFC ... 1364
- CSharpSettings_ProjectType ... 1364
- OutputPath ... 1364
- OutputPathDialogAction ... 1365
- OutputResultDialogAction ... 1365
- Parent ... 1365
- ProgrammingLanguage ... 1365
- PropertySheetDialogAction ... 1366
- TemplateFileName ... 1366

3.2.8 DatabaseConnection ... 1366
- ADOConnection ... 1367
- AsAttributes ... 1367
- CommentIncluded ... 1368
- CreateMissingTables ... 1368
- CreateNew ... 1368
- DatabaseKind ... 1368
- DatabaseSchema ... 1369
- ExcludeKeys ... 1369
- File ... 1369
- ForeignKeys ... 1369
- ImportColumnsType ... 1370
- IncludeEmptyElements ... 1370
- NullReplacement ... 1370
- NumberDateTimeFormat ... 1370
- ODBCConnection ... 1371
- PrimaryKeys ... 1371
- SchemaExtensionType ... 1371
- SchemaFormat ... 1371
- SQLSelect ... 1372
- TextFieldLen ... 1372
- UniqueKeys ... 1372

3.2.9 Dialogs ... 1372
- Application ... 1373
- CodeGeneratorDlg ... 1373
- FileSelectionDlg ... 1373
- Parent ... 1374
- SchemaDocumentationDlg ... 1374
- GenerateSampleXMLDlg ... 1374
- DTDSchemaGeneratorDlg ... 1374
- FindInFilesDlg ... 1375
- WSDLDocumentationDlg ... 1375
- WSDL20DocumentationDlg ... 1375

- XBRLDocumentationDlg ... 1375

3.2.10 Document ... 1376

- Events ... 1377
  - *OnBeforeSaveDocument* ... 1377
  - *OnBeforeCloseDocument* ... 1378
  - *OnBeforeValidate* ... 1378
  - *OnCloseDocument* ... 1379
  - *OnViewActivation* ... 1379
- Application ... 1380
- AssignDTD ... 1380
- AssignSchema ... 1380
- AssignXSL ... 1381
- AssignXSLFO ... 1381
- AsXMLString ... 1381
- AuthenticView ... 1381
- Close ... 1382
- ConvertDTDOrSchema ... 1382
- ConvertDTDOrSchemaEx ... 1383
- ConvertToWSDL20 ... 1383
- CreateChild ... 1384
- CreateDBStructureFromXMLSchema ... 1384
- CreateSchemaDiagram ... 1385
- CurrentViewMode ... 1385
- DataRoot ... 1385
- DocEditView ... 1386
- Encoding ... 1386
- EndChanges ... 1386
- ExecuteXQuery ... 1387
- ExportToDatabase ... 1387
- ExportToText ... 1388
- FlattenDTDOrSchema ... 1389
- FullName ... 1389
- GenerateDTDOrSchema ... 1390
- GenerateDTDOrSchemaEx ... 1390
- GenerateProgramCode ... 1390
- GenerateSampleXML ... 1391
- GenerateSchemaDocumentation ... 1391
- GenerateWSDL20Documentation ... 1391
- GenerateWSDLDocumentation ... 1392
- GenerateXBRLDocumentation ... 1392
- GetDBStructureList ... 1392
- GetExportElementList ... 1393
- GetPathName (obsolete) ... 1393
- GridView ... 1394
- IsModified ... 1394

- IsValid 1394
- IsValidEx 1394
- IsWellFormed 1395
- Name 1396
- Parent 1396
- Path 1396
- RootElement 1396
- Save 1397
- SaveAs 1397
- Saved 1397
- SaveInString 1397
- SaveToURL 1398
- SetActiveDocument 1398
- SetEncoding (obsolete) 1398
- SetExternalIsValid 1400
- SetPathName (obsolete) 1400
- StartChanges 1400
- Suggestions 1401
- SwitchViewMode 1401
- TextView 1401
- Title 1401
- TransformXSL 1402
- TransformXSLEx 1402
- TransformXSLFO 1402
- TreatXBRLInconsistenciesAsErrors 1402
- UpdateViews 1403
- UpdateXMLData 1403

3.2.11 Documents 1403

- Count 1404
- Item 1404
- NewAuthenticFile 1404
- NewFile 1405
- NewFileFromText 1405
- OpenAuthenticFile 1405
- OpenFile 1406
- OpenURL 1406
- OpenURLDialog 1407

3.2.12 DTDSchemaGeneratorDlg 1407

- Application 1408
- AttributeTypeDefinition 1408
- DTDSchemaFormat 1408
- FrequentElements 1408
- GlobalAttributes 1409
- MaxEnumLength 1409
- MergeAllEqualNamed 1409

- OnlyStringEnums 1409
- OutputPath 1409
- OutputPathDialogAction 1410
- Parent 1410
- ResolveEntities 1410
- TypeDetection 1410
- ValueList 1410

3.2.13 ElementList 1411
- Count 1411
- Item 1411
- RemoveElement 1411

3.2.14 ElementListItem 1412
- ElementKind 1412
- FieldCount 1412
- Name 1412
- RecordCount 1412

3.2.15 ExportSettings 1413
- CreateKeys 1413
- ElementList 1413
- EntitiesToText 1413
- ExportAllElements 1414
- ExportCompleteXML 1414
- FromAttributes 1414
- FromSingleSubElements 1414
- FromTextValues 1414
- IndependentPrimaryKey 1414
- Namespace 1415
- StartFromElement 1415
- SubLevelLimit 1415

3.2.16 FileSelectionDlg 1415
- Application 1416
- DialogAction 1416
- FullName 1416
- Parent 1416

3.2.17 FindInFilesDlg 1417
- AdvancedXMLSearch 1417
- Application 1417
- DoReplace 1418
- FileExtension 1418
- Find 1418
- IncludeSubfolders 1418
- MatchCase 1418
- MatchWholeWord 1419
- Parent 1419
- RegularExpression 1419

- Replace 1419
- ReplaceOnDisk 1419
- SearchInProjectFilesDoExternal 1420
- SearchLocation 1420
- ShowResult 1420
- StartFolder 1420
- XMLAttributeContents 1420
- XMLAttributeNames 1421
- XMLCData 1421
- XMLComments 1421
- XMLElementContents 1421
- XMLElementNames 1421
- XMLPI 1422
- XMLRest 1422

3.2.18 FindInFilesResult 1422
- Application 1422
- Count 1423
- Document 1423
- Item 1423
- Parent 1423
- Path 1423

3.2.19 FindInFilesResultMatch 1423
- Application 1424
- Length 1424
- Line 1424
- LineText 1424
- Parent 1425
- Position 1425
- Replaced 1425

3.2.20 FindInFilesResults 1425
- Application 1425
- Count 1426
- Item 1426
- Parent 1426

3.2.21 GenerateSampleXMLDlg 1426
- Application 1427
- ChoiceMode 1427
- ConsiderSampleValueHints 1427
- ContentOfNillableElementsIsNonMandatory 1427
- FillAttributesWithSampleData 1428
- FillElementsWithSampleData 1428
- FillWithSampleData - obsolete 1428
- LocalNameOfRootElement 1428
- NamespaceURIOfRootElement 1428
- NonMandatoryAttributes 1429

- NonMandatoryElements ... 1429
- Optimization - obsolete ... 1429
- OptionsDialogAction ... 1429
- Parent ... 1429
- RepeatCount ... 1430
- SampleValueHints ... 1430
- SchemaOrDTDAssignment ... 1430
- TakeFirstChoice - obsolete ... 1430
- TryToUseNonAbstractTypes ... 1430

3.2.22 GridView ... 1431
- Events ... 1431
  OnBeforeDrag ... 1431
  OnBeforeDrop ... 1431
  OnBeforeStartEditing ... 1432
  OnEditingFinished ... 1432
  OnFocusChanged ... 1433
- CurrentFocus ... 1433
- Deselect ... 1433
- IsVisible ... 1433
- Select ... 1434
- SetFocus ... 1434

3.2.23 SchemaDocumentationDlg ... 1434
- AllDetails ... 1435
- Application ... 1435
- CreateDiagramsFolder ... 1435
- DiagramFormat ... 1436
- EmbedCSSInHTML ... 1436
- EmbedDiagrams ... 1436
- GenerateRelativeLinks ... 1437
- IncludeAll ... 1437
- IncludeAttributeGroups ... 1437
- IncludeComplexTypes ... 1437
- IncludeGlobalAttributes ... 1438
- IncludeGlobalElements ... 1438
- IncludeGroups ... 1438
- IncludeIndex ... 1438
- IncludeLocalAttributes ... 1439
- IncludeLocalElements ... 1439
- IncludeRedefines ... 1439
- IncludeReferencedSchemas ... 1440
- IncludeSimpleTypes ... 1440
- MultipleOutputFiles ... 1440
- OptionsDialogAction ... 1440
- OutputFile ... 1441
- OutputFileDialogAction ... 1441

- OutputFormat 1441
- Parent 1442
- ShowAnnotations 1442
- ShowAttributes 1442
- ShowChildren 1442
- ShowDiagram 1443
- ShowEnumerations 1443
- ShowIdentityConstraints 1443
- ShowNamespace 1443
- ShowPatterns 1444
- ShowProgressBar 1444
- ShowProperties 1444
- ShowResult 1445
- ShowSingleFacets 1445
- ShowSourceCode 1445
- ShowType 1445
- ShowUsedBy 1446
- SPSFile 1446
- UseFixedDesign 1446

3.2.24 SpyProject 1446
- CloseProject 1447
- ProjectFile 1447
- RootItems 1447
- SaveProject 1447
- SaveProjectAs 1448

3.2.25 SpyProjectItem 1448
- ChildItems 1448
- FileExtensions 1448
- ItemType 1449
- Name 1449
- Open 1449
- ParentItem 1449
- Path 1449
- ValidateWith 1449
- XMLForXSLTransformation 1450
- XSLForXMLTransformation 1450
- XSLTransformationFileExtension 1450
- XSLTransformationFolder 1450

3.2.26 SpyProjectItems 1450
- AddFile 1450
- AddFolder 1451
- AddURL 1451
- Count 1451
- Item 1452
- RemoveItem 1452

3.2.27 TextImportExportSettings 1452
- CommentIncluded 1452
- DestinationFolder 1452
- EnclosingCharacter 1453
- Encoding 1453
- EncodingByteOrder 1453
- FieldDelimiter 1453
- FileExtension 1453
- HeaderRow 1453
- ImportFile 1454
- RemoveDelimiter 1454
- RemoveNewline 1454

3.2.28 TextView 1454
- Events 1455

OnBeforeShowSuggestions 1455
OnChar 1455
- Application 1456
- GetRangeText 1456
- GoToLineChar 1456
- Length 1456
- LineCount 1457
- LineFromPosition 1457
- LineLength 1457
- MoveCaret 1457
- Parent 1457
- PositionFromLine 1458
- ReplaceText 1458
- SelectionEnd 1458
- SelectionStart 1458
- SelectText 1458
- SelText 1459
- Text 1459

3.2.29 WSDLDocumentationDlg 1459
- AllDetails 1460
- Application 1460
- CreateDiagramsFolder 1460
- DiagramFormat 1461
- EmbedCSSInHTML 1461
- EmbedDiagrams 1461
- GlobalElementsAndTypesOnly 1462
- IncludeAll 1462
- IncludeBinding 1462
- IncludeImportedWSDLFiles 1462
- IncludeMessages 1463
- IncludeOverview 1463

- IncludePortType 1463
- IncludeService 1463
- IncludeTypes 1464
- MultipleOutputFiles 1464
- OptionsDialogAction 1464
- OutputFile 1465
- OutputFileDialogAction 1465
- OutputFormat 1465
- Parent 1465
- SeparateSchemaDocument 1466
- ShowBindingDiagram 1466
- ShowExtensibility 1466
- ShowMessageParts 1466
- ShowPort 1467
- ShowPortTypeDiagram 1467
- ShowPortTypeOperations 1467
- ShowProgressBar 1468
- ShowResult 1468
- ShowServiceDiagram 1468
- ShowSourceCode 1468
- ShowTypesDiagram 1469
- ShowUsedBy 1469
- UseFixedDesign 1469
- SPSFile 1469

3.2.30 WSDL20DocumentationDlg 1470

- AllDetails 1471
- Application 1471
- CreateDiagramsFolder 1471
- DiagramFormat 1471
- EmbedCSSInHTML 1472
- EmbedDiagrams 1472
- GlobalElementsAndTypesOnly 1472
- IncludeAll 1473
- IncludeBinding 1473
- IncludeImportedWSDLFiles 1473
- IncludeInterface 1473
- IncludeOverview 1474
- IncludeService 1474
- IncludeTypes 1474
- MultipleOutputFiles 1474
- OptionsDialogAction 1475
- OutputFile 1475
- OutputFileDialogAction 1475
- OutputFormat 1476
- Parent 1476

- SeparateSchemaDocument ... 1476
- ShowBindingDiagram ... 1476
- ShowEndpoint ... 1477
- ShowExtensibility ... 1477
- ShowFault ... 1477
- ShowInterfaceDiagram ... 1477
- ShowOperation ... 1478
- ShowProgressBar ... 1478
- ShowResult ... 1478
- ShowServiceDiagram ... 1479
- ShowSourceCode ... 1479
- ShowTypesDiagram ... 1479
- ShowUsedBy ... 1479
- SPSFile ... 1480
- UseFixedDesign ... 1480

3.2.31 XBRLDocumentationDlg ... 1480

- AllDetails ... 1481
- Application ... 1481
- CreateDiagramsFolder ... 1482
- DiagramFormat ... 1482
- EmbedCSSInHTML ... 1482
- EmbedDiagrams ... 1483
- IncludeAll ... 1483
- IncludeCalculationLinkroles ... 1483
- IncludeDefinitionLinkroles ... 1483
- IncludeGlobalElements ... 1484
- IncludeNamespacePrefixes ... 1484
- IncludeOverview ... 1484
- IncludePresentationLinkroles ... 1484
- OptionsDialogAction ... 1485
- OutputFile ... 1485
- OutputFileDialogAction ... 1485
- OutputFormat ... 1486
- Parent ... 1486
- ShortQualifiedName ... 1486
- ShowAbstract ... 1486
- ShowBalance ... 1487
- ShowDiagram ... 1487
- ShowImportedElements ... 1487
- ShowItemtype ... 1487
- ShowLabels ... 1488
- ShowLinkbaseReferences ... 1488
- ShowNillable ... 1488
- ShowPeriod ... 1489
- ShowProgressBar ... 1489

- ShowReferences 1489
- ShowResult 1489
- ShowSubstitutiongroup 1490
- SPSFile 1490
- UseFixedDesign 1490

3.2.32 XMLData 1491

- AppendChild 1492
- CountChildren 1492
- CountChildrenKind 1493
- EraseAllChildren 1493
- EraseChild 1493
- EraseCurrentChild 1493
- GetChild 1494
- GetChildAttribute 1494
- GetChildElement 1495
- GetChildKind 1495
- GetCurrentChild 1495
- GetFirstChild 1496
- GetNamespacePrefixForURI 1496
- GetNextChild 1496
- GetTextValueXMLDecoded 1497
- HasChildren 1498
- HasChildrenKind 1498
- InsertChild 1498
- InsertChildAfter 1498
- InsertChildBefore 1499
- IsSameNode 1499
- Kind 1499
- MayHaveChildren 1500
- Name 1500
- Parent 1500
- SetTextValueXMLEncoded 1500
- TextValue 1501

3.3 Interfaces (obsolete) 1502

3.3.1 AuthenticEvent (obsolete) 1502

- altKey (obsolete) 1503
- altLeft (obsolete) 1504
- button (obsolete) 1505
- cancelBubble (obsolete) 1506
- clientX (obsolete) 1507
- clientY (obsolete) 1508
- ctrlKey (obsolete) 1509
- ctrlLeft (obsolete) 1510
- dataTransfer (obsolete) 1511
- fromElement (obsolete) 1512

- keyCode (obsolete) ... 1512
- propertyName (obsolete) ... 1513
- repeat (obsolete) ... 1513
- returnValue (obsolete) ... 1513
- shiftKey (obsolete) ... 1514
- shiftLeft (obsolete) ... 1515
- srcElement (obsolete) ... 1516
- type (obsolete) ... 1517

3.3.2 AuthenticSelection (obsolete) ... 1518

- End (obsolete) ... 1519
- EndTextPosition (obsolete) ... 1519
- Start (obsolete) ... 1520
- StartTextPosition (obsolete) ... 1520

3.3.3 OldAuthentictView (obsolete) ... 1521

- ApplyTextState (obsolete) ... 1523
- CurrentSelection (obsolete) ... 1524
- EditClear (obsolete) ... 1524
- EditCopy (obsolete) ... 1525
- EditCut (obsolete) ... 1525
- EditPaste (obsolete) ... 1526
- EditRedo (obsolete) ... 1526
- EditSelectAll (obsolete) ... 1527
- EditUndo (obsolete) ... 1527
- event (obsolete) ... 1528
- GetAllowedElements (obsolete) ... 1528
- GetNextVisible (obsolete) ... 1531
- GetPreviousVisible (obsolete) ... 1532
- IsEditClearEnabled (obsolete) ... 1533
- IsEditCopyEnabled (obsolete) ... 1533
- IsEditCutEnabled (obsolete) ... 1534
- IsEditPasteEnabled (obsolete) ... 1534
- IsEditRedoEnabled (obsolete) ... 1535
- IsEditUndoEnabled (obsolete) ... 1535
- IsRowAppendEnabled (obsolete) ... 1536
- IsRowDeleteEnabled (obsolete) ... 1536
- IsRowDuplicateEnabled (obsolete) ... 1537
- IsRowInsertEnabled (obsolete) ... 1537
- IsRowMoveDownEnabled (obsolete) ... 1538
- IsRowMoveUpEnabled (obsolete) ... 1538
- IsTextStateApplied (obsolete) ... 1539
- IsTextStateEnabled (obsolete) ... 1539
- LoadXML (obsolete) ... 1540
- MarkUpView (obsolete) ... 1540
- RowAppend (obsolete) ... 1541
- RowDelete (obsolete) ... 1541

- RowDuplicate (obsolete) 1542
- RowInsert (obsolete) 1542
- RowMoveDown (obsolete) 1543
- RowMoveUp (obsolete) 1543
- SaveXML (obsolete) 1544
- SelectionMoveTabOrder (obsolete) 1545
- SelectionSet (obsolete) 1546
- XMLRoot (obsolete) 1547

3.4 Enumerations 1548
3.4.1 ENUMApplicationStatus 1548
3.4.2 SPYAttributeTypeDefinition 1548
3.4.3 SPYAuthenticActions 1548
3.4.4 SPYAuthenticDocumentPosition 1548
3.4.5 SPYAuthenticElementActions 1549
3.4.6 SPYAuthenticElementKind 1549
3.4.7 SPYAuthenticMarkupVisibility 1549
3.4.8 SPYAuthenticToolbarButtonState 1550
3.4.9 SPYDatabaseKind 1550
3.4.10 SPYDialogAction 1550
3.4.11 SPYDOMType 1550
3.4.12 SPYDTDSchemaFormat 1551
3.4.13 SPYEncodingByteOrder 1551
3.4.14 SPYExportNamespace 1551
3.4.15 SPYFindInFilesSearchLocation 1551
3.4.16 SPYFrequentElements 1551
3.4.17 SPYImageKind 1551
3.4.18 SPYImportColumnsType 1552
3.4.19 SPYKeyEvent 1552
3.4.20 SPYKeyStatus 1552
3.4.21 SPYLibType 1552
3.4.22 SPYLoading 1552
3.4.23 SPYMouseEvent 1553
3.4.24 SPYNumberDateTimeFormat 1553
3.4.25 SPYProgrammingLanguage 1554
3.4.26 SPYProjectItemTypes 1554
3.4.27 SPYProjectType 1554
3.4.28 SpySampleXMLGenerationChoiceMode 1554
3.4.29 SPYSampleXMLGenerationOptimization (Obsolete) 1555
3.4.30 SpySampleXMLGenerationSampleValueHints 1555
3.4.31 SPYSampleXMLGenerationSchemaOrDTDAssignment 1555
3.4.32 SPYSchemaDefKind 1556
3.4.33 SPYSchemaDocumentationFormat 1556
3.4.34 SPYSchemaExtensionType 1556
3.4.35 SPYSchemaFormat 1557
3.4.36 SPYTextDelimiters 1557

3.4.37 SPYTextEnclosing 1557
3.4.38 SPYTypeDetection 1557
3.4.39 SPYURLTypes 1557
3.4.40 SPYValidateXSDVersion 1558
3.4.41 SPYValidateErrorFormat 1558
3.4.42 SPYViewModes 1558
3.4.43 SPYVirtualKeyMask 1559
3.4.44 SPYXMLDataKind 1559
3.5 Application API for Java (obsolete) 1561
3.5.1 Sample source code (obsolete) 1562
3.5.2 SpyApplication (obsolete) 1564
3.5.3 SpyCodeGeneratorDlg (obsolete) 1565
3.5.4 SpyDatabaseConnection (obsolete) 1565
3.5.5 SpyDialogs (obsolete) 1566
3.5.6 SpyDoc (obsolete) 1567
3.5.7 SpyDocuments (obsolete) 1568
3.5.8 SpyDTDSchemaGeneratorDlg (obsolete) 1569
3.5.9 SpyElementList (obsolete) 1569
3.5.10 SpyElementListItem (obsolete) 1570
3.5.11 SpyExportSettings (obsolete) 1570
3.5.12 SpyFileSelectionDlg (obsolete) 1571
3.5.13 SpyFindInFilesDlg (obsolete) 1571
3.5.14 SpyFindInFilesMatch (obsolete) 1572
3.5.15 SpyFindInFilesResult (obsolete) 1572
3.5.16 SpyFindInFilesResults (obsolete) 1573
3.5.17 SpyGenerateSampleXMLDlg (obsolete) 1573
3.5.18 SpyGridView (obsolete) 1574
3.5.19 SpyProject (obsolete) 1574
3.5.20 SpyProjectItem (obsolete) 1575
3.5.21 SpyProjectItems (obsolete) 1575
3.5.22 SpySchemaDocumentationDlg (obsolete) 1575
3.5.23 SpyTextImportExportSettings (obsolete) 1577
3.5.24 SpyTextView (obsolete) 1578
3.5.25 SpyWSDL20DocumentationDlg (obsolete) 1578
3.5.26 SpyWSDLDocumentationDlg (obsolete) 1580
3.5.27 SpyXBRLDocumentationDlg (obsolete) 1582
3.5.28 SpyXMLData (obsolete) 1584
3.5.29 Authentic (obsolete) 1585
- SpyAuthenticRange (obsolete) 1585
- SpyAuthenticView (obsolete) 1586
- SpyDocEditSelection (obsolete) 1587
- SpyDocEditView (obsolete) 1587
3.5.30 Predefined constants (obsolete) 1588
- SPYApplicationStatus (obsolete) 1588
- SPYAttributeTypeDefinition (obsolete) 1589

- SPYAuthenticActions (obsolete) 1589
- SPYAuthenticDocumentPosition (obsolete) 1589
- SPYAuthenticElementKind (obsolete) 1590
- SPYAuthenticMarkupVisibility (obsolete) 1590
- SPYDatabaseKind (obsolete) 1591
- SPYDialogAction (obsolete) 1591
- SPYDOMType (obsolete) 1591
- SPYDTDSchemaFormat (obsolete) 1592
- SPYEncodingByteOrder (obsolete) 1592
- SPYExportNamespace (obsolete) 1592
- SPYFindInFilesSearchLocation (obsolete) 1593
- SPYFrequentElements (obsolete) 1593
- SPYImageKind (obsolete) 1593
- SPYImportColumnsType (obsolete) 1594
- SPYLibType (obsolete) 1594
- SPYLoading (obsolete) 1594
- SPYNumberDateTimeFormat (obsolete) 1594
- SPYProgrammingLanguage (obsolete) 1595
- SPYProjectItemTypes (obsolete) 1595
- SPYProjectType (obsolete) 1595
- SPYSampleXMLGenerationOptimization (obsolete) 1596
- SPYSampleXMLGenerationSchemaOrDTDAssignment (obsolete) 1596
- SPYSchemaDefKind (obsolete) 1597
- SPYSchemaDocumentationFormat (obsolete) 1597
- SPYSchemaExtensionType (obsolete) 1598
- SPYSchemaFormat (obsolete) 1598
- SPYTextDelimiters (obsolete) 1598
- SPYTextEnclosing (obsolete) 1599
- SPYTypeDetection (obsolete) 1599
- SPYURLTypes (obsolete) 1599
- SpyViewModes (obsolete) 1600
- SPYWhitespaceComparison (obsolete) 1600
- SPYXMLDataKind (obsolete) 1600

4 ActiveX Integration 1602

4.1 Prerequisites 1603
4.2 Adding the ActiveX Controls to the Toolbox 1605
4.3 Integration at Application Level 1607
4.4 Integration at Document Level 1609
4.5 ActiveX Integration Examples 1613
4.5.1 C# 1613
- Running the Sample C# Solution 1613

4.5.2 HTML 1615

- HTML Integration at Application Level ... 1615
  - *Instantiate the Control* ... 1615
  - *Add Button to Open Default Document* ... 1616
  - *Add Buttons for Code Generation* ... 1616
  - *Connect to Custom Events* ... 1617
- HTML Integration at Document Level ... 1617
  - *Instantiate the XMLSpyControl* ... 1618
  - *Create Editor Window* ... 1618
  - *Create Project Window* ... 1618
  - *Create Placeholder for Helper Windows* ... 1619
  - *Create a Custom Toolbar* ... 1619
  - *Create More Buttons* ... 1620
  - *Create Event Handler to Update Button Status* ... 1621

4.5.3 Java ... 1622
- Example Java Project ... 1623
- Creating the ActiveX Controls ... 1625
- Loading Data in the Controls ... 1626
- Basic Event Handling ... 1626
- Menus ... 1627
- UI Update Event Handling ... 1628
- Creating an XML Tree ... 1628

4.6 Command Reference ... 1630
- 4.6.1 "File" Menu ... 1630
- 4.6.2 "Edit" Menu ... 1631
- 4.6.3 "Project" Menu ... 1631
- 4.6.4 "XML" Menu ... 1632
- 4.6.5 "DTD/Schema" Menu ... 1635
- 4.6.6 "Schema design" Menu ... 1635
- 4.6.7 "XSL/XQuery" Menu ... 1636
- 4.6.8 "Authentic" Menu ... 1638
- 4.6.9 "DB" Menu ... 1639
- 4.6.10 "Convert" Menu ... 1639
- 4.6.11 "View" Menu ... 1640
- 4.6.12 "Browser" Menu ... 1640
- 4.6.13 "WSDL" Menu ... 1641
- 4.6.14 "SOAP" Menu ... 1644
- 4.6.15 "XBRL" Menu ... 1644
- 4.6.16 "Tools" Menu ... 1645
- 4.6.17 "Window" Menu ... 1645
- 4.6.18 "Help" Menu ... 1646

4.7 Object Reference ... 1647

4.7.1 XMLSpyCommand ... 1647
- Accelerator ... 1647
- ID ... 1648
- IsSeparator ... 1648

- Label 1648
- Name 1648
- StatusText 1648
- SubCommands 1648
- ToolTip 1649

4.7.2 XMLSpyCommands 1649
- Count 1649
- Item 1649

4.7.3 XMLSpyControl 1649
- Properties 1650

Appearance 1650
Application 1650
BorderStyle 1650
CommandsList 1651
EnableUserPrompts 1651
IntegrationLevel 1651
MainMenu 1652
Toolbars 1652

- Methods 1653

Exec 1653
Open 1653
QueryStatus 1654

- Events 1654

OnCloseEditingWindow 1654
OnContextChanged 1654
OnDocumentOpened 1655
OnFileChangedAlert 1655
OnLicenseProblem 1655
OnOpenedOrFocused 1655
OnToolWindowUpdated 1656
OnUpdateCmdUI 1656
OnValidationWindowUpdated 1656

4.7.4 XMLSpyControlDocument 1656
- Properties 1657

Appearance 1657
BorderStyle 1657
Document 1657
IsModified 1658
Path 1658
ReadOnly 1658

- Methods 1658

Exec 1659
New 1659
Open 1659
QueryStatus 1659

Reload 1660
Save 1660
SaveAs 1660
・Events 1660
OnActivate 1660
OnContextChanged 1661
OnDocumentClosed 1661
OnDocumentOpened 1661
OnDocumentSaveAs 1661
OnFileChangedAlert 1661
OnModifiedFlagChanged 1662
OnSetEditorTitle 1662
4.7.5 XMLSpyControlPlaceHolder 1662
・Properties 1662
Label 1663
PlaceholderWindowID 1663
Project 1663
・Methods 1663
OpenProject 1663
CloseProject 1664
・Events 1664
OnModifiedFlagChanged 1664
OnSetLabel 1664
4.7.6 Enumerations 1664
・ICActiveXIntegrationLevel 1664
・XMLSpyControlPlaceholderWindow 1665

付録 1668

1 XSLT および XQuery エンジンに関する情報 1669

1.1 XSLT 1.0 1670
1.2 XSLT 2.0 1671
1.3 XSLT 3.0 1673
1.4 XQuery 1.0 1674
1.5 XQuery 3.1 1677

2 XSLT とXPath/ XQuery 関数 1678

2.1 Altova 拡張関数 1680
2.1.1 XSLT 関数 1681

2.1.2 XPath/ XQuery 関数 :日付と時刻 1683
2.1.3 XPath/ XQuery 関数 :位置情報 1695
2.1.4 XPath/ XQuery 関数 :イメージに関連 1703
2.1.5 XPath/ XQuery 関数 :数値 1711
2.1.6 XPath/ XQuery 関数 :シーケンス 1714
2.1.7 XPath/ XQuery 関数 :文字列 1719
2.1.8 XPath/ XQuery 関数 :その他 1725
2.1.9 チャート関数 1725
・チャートデータの XML 構造 1729
・チャート関数 1734
2.2 その他の拡張関数 1738
2.2.1 Java 拡張関数 1738
・ユーザー定義のクラスファイル 1739
・ユーザー定義の JAR ファイル 1742
・Java :コンストラクター 1743
・Java :静的メソッドと静的フィールド 1743
・Java :インスタンスメソッドとインスタンスフィールド 1744
・データ型 :XPath/ XQuery から Java へ 1745
・データ型 :Java から XPath/ XQuery へ 1746
2.2.2 .NET 拡張関数 1746
・.NET コンストラクター 1748
・.NET :静的メソッドと静的フィールド 1749
・.NET :インスタンスメソッドとインスタンスフィールド 1749
・データ型 :XPath/ XQuery から .NET へ 1750
・データ型 :.NET から XPath/ XQuery へ 1751
2.2.3 XSLT に対する MSXSL スクリプト 1751

3 DB 内で生成された XML スキーマのデータ型 1754

3.1 ADO 1755
3.2 MS Access 1756
3.3 MS SQL Server 1757
3.4 MySQL 1758
3.5 ODBC 1759
3.6 Oracle 1760
3.7 Sybase 1761

4 XML スキーマから生成された DB のデータ型 1762

4.1 MS Access 1763
4.2 MS SQL Server 1765
4.3 MySQL 1767
4.4 Oracle 1769

5 技術データ 1771

5.1 OS とメモリ要件 1772
5.2 Altova XML バリデーター 1773
5.3 Altova XSLT と XQuery エンジン 1774
5.4 Unicode のサポート 1775
5.5 インターネットの使用 1776

6 ライセンス情報 1777

6.1 電子的なソフトウェアの配布 1778
6.2 ソフトウェアのアクティベーションとライセンスの計測 1779
6.3 知的所有権 1780
6.4 エンドユーザー使用許諾契約書 1781

インデックス 1793

Altova XMLSpy 2017 Enterprise Edition

Altova XMLSpy 2017 Enterprise Edition

Altova XMLSpy 2017 Enterprise Edition

Altova XMLSpy 2017 Enterprise Edition はXML、XML スキーマ、XML/XSLT、XQuery、SOAP、WSDL とWeb サービス技術を含むエンタープライズクラスのアプリケーションのデザイン、編集、デバッグを行うための業界標準のXML 開発環境です。J2EE、NET、そしてデータベース開発者に対して究極の生産性向上をもたらすでしょう。XMLSpy® は Windows XP/Vista, Windows 7/8/10 とWindows Server 2003/2008/2012/2016 で動作します。XMLSpy は64ビット版と32ビット版がご利用になれます。

このドキュメントは以下のセクションにより構成されています：

- XMLSpy チュートリアル
- ユーザーマニュアルとレファレンス
- プログラマーのレファレンス
- 付録

以下の書式で利用可能です：

- XMLSpy に内蔵のヘルプマニュアルとして（ヘルプメニューか**F1** から）
- HTMLやPDF ファイルとして、または書籍の購入を通して（Altova ウェブサイトからお求めいただけます）

最終更新日：*2017年 -04月 -03日*

Altova XMLSpy 2017 Enterprise Edition

2017 の新規機能

2017 の新規機能

バージョン 2017 リリース 3

XMLSpy **バージョン 2017** にて新たに追加された機能は以下のとおりです。

- バックマッピング 機能が有効化されていると、XSLT 変換とXQuery 実行が行われ、結果ドキュメントが元の XSLT+XML または XQuery+XML ドキュメントにバックマッピングされます。結果ドキュメント内のノードをクリックすると、**XSLT 命令** とその特定の結果フラグメントを生成した**XML ソースデータ** また、XSLT 命令、または、XML データノードをクリックすると、他の2つのドキュメント内の対応するノードがハイライトされます。詳細に関しては、**XSL/XQuery | XSLT/XQuery バックマッピングの有効化** を参照してください。
- JSON5 へのサポート: この機能には自動補完、JSON スキーマに対する検証、XML からの変換、XML への変換が含まれます。
- テキストビュー : JSON、XML、などの異なるドキュメントの種類の検証エラー は使用できる場合、番号付けられたマージン、スクロールバーにより示されており、エラーを素早くフィックスします。
- テキストビュー : ナビゲーション と検索と置換 関数が強化されました。

バージョン 2017

XMLSpy **バージョン 2017** にて新たに追加された機能は以下のとおりです。

- テキストビュー : テキストのハイライト は、現在選択されているテキストの一致を、一致に関するメタデータと共に表示します。
- テキストビュー : スクロールバー は、ドキュメントのサイズ、ドキュメントのサイズに相対した現在の位置に関する情報を含んでいます。
- テキストビュー : 検索と置き換え関数への追加関数と簡単な使用方法
- Avro データシリアル化: Avro データ構造とAvro スキーマのためのインテリジェントな編集のサポートです。JSON とAvro スキーマ内のAvro データ構造の検証を行います。
- Avro ビュー: 検索することのできるグリッドフォーマットのAvro バイナリのデータブロックを表示するAvro バイナリ専用のグリッドビューです。
- インラインのXBRL ドキュメントをXBRL インスタンスドキュメントに変換することが可能になりました。
- XMLSpy と共にダウンロードすることができるXBRL タクソノミパッケージによりXMLSpy は、パッケージのオフラインリソースを検索し使用できるようになりました。
- RaptorXML Server の追加機能

1 バージョン 2015

バージョン 2015 リリース 4

XMLSpy **バージョン 2015-4** の新機能は以下にリストされています。

- XMLSpy 内部からAltova RaptorXML Server を使用してXML およびXBRL ドキュメントを検証する機能。Raptor の検証オプションを使用して検証をカスタム化することができます。

バージョン 2015 リリース 3

XMLSpy **バージョン 2015-3** の新機能は以下にリストされています。

- XMLSpy のXPath/XQuery ビルダーとエバリュエーター内およびXSLT およびXQuery エンジン内およびXPath/XQuery ビルダーおよびエバリュエーター内の2014 年 12月 18日版の XPath とXQuery 関数 および演算子 31候補推奨へのサポート。
- ポップアップする改善されたウィンドウレイアウトとインタラクティブな関数と演算子情報などを含むXPath/XQuery ビルダーとエバリュエーターへの強化。
- 位置情報のための新規 Altova XPath/XQuery 拡張関数 。
- SOAP 通信のための、セキュリティ設定 は、HTTP レイヤーのセキュリティに加え、ウェブサービスレイヤーでサポートされています。HTTP レイヤーでのサポートは、プロパティ認証も含みます。SOAP デバッガーの接続設定 も、強化されたこのHTTPS セキュリティをサポートします。
- `enum:enumerationItemType` 型のコンセプトに関しては、マルチ言語のラベルを持つ、2014 年版 10月 29 日版拡張可能な列挙の推奨で定義されている拡張可能な列挙がサポートされます。

バージョン 2015

XMLSpy **バージョン 2015** の新機能は以下にリストされています。

- XQuery Update Facility へのサポート：XMLSpy はXQuery Update ドキュメント型を認識しXQuery Update ドキュメント内の式によりターゲットXML ドキュメントが更新されるように、ドキュメントを処理します。 この機能により、ターゲットのドキュメント全体を再生成することなく、ドキュメントの特定の箇所を更新することができます。
- XQuery Update プレビュー: XQuery Update 式は、XPath/XQuery 出力ウィンドウに入力されることができ、更新をプレビューすることができます。これらの"保留中の更新 "は、適用または拒否されることができます。
- XBRL テーブルレイアウトプレビュー は、(XBRL ビュー内の)テーブル定義エディターにより選択されたテーブルを表示します。テーブルのレイアウトはテーブルの構造で定義されています。
- (テーブルパラメータにより定義された)XBRL テーブルパラメーターとXBRL テーブルセットへのサポート。
- XBRL テーブルレイアウトプレビューからXBRL フォーミュラを構築する。

2 バージョン 2016

バージョン 2016 リリース 2

XMLSpy **バージョン 2016 リリース 2** の新機能は以下にリストされています。

- XPath/XQuery デバッグ機能 がXPath/XQuery ウィンドウに追加されました。
- JSON スキーマドキュメンテーション をHTML、RTF、およびMS Word フォーマットで自動的に生成することができます。
- JSON スキーマを元に構成されたJSON データを含む、JSON インスタンスファイルがJSON スキーマから生成されることができます。
- XML インスタンスへからJSON インスタンス への変換が向上され、ユーザーは変換オプションを選択できるようになりました。
- XML スキーマへからJSON スキーマ への変換が向上され、ユーザーは変換オプションを選択できるようになりました。
- JSON インスタンスがアクティブな場合、それぞれ対応するコマンドを**DTD/スキーマ**メニュー内でクリックするとJSON スキーマ とJSON 定義 へ移動することができます。
- テキストビュー内でXML ドキュメントの要素の名前を編集する場合、開始と終了タグの名前が一致するかにより名前が異なる色で表示されます。この機能は視覚的な編集のヘルプの役割を果たします。
- テキストビュー 内のXML 、XQuery 、JSON とCSS ドキュメント内でのソースの折りたたみと、XPath/XQuery ウィンドウ内の折りたたまれたノードのための省略記号の表示が追加されました。省略記号にマウスをポイントすると折りたたまれたノードの内容がポップアップで表示されます。
- Eclipse へのサポートがEclipse 4.5 に拡張されました。
- XBRL タクソノミエディター内でのXBRL アサーションの重要度へのサポート

バージョン 2016

XMLSpy **バージョン 2016** の新機能は以下にリストされています。

- 新しいJSON スキーマビュー が導入されましたこの機能によりJSON キーマをグラフィカルに編集することができます。
- JSON インスタンスからのJSON スキーマの生成。
- JSON ドキュメント検証: JSON インスタンスとJSON スキーマの検証を行います。
- テキストビュー およびグリッドビュー 内でのJSON ドキュメントの検証が導入されました。(JSON スキーマに関連により取得された情報をベースにして新しい自動補完機能によりインスタンスおよびスキーマドキュメントを素早く正確に作成することができます。
- XMLSpy 内からAltova RaptorXML Server を使用したXSLT 変換。この機能によりRaptor の広範囲のオプションを使用してXSLT 高速変換をカスタマイズすることができます。
- XMLSpy 内からAltova RaptorXML Server を使用したXQuery の実行。この機能によりRaptor の広範囲のオプションを使用してXQuery の実行をカスタマイズすることができます。

3 バージョン 2014

バージョン 2014 リリース 2

XMLSpy **バージョン 2014 r2** の新機能は以下にリストされています。

- 新しい Altova XSLT と XPath/XQuery 拡張機能 の追加。
- XML スキーマ 1.1 アサーションのためのメッセージの追加機能。これらのメッセージにより、アサーションのテストの失敗が無効性の理由であり、XML ドキュメントがなぜ無効であるかを説明することができます。
- XSL スピードオプティマイザー は XSLT スタイルシートを分析し、XSLT スタイルシートと共に保存される最適化のための戦略を導き出します。最適化されたスタイルシートを使用した、後続の変換はより速く実行されます。
- XBRL テーブル定義エディター がビューに搭載されました。
- XBRL インスタンスドキュメントからの XBRL フォーミュラの評価の生成 (XML と JSON フォーマット)。
- XBRL タクソノミとインスタンスドキュメントからの XBRL テーブルの生成 (XML と HTML フォーマット)。

バージョン 2014

XMLSpy **バージョン 2014** の新機能は以下にリストされています。

- XSLT 3.0 と XQuery 3.0 エンジンが (XPath と XQuery 関数 3.0 をサポートする) XMLSpy 2014 に搭載されました。XSLT、XQuery、XPath/XQuery 関数に関連する機能で XSLT 3.0、XQuery 3.0、XPath および XQuery 3.0 関数のための 3.0 インテリジェントな編集、検証と変換 実行を使用することができます。
- XPath/XQuery ウィンドウ内での XPath 式の評価 が拡張され XPath 3.0. もカバーするようになりました。
- XML スキーマ (XSD) 1.1 は次をサポートします スキーマビュー内での XSD 1.1 ドキュメントのインテリジェントな編集と検証。これは、XSD 1.0 ドキュメントのインテリジェントな編集と検証の前の機能への追加です。
- ユーザー定義のエラー制限 により XML スキーマ検証を指定することができます。これは、エラーをバッチで修正し、時間を短縮するために役に立ちます。
- DTD から XML スキーマへの変換 また XML スキーマから DTD への変換が強化されました。また、DTD と XML スキーマはフラット化することができます。フラット化は、含まれた/インポートされたモジュールが親ファイル内のディレクトリに直接保存されるプロセスです。
- XML スキーマからのサンプル XML ドキュメントの生成 が強化され、`choice` グループの特定のブランチの要素を選択することができます。
- XSL オンライン ウィンドウ内では、変換に使用されるモードがコンボボックスでグラフィカルに選択することができます。
- Eclipse 4.3 との統合は、バージョン 3.8 および 4.2 へのサポートに追加して、Eclipse プラットフォームの最新のバージョンをサポートします。Eclipse 3.6 と 3.7 へのサポートは中止されました。
- XBRL フォーミュラエディター: XBRL フォーミュラは XBRL ビューのフォーミュラタブで定義および管理することができます。フォーミュラタブは概要入力ヘルパーと詳細入力ヘルパーを使用して、フォーミュラを作成し編集し、フォーミュラコンポーネント間の関係を定義します。
- XBRL ビューの詳細入力ヘルパーは、選択されたアイテムの型の詳細を表示するタイプタブを含みます。

4 バージョン 2013

バージョン2013

XMLSpy **バージョン 2013**の新機能は以下にリストされています。

- コマンドをインターフェースへ追加することで (ツール| カスタマイズ| ツール)、XMLSpy 内部からアクティブなドキュメントやアクティブなプロジェクトを外部アプリケーションで開くことができます。
- XML ファイルが妥当でない場合、XML バリデーターに搭載されたスマートフィックスによるエラーの修正をスキーマ内の情報をベースに行うことができます。スマートフィックスへはメッセージウィンドウからアクセスすることができます。
- 空白の削除 コマンドにより、XML ドキュメントから全ての空白を削除することができます。
- WSDL ドキュメント内部にXML スキーマを埋め込みならびにWSDL ドキュメントからXML スキーマを抽出することで、WSDL ドキュメント内に配置されたXML インラインスキーマと、WSDL ドキュメントへのスキーマインポートを簡単に切り替えることができます。
- WSDL ドキュメンテーション機能の改善により、インポートまたはインクルードされたXML スキーマのコンポーネントをレポートに含める除外することができるようになりました。更に、スキーマのレポートを独立したドキュメンテーションとして生成することができるようになりました。

5 バージョン 2012

バージョン 2012 リリース 2

XMLSpy **バージョン 2012 リリース 2** の新機能は以下にリストされています。

- EPUB ファイルとはデジタル形式の出版物 (EPUB books) を配布するためのファイルの集合体のことです。Archive View ではEPUB ファイルを開いたり、EPUB ファイルの作成、デジタルEPUB ブックのプレビュー、EPUB アーカイブのコンポーネントファイルをXMLSpy にて直接編集したり、EPUB ファイルを検証、コンポーネントをEPUB アーカイブへ保存しなおすことができます。詳細についてはEPUB ファイルのセクションを参照ください。
- IBM DB2 論理ファイルのサポート：DB2 データベースのIBM iSeries エディションに含まれる論理ファイルにより、1つ以上のファイルが表すことができます。論理ファイルにより一連のデータや物理ファイルとは異なるフォーマットへアクセスすることができます。IBM iSeries コンピューターへアクセスする場合、論理ファイルにより構成されたデータベースへアクセスすることになる場合があります。このようなデータへのアクセスはこれまで行うことができなかったのですが、バージョン 2012 リリース 2 からサポートされるようになりました。
- コンテキストメニューのコマンドを使用することでXML の特殊な文字をエスケープまたはエスケープ状態から戻すことができます。
- バグの修正

バージョン 2012

XMLSpy **バージョン 2012** の新機能は以下にリストされています。

- プロジェクト内の検索機能により、プロジェクトウィンドウ内にあるファイルやフォルダーを素早く検索
- 入力ヘルパーならびにインテリジェントな編集機能が、既存のCSS 2.1 に加えてCSS 3.0 に対して新たにサポートされました。
- HTML 5.0 のサポート
- データベースへのJDBC 接続サポート
- Java に対応したアプリケーションAPI

6 バージョン 2011

バージョン 2011 リリース 3

XMLSpy **バージョン 2011r3** の新機能は以下にリストされています。

- シンプルになったスペルチェッカー
- Enterprise Edition のXMLSpy において、インターフェース上からXML ファイルのデジタル署名行うことができるようになりました。署名はXML ファイル内部に配置することもできますし、独立したファイルとして保存することもできます。XMLSpy では署名の検証を行うこともできます。
- XMLSpy のスクリプトならびにスクリプトエディターにて .NET バージョン 4.0 が (バージョン 2.0、3.0、3.5 とともに) サポートされるようになりました。

バージョン 2011 リリース 2

XMLSpy **バージョン 2011r2** の新機能は以下にリストされています。

- スキーマビューにて、Altova StyleVision でカスタマイズ可能なSPS を使って生成されるXML スキーマドキュメントのドキュメンテーション。前バージョンのXMLSpy では、標準的なデフォルトデザインによりスキーマドキュメンテーションが生成されていました。この機能により、ドキュメンテーションの構造や書式をユーザーがデザインできるようになります。
- イメージファイルの様な外部ファイルのコンテンツをBase-16 またはBase-64 テキストとしてエンコードし、テキストビューやグリッドビューに挿入。この機能は編集メニューの **挿入 | エンコードされた外部ファイル** コマンドからアクセスできます。
- スキーマビューが以下の機能とともに強化されました：(i) グローバルコンポーネント、属性、同一性制約の並び替え。並べ替えはコンテンツモデルビューにも導入されました。(ii) スキーマファイルや関連するスキーマファイル内にあるグローバルコンポーネントの検索と名前の変更。(ii) 派生型の基底型が編集され、派生型にて定義されたコンテンツ、属性、ファセット、サンプル値が新たな基底型でも許される場合、それらの値を保持することができます。
- WSDL ビューにて、Altova StyleVision 製品でカスタマイズ可能なSPS を使って生成されるWSDL ドキュメントのドキュメンテーション。前バージョンのXMLSpy では、標準的なデフォルトデザインによりWSDL ドキュメンテーションが生成されていました。この機能により、ドキュメンテーションの構造や書式をユーザーがデザインできるようになります。
- チャート機能にスタックされた棒グラス エリアチャート、スタックされたエリアチャート、ローソク足チャート、そしてオーバーレイチャートが追加されました。チャートやテキストビューまたはグリッドビューに表示されたXML から生成することができます。
- XBRL ビューにて、Altova StyleVision 製品でカスタマイズ可能なSPS を使って生成されるXBRL ドキュメントのドキュメンテーション。前バージョンのXMLSpy では、標準的なデフォルトデザインによりXBRL ドキュメンテーションが生成されていました。この機能により、ドキュメンテーションの構造や書式をユーザーがデザインできるようになります。

バージョン 2011

XMLSpy **バージョン 2011** の新機能は以下にリストされています。

- スキーマビューにて、単一のスキーマファイルからスキーマサブセットを作成。スキーマファイルからサブセットへの分割を簡単に行うことができます。それとは反対に、インクルードされたスキーマサブセットのコンポーネントを組み合わせて、アクティブなファイルへの同化もサポートされます。
- テキストビューまたはグリッドビューにおいて、XML ドキュメント内のデータから (円グラフや棒グラフのような) チャートの生成 (Enterprise Edition のみ)。更に、XSLT/XQuery プロファイラーの結果もチャートの形式で表示することができます (Enterprise Edition のみ)。

- WSDL に対するSOAP メッセージの検証。SOAP メッセージの検証を、SOAP 仕様だけではなく、WSDL 定義により参照されている任意のXML スキーマに対して行うことができます (Enterprise Edition のみ)。

7 バージョン 2010

バージョン 2010 リリース 3

XMLSpy **バージョン 2010-3** の新機能は以下にリストされています。

- スキーマビューにおいてスキーマルールを作成し、新たなスキーマ制約を指定 (Enterprise Edition のみ)。
- XML スキーマから生成されたサンプルファイル内のサンプル値を生成 (Enterprise ならびに Professional Edition のみ)。
- Microsoft Visual Studio 2010 との統合。バージョン 2005 と 2008 と共に、最新版の Visual Studio がサポートされます。Visual Studio バージョン 2003 へのサポートは継続されません。

バージョン 2010 リリース 2

XMLSpy **バージョン 2010-2** の新機能は以下にリストされています。

- テキストビューならびにグリッドビューにて巨大なファイルを扱う際の最適化。
- テキストビューの XML スキーマ検証の (最大で 3 倍にもなる) スピード向上。
- SharePoint(C) サーバーサポート XMLSpy のインターフェースから直接ファイルのチェックインやチェックアウトを行うことが可能に。
- 情報ウィンドウ内の CSS と HTML により GUI による CSS や HTML ファイルのリンク作成や、外部ブラウザーやエディターなどで開くことが可能に。
- 生成された C++ コードの Linux でのサポート。
- 64 ビット版の XMLSpy が Enterprise ならびに Professional Edition でご利用になれます。

Altova XMLSpy 2017 Enterprise Edition

RaptorXML Server

RaptorXML Server

(Raptor または RaptorXML とも呼ばれる) Altova RaptorXML(+XBRL) Server は、Altova の第 3 世代の非常に高速な XML と XBRL* プロセッサーです。最新の標準や並列計算環境における使用に最適化される形で開発されました。クロスプラットフォームにて作動するようにデザインされ、今日利用可能な複数 CPU による計算能力を引き出すことにより、非常に高速な XML 並びに XBRL データの処理が実現します。

RaptorXML には2つのエディションがあります:

- RaptorXML Server は、ネットワークからアクセスすることができ、一度に複数のファイルを変換することのできます。
- ネットワークからアクセスすることができ、一度に複数のファイルを変換することのできる RaptorXML+XBRL Server Edition は更に XBRL 検証をサポートします。

RaptorXML をコマンドラインから実行することができ、COM、Java、.NET, および Python のためのインターフェイスがあります。ですから、RaptorXML を XML ドキュメントの検証、XSLT 変換処理、XQuery ドキュメントの実行環境から使用することができます。

XML および XBRL* ドキュメントを検証するために XMLSpy GUI から Raptor サーバーを使用することもできます。

* **メモ:** XBRL 処理は RaptorXML+XBRL Server でのみで使用することができます。RaptorXML Server では使用することができません。

RaptorXML についての詳細は Altova Web サイトを参照してください。

Altova XMLSpy 2017 Enterprise Edition

XMLSpy チュートリアル

XMLSpy チュートリアル

このチュートリアルではXML の概要を理解し、数々の重要なXML のタスクを体験することができます。これにより、XMLSpy の最も重要な機能を学習します。

このチュートリアルは以下のように構成されます：

- XMLSpy インターフェイス。アプリケーションのグラフィカルユーザーインターフェイス（GUI）を詳しく理解することができます。
- XML スキーマの作成。XMLSpy の直感的なスキーマビューにて XML スキーマを作成する方法、ドラッグアンドドロップにより複雑なコンテンツモデルを作成する方法、そしてスキーマの設定方法を学習します。
- スキーマビューの機能を使用。複合型ならびに単純型、グローバルな要素参照、そして属性の列挙の作成方法を学習します。
- スキーマビューを使ったスキーマのナビゲーション方法、スキーマのナビゲーション方法を学習します。
- XML ドキュメントの作成。スキーマをXML ドキュメントに割り当て、グリッドビューやテキストビューを使ってXML ドキュメントを編集する方法、そしてXMLSpy に組み込まれているバリデーターを使用して、XML ドキュメントを検証する方法を学習します。
- XSLT スタイルシートを使用してXML ファイルを変換する。XSLT ファイルを割り当て、XMLSpy に組み込まれたXSLT エンジンを使用することで変換を行います。
- XMLSpy プロジェクトにて作業することにより、XML ドキュメントを簡単に整理することができます。

インストールと設定

このチュートリアルは、あなたのコンピューターにXMLSpy がインストールされており、無料評価キーコードを受け取っているか、登録されたユーザーであることを前提としています。XMLSpy の評価版の機能は製品版と変わりませんが、その使用は30日間に限定されています。XMLSpy の一般ライセンスは、Altova のセキュアーウェブサーバ、または販売店からお求めになれます。

チュートリアルファイル

チュートリアル ファイルはアプリケーション フォルダにあります：

```
C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples
\Tutorial
```

`Examples` フォルダーには色々なことを試すためのXML ファイルが配置されており、`Tutorial` フォルダーには、このチュートリアルで使用される全てのファイルが含まれています。

アプリケーションフォルダー内の `Template` フォルダー（通常は`c:\Program Files\Altova` になります）には **「ファイル | 新規作成」** メニューオプションから選択できる全てのXML テンプレートファイルが収められています。これらのファイルには必要なデータ（名前空間とXML 宣言）が記述されており、すぐに作業を始められるようになっています。

1 XMLSpy インターフェイス

チュートリアルのこのセクションで、XMLSpy を開始してインターフェイスについて理解することができます。

XMLSpy の開始

XMLSpy を開始するには、デスクトップの XMLSpy アイコンをダブルクリックするか、または **[スタート] | [すべてのプログラム]** メニューを使用して XMLSpy プログラムにアクセスします。XMLSpy は、ドキュメントがインターフェイス上で開かれていない状態で開始されます。XMLSpy を開きます。

インターフェイスの概要

XMLSpy インターフェイスのデフォルトのビューは3つの垂直のエリアにより構成されています (下のフィギュア) の 3 つのウィンドウにより構成されています : (i) プロジェクトと情報ウィンドウ; (i) メインと出力ウィンドウ; そして (ii) 入力ヘルパーウィンドウ。プロジェクトウィンドウを確認してください。 XMLSpy を最初に開始した時にデフォルトで開かれる、サンプルプロジェクトが表示されています。

以下は、インターフェイスのレイアウトと多種のコンポーネントの機能を理解するために役に立つ主要なポイントです。チュートリアルのこの最初のパートのサブセクションはインターフェイスに慣れ親しむために役に立ちます。

メインウィンドウ内のドキュメントバー : 複数のドキュメントが開かれている場合、各ドキュメントはメインウィンドウのドキュメントバーのタブに表示されています (*図を参照*)。タブをクリックすることでドキュメントをアクティブ化することができます。ドキュメントバーの右側の矢印をクリックすることでドキュメントタブをスクロールすることができます。(例えば、Examples プロジェクトから) 2 つ以上のファイルを開き、タブの動作について確認してください。

ドキュメント編集ビュー : アクティブなドキュメントは適用することのできる複数の編集ビューの 1 つにより表示することができます。例えば :

- XML (`.xml`) ドキュメントはテキストビュー、グリッドビュー、Authentic ビューで表示されますが、スキーマビューなどの他のビューでは表示することができません。
- XML スキーマ (`.xsd`) ドキュメント、他方、テキストビュー、グリッドビュー、スキーマビュー、ブラウザービューで表示されますが、Authentic ビューでは表示することができません。

以下のビューを使用することができます : テキストビュー、グリッドビュー、スキーマビュー、WSDL ビュー、XBRL ビュー、Authentic View- ビュー 、および ブラウザービュー。

入力ヘルパー: 入力ヘルパーウィンドウはアクティブなドキュメント (例えばXML またはXSD またはCSS またはWSDL) および、現在アクティブなドキュメントビュー (例えば、テキストビューまたはスキーマビュー) に従い変更されます。入力ヘルパーによりアクティブなドキュメントをコンテキストに敏感な編集サポートにより素早くかつ正確に編集することができます。

(C) 2017 Altova GmbH

1.1 表示

チュートリアルのこのパートでは以下を学ぶことができます：(i) ドキュメントの編集ビューの切り替え、および (ii) 特定のドキュメントの型のデフォルトの編集ビューの変更。

ドキュメントビューの切り替え

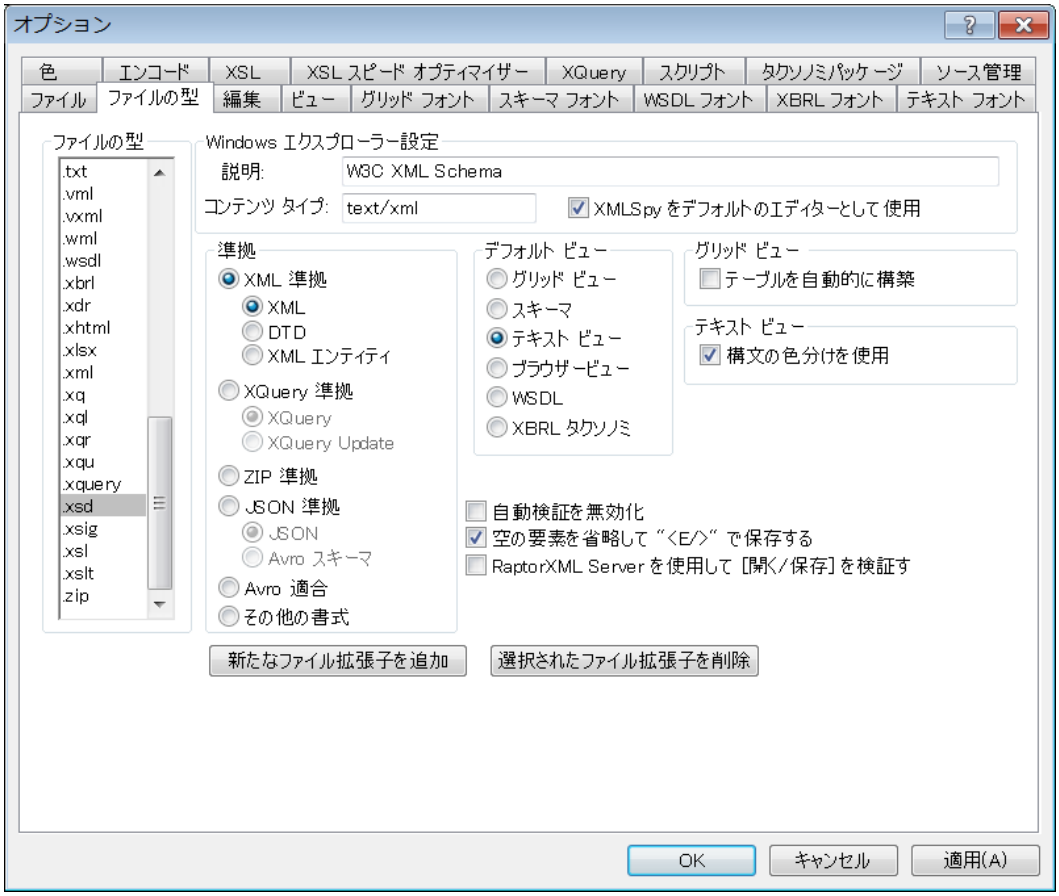
ドキュメントを開くと、そのドキュメントの型のために設定されたデフォルトのビューで開かれます。ドキュメントを以下の方法で開きます：

1. **「ファイル | 開く」**コマンドをクリックします。
2. `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\Tutorial` フォルダーにあるファイル `Address.xsd` を参照し選択します、**開く**をクリックするとファイルがスキーマビューで開かれます。
3. メインウィンドウの下にあるビュータブをクリックして多種のビュー（テキストビュー、グリッドビュー、など）を切り替えることができます。XML スキーマドキュメントはテキストビュー、グリッドビュー、スキーマビューおよびブラウザービューで表示されます。
4. **「表示」**メニューのオプションからビューのオプションを選択してビューを変更することができます。**「表示」**メニューコマンドを使用して `AddressFirst.xsd` ドキュメントのビューを切り替えてください。
5. **「ファイル | 閉じる」**を使用してドキュメントを閉じてください。

ドキュメントの型のデフォルトのビューの変更

`.xsd` 拡張子を持つすべてのドキュメントは、デフォルトでスキーマビューで開かれます。全ての型のドキュメントのデフォルトで開かれるビューをオプションダイアログで変更することができます。ここでは、`.xsd` ドキュメントを使用して行います。

1. **「ツール | オプション」**コマンドをクリックし、ファイルタイプタブに移動します（下のスクリーンショット参照。）。
2. 「ファイルタイプ」ペインで、`.xsd` にスクロールして選択します（*スクリーンショットでハイライトされています*）。
3. 「デフォルトのビュー」ペインでテキストビューを選択します。



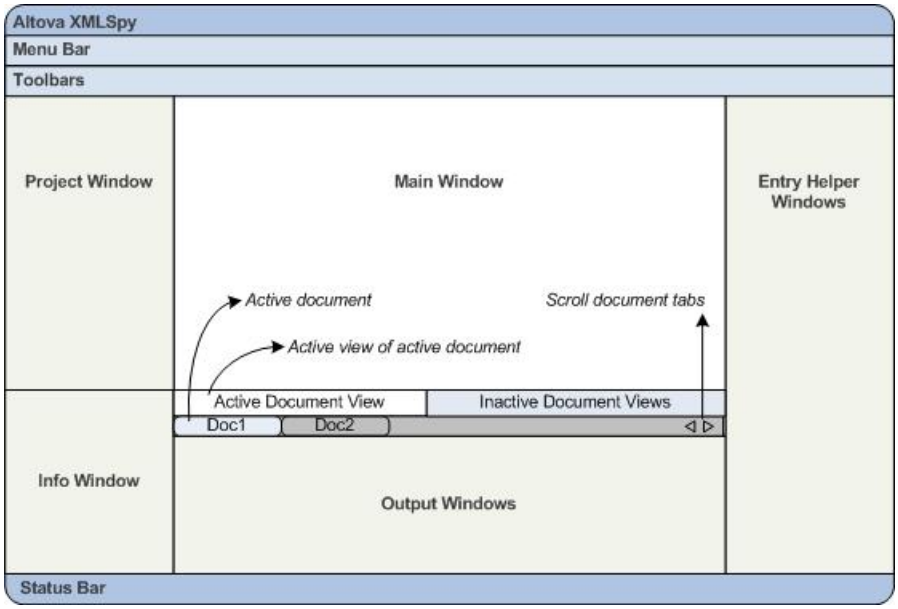
4. **「OK」**をクリックします。
5. **「ファイル | 開く」**コマンドをクリックして、`AddressFirst.xsd` ファイルを開きます。ファイルはテキストビューで開かれます。
6. スキーマビューに切り替えて、ファイルをこのビューで表示し、ファイルを **「ファイル | 閉じる」**を使用して閉じます。
7. **「ツール | オプション」**を使用して、オプションダイアログに戻り、ファイルタイプタブで `.xsd` ファイルのデフォルトのビューをスキーマビューに戻します。

メモ: オプションダイアログのファイルタイプのタブで (*上のスクリーンショット*) リストされるファイルの拡張子のデフォルトのビューを変更することができます。ファイルの拡張子は **「新たな拡張子を追加」**ボタンを使用してリストに追加することができます。

1.2 ウィンドウ

デフォルトでは、複数のウィンドウがメインウィンドウの周りに表示されて (*下のスクリーンショット参照*) 次のウィンドウのグループに整理されています :

- プロジェクトウィンドウ
- 情報ウィンドウ
- 入力ヘルパー (現在アクティブなドキュメントの型により複数存在します)
- 出力ウィンドウ : メッセージ、チャート、XPath、XSL アウトライン、ファイルの中を検索、スキーマの中を検索、、XBRL 内を検索



このセクションでは、ウィンドウのグループの表示のオンとオフ、ウィンドウのスクリーン上の移動方法の説明を行います。ウィンドウの表示の管理は、インターフェイス上のスペース管理が必要な場合役に立ちます。

ウィンドウグループの表示のオンオフの切り替え

ウィンドウグループ (プロジェクトウィンドウ、情報ウィンドウ、入力ヘルパー、出力ウィンドウ) は、**ウィンドウ** メニューのコマンドでオンとオフを切り替えることにより、表示または非表示にすることができます。表示されたウィンドウグループはタイトルバーを右クリックしてコマンド **非表示** を選択することで非表示にすることができます。非表示のウィンドウは **ウィンドウ** メニューを使用してのみ表示することができます。

`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\Tutorial` フォルダー内の XML ファイルを開き、基本のコマンドの使用を練習し、コマンドの作動方法に慣れ親しんでください。ウィンドウグループの表示と非表示に関しての詳細は XMLSpy インターフェイスのセクションを参照してください。

スクリーン上でウィンドウを移動する

個別のウィンドウは、インターフェイス上を自由にフロートまたはインターフェイス内でドッキングさせることができます。ウィンドウはウィンドウグループ内のタブとしてドッキングすることができます (*ウィンドウグループは上で説明されています*)。例えば、下のスクリーンショットでは、3つのタブ (グローバルウィンドウ、名前空間ウィンドウ、ID 制約) 付きのウィンドウを持つ、コンポーネントの入力ヘルパーがスキーマビューで表示されています。

ウィンドウは、以下の方法を用いてフロートまたはドッキングすることができます：

- ウィンドウのタイトルバーをダブルクリックします。ドッキングすると、ウィンドウはフロートすることができます。フロートしていると、ウィンドウは最後にドッキングしていた箇所にドッキングします。
- ウィンドウのタイトルバーを右クリックして、必要なコマンド (**フロート** または **ドッキング**) を選択します。
- (タイトルバーをハンドルとして) ウィンドウをドッキングされている場所からドラッグしてフロートさせます。フロートしているウィンドウを (タイトルバーをハンドルとして) ドラッグしてドッキングする場所にドラッグします。 2つのセットの青い矢印が表示されます。外側の4つの矢印のセットにより、アプリケーションウィンドウ (GUI の上、右、下、および左端) に相対してウィンドウをドッキングすることができます。内側の矢印のセットは、カーソルが現在置かれているウィンドウに相対してウィンドウをドッキングすることができます。ドラッグされたウィンドウを内側の矢印のセット (または、ウィンドウのタイトルバーの上) の中心の下にドロップすることで、ドラッグされたウィンドウをドロップされたウィンドウ内のタブされたウィンドウとしてドッキングすることができます。

タブされたウィンドウをフロートするには、そのタブをダブルクリックします。タブされたウィンドウをタブされたグループからドラッグするには、タブをドラッグします。

スキーマビュー内で、`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\Tutorial` フォルダーからXML スキーマファイルを移動させる練習をするには、上記の方法でインターフェイス内でのウィンドウの移動に慣れ親しんでください。

1.3 メニューとツールバー

チュートリアルのこのセクションでは、XMLSpy のメニューとツールバーの機能を学習します。

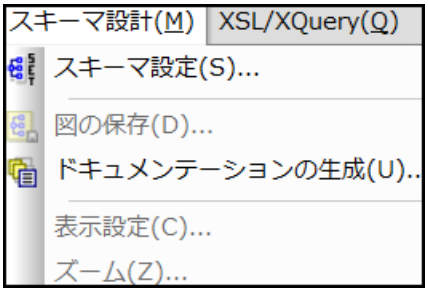
メニュー

2 つのメニューバーがあります：(i) ドキュメントが開かれていない時に表示されるデフォルトメニュー、および (ii) ドキュメントが開かれると表示される XMLSpy アプリケーションメニュー全体。以下を行ってください：

1. メニューコマンド **「ファイル | 閉じる」** を使用してすべての開かれているドキュメントを閉じます。デフォルトのメニューが表示されます。
2. `AddressFirst.xsd` ファイルを **「ファイル」** メニューの下にある最近開かれたファイルのリストからクリックして開きます。ファイルがスキーマビューで開かれている場合、メニューは XMLSpy アプリケーションメニューに変更されます。

メニューは主に機能により整理されており、メニュー内のコマンドは、カーソルポイントまたはアクティブドキュメントの現在のビューの選択の場合のみ実行することができます。メニューコマンドが有効化または無効化されているかを決定する要素を理解するためには以下を行います：

1. **「スキーマデザイン」** メニューをクリックします。 **「図の保存」**、 **「表示設定」**、および **「ズーム」** コマンドが無効化されていることに注意してください (*下のスクリーンショット*)。

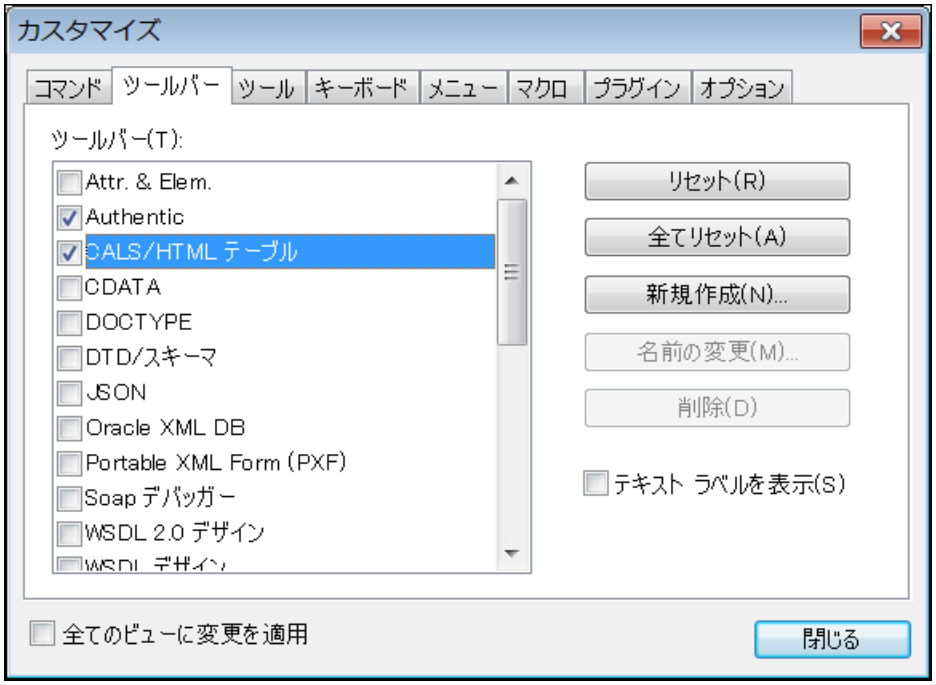


2. メニューの外側の空白をクリックして、メニューを消します。要素コンポーネントの左横にある **「図の表示」** アイコン をクリックします。これによりスキーマビューのコンポーネントモデルビューに移動します (スキーマビューの 2 番目のビュー；1 番目のビューはスキーマ概要です)。スキーマデザインメニューを確認すると、 **「図の保存」**、 **「表示設定」**、および **「ズーム」** コマンドが有効化されていることが確認できます。これらのコマンドはスキーマビューのコンテンツモデルビュー内のみで有効化されますが、スキーマビューのスキーマ概要またはその他のビューでは有効化されません。
XML スキーマファイルはスキーマビューのみで開かれることに注意してください。
3. XML スキーマファイルもまた XML ファイルですから、このファイルもまたテキストビューおよびグリッドビューで XML ファイルとして表示されます。XML ファイルに対して適用することのできるメニューコマンドすべてはこれらのビューで有効化されます。スキーマビューとテキストビュー内の **「編集」** メニューのコマンドを (有効化または無効化されているかを) 比較してください。
4. 次に、テキストビューとグリッドビューの **「XML | 挿入」** メニュー内のコマンドを (有効化または無効化されているかを) 比較してください。このメニュー内のコマンドはグリッドビューのみで有効化されています。

全てのメニューコマンドの詳細に関しては、ユーザーマニュアルの「ユーザーレファレンス」のセクションを参照してください。

ツールバー

ツールバーの表示は現在のビューにより異なります。アプリケーションのデフォルトの設定は各ビューの正しいツールバーを表示します、ツールバーは各ビューによって異なります。ですが、カスタマイズダイアログ内のツールバー (**「ツール | カスタマイズ | ツールバー」**、*下のスクリーンショット*) によりカスタム化することができます。



ツールバーをGUI の周辺で移動させる練習をしてください。ツールバーのハンドルをクリックして、GUI 内の希望する場所にドラッグします。 (ツールバーハンドルは各ツールバーの左横にある垂直の点線で表示されています。 *下のスクリーンショット参照*)。

以下の場所にツールバーをドラッグします : (i) ツールバーエリア内の他のライン、(ii) 他のツールバーの左側または右側、(iii) メインウィンドウの中央、 (iv) アプリケーションウィンドウの左側または右側 (この操作には、グラブするハンドルがアプリケーションウィンドウの左側または右側の上に置かれる必要があります)。

終了すると`AddressFirst.xsd` ファイルを閉じます。

1.4 テキストビューの設定

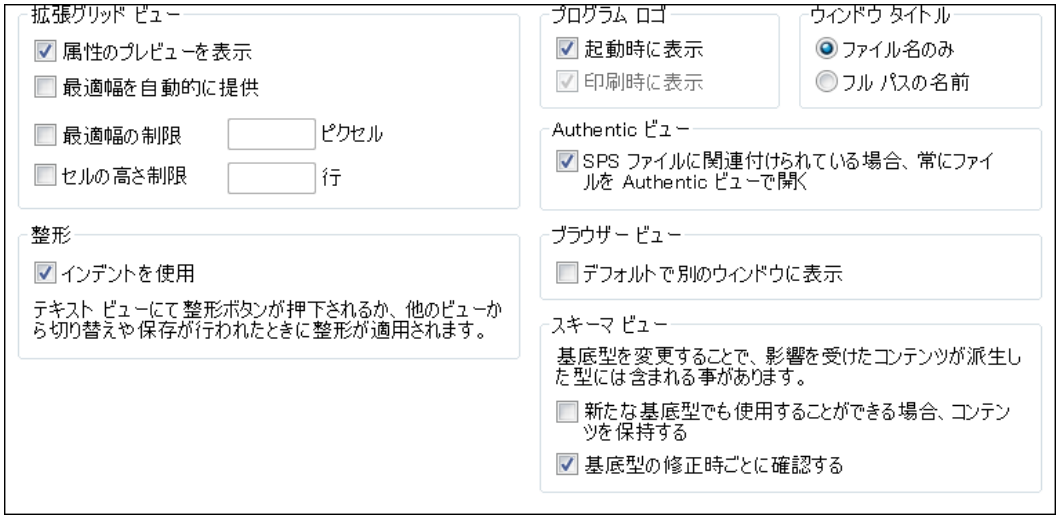
このセクションでは、再フォーマットされたドキュメントの設定および編集中のブックマークの使用方法を学習します。ドキュメントが再フォーマットされると、テキストビューで表示され、XML 階層の各下のレベルが前のレベルに比較してインデントされて表示されます（*下のスクリーンショット参照*）。ブックマークにより、ドキュメントの位置をマークし、戻る場所を素早く検索することができます。

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <Company>
3       <Address xsi:type="US-Address">
4           <Name>US dependency</Name>
5           <Street>Noble Ave.</Street>
6           <City>Dallas</City>
7           <Zip>04812</Zip>
8           <State>Texas</State>
9       </Address>
10      <Person Manager="true" Degree="BA" Programmer="false"
11          <First>Fred</First>
12          <Last>Smith</Last>
13          <PhoneExt>22</PhoneExt>
14          <Email>Smith@work.com</Email>
15      </Person>
16  </Company>
```

再フォーマット

再フォーマットには 2 つのステップがあります：(i) 再フォーマットをオンにして、インデントの数を設定します、そして (ii) 再フォーマットを適用します。

1. `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\Tutorial` フォルダー内にあるファイル `CompanyFirst.xml` を開きます。（XML ドキュメントを表示するデフォルトの設定がテキストビューでない場合、テキストビューに切り替えます）。
2. オプションダイアログ（**ツール | オプション | ビュー**」*下のスクリーンショット*）ビュータブで「インデントを使用」チェックボックスをチェックします。これによりインデントのある再フォーマットが（デフォルトの設定として）オンになります。終了すると「**OK**」をクリックします。この設定はテキストビューで開かれるすべてのファイルに適用されます。



3. テキストビュー設定ダイアログを「**表示 | テキストビューの設定**」コマンドを使用して（*下のスクリーンショット*）開きます。タブペインでタブサイズを 3 に減らします。タブの挿入の率ボタンのデフォルトの選択をそのままにしておいてください。これにより再フォーマットのインデントが（スペースではなく）タブになるように設定されます、各タブは、3 つの空白の幅になります。終了すると「**OK**」をクリックします。
4. メニューコマンド「**編集 | 再フォーマット**」をクリックします。ドキュメントは、新しいタブ値を持つように再フォーマットされます。

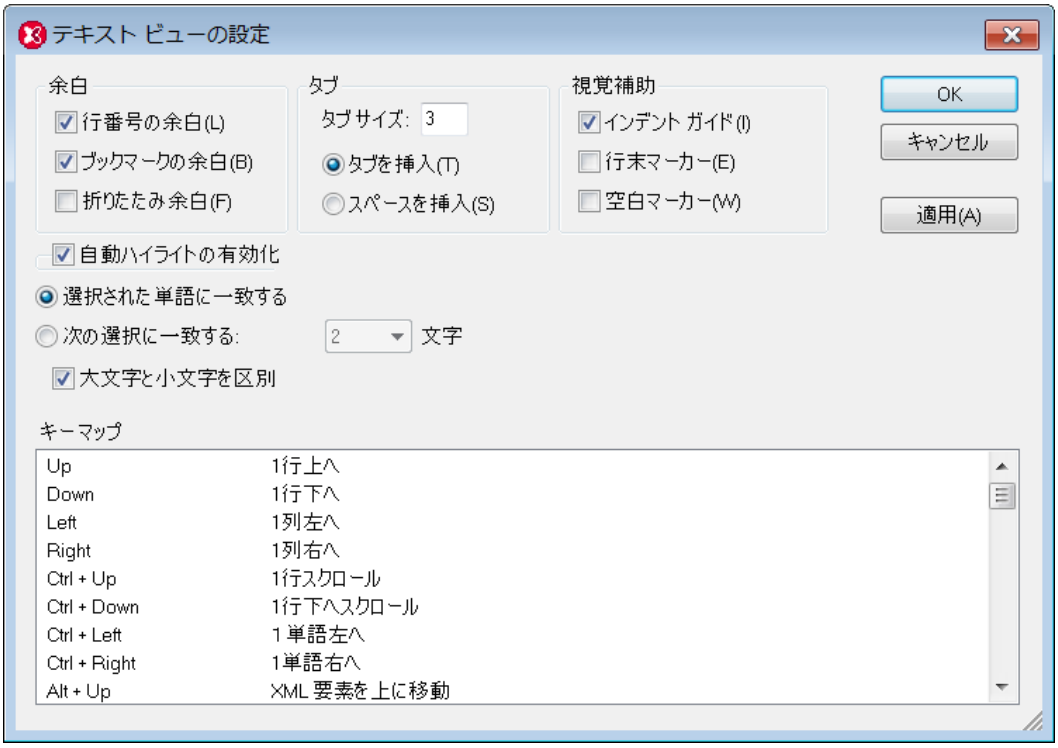
5. テキストビュー設定 ダイアログ（**表示 | テキストビューの設定**」）を開いて、視覚補助ペイン内の行末マーカーを解除します。
6. テキストビュー内で、行末に移動して、行末マーカーを削除することにより、次のラインが上に移動します。
7. グリッドビューに切り替えて、再度 テキストビューに切り替えます。ドキュメントは、再フォーマットされ、削除された行末マーカーが回復されます。

メモ: オプションダイアログのビュータブ内で（**「ツール | オプション」 | ビュー**、*上のスクリーンショット*）、インデントの使用のチェックボックスのチェックを解除すると、すべての行はインデント無しに再フォーマットされます。

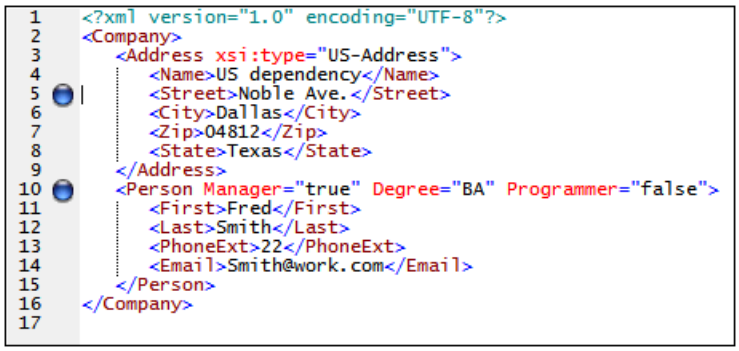
ブックマーク

マークする行の左側のブックマークマージンにブックマークを表示することができます。ドキュメントの行をブックマークして素早い参照やアクセスを行うことができます。

1. テキストビューの設定（**表示 | テキストビューの設定**」、*のスクリーンショット*）で、ブックマージンのオプションがマージンペインで選択されていることを確認します。完了すると **「OK」**をクリックします。



2. ファイル `CompanyFirst.xml` のテキストビューで、ブックマークするラインにカーソルをポイントし、メニューコマンド **「編集 | ブックマークの挿入 削除」**を選択します。ラインはブックマークされ、ブックマークマージン内に青いブックマークで表示されます（*下のスクリーンショット参照*）。
3. ステップ 2 と同様に、他のラインのブックマークを作成します。



4. 「**F2**」（または コマンド **編集 | 次のブックマークへ移動**」）を押して、ドキュメント内の次のブックマークへ移動します。 「**Shift+F2**」（または コマンド **編集 | 次のブックマークへ移動**」）を押して、ドキュメント内の前のブックマークに移動します。この2つのコマンドを制限なく繰り返すことができます。
5. ブックマークされたラインにカーソルをポイントして、メニューコマンド **編集 | ブックマークの挿入 削除**」を選択すると ブックマークは削除されます。
6. 保存してファイルを閉じます。ブックマークの情報はファイルに保存されないか、ファイルを再度開き確認してください。

2 XML スキーマ :基本

XML スキーマには XML ドキュメントの構造が記述されます。XML ドキュメントがスキーマ内で指定された要件に合致するかを検証するために XML スキーマは使用されます。合致する場合、その XML ドキュメントは**妥当**であると言われ、そうでない場合は**不正**であると言われます。XML スキーマを使って XML ドキュメントで許される構造とコンテンツをデザインすることができ、XML ドキュメントが妥当かどうか検証することができます。

XML スキーマドキュメントの構造と構文は複雑で、それ自体が XML ドキュメントになっています。XML スキーマそのものも XML スキーマの仕様に従った妥当なものでなければなりません。XMLSpy では、スキーマビューで使えるドラッグアンドドロップにより、妥当な XML スキーマを簡単に作成することができます。作成された XML スキーマドキュメントはテキストビューやグリッドビューでも編集可能ですが、スキーマビューを使ったほうが作成や編集をより簡単に行うことができます。

目的

このセクションでは、スキーマビューを使っての XML スキーマの編集方法を学習します。特に、以下のことがらについて学習します :

- スキーマファイルを新たに作成する
- スキーマの名前空間を定義する
- 基本的なコンテンツモデルを定義する
- コンテキストメニューとドラッグアンドドロップを使ってコンテンツモデルに要素を追加する
- コンテンツモデルビューを設定する

基本的なスキーマが完成したら、%XMLSPY%> による XML スキーマのさらに高度な機能について次のセクションで学習します。その次には %XMLSPY%> のスキーマナビゲーションとドキュメンテーションに関するセクションが続きます。

このセクションで使用されるコマンド

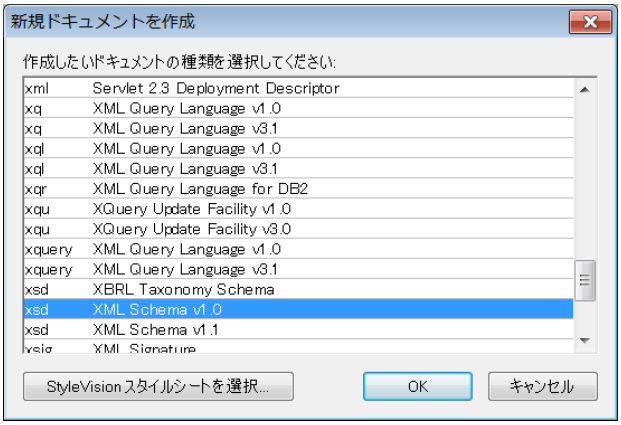
このセクションではスキーマビューだけを使用します。以下のコマンドが使われます :

ダイアグラム表示（または コンテンツモデルビュー表示）。このアイコンはスキーマ概要にある全てのグローバルコンポーネントの左側に配置されています。このアイコンをクリックすることで、対応するグローバルコンポーネントのコンテンツモデルを表示します。

2.1 新しい XML スキーマファイルの作成

新しいXML スキーマファイルの作成 :

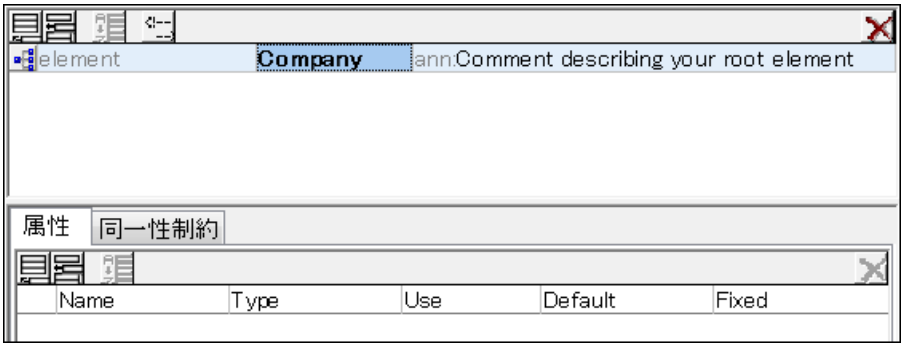
1. メニューオプションから **「ファイル | 新規作成」** を選択します。 **「新規ドキュメント作成」** ダイアログが開きます。



2. ダイアログから、 `xsd (XML Schema v1.0)` エントリ (ウィンドウ内のドキュメントの詳細とリストはスクリーンショットと異なる場合があります。) を選択し、 **「OK」** で確定します。空のスキーマファイルがウィンドウのスキーマビューに表示されます。
3. スキーマ設計ツールバーから **「XSD 1.0」** モードボタン (*下のスクリーンショット参照*) をクリックして、スキーマビューがXSD 1.0 編集モードになるように設定します。ます。

4. ルート要素を入力するよう求められます。

5. ハイライトされたフィールドをクリックして、`Company` と入力します。 **「Enter」** キーを押下することで確定します。`Company` がこのスキーマのルート要素になり、グローバル要素として作成されます。メインウィンドウで表示されるビュー (*下のスクリーンショット*) はスキーマ概要と呼ばれます。メインウィンドウの上部に全てのグローバルコンポーネントのリストを表示することでスキーマの概要を提供し、下部には選択されたグローバルコンポーネントの属性と同一性制約が表示されます。 (個々のグローバルコンポーネントのコンテンツモデルの表示と編集は、グローバルコンポーネントの左にあるダイアグラム表示アイコンから行うことができます。)



6. 要素の注釈フィールド (ann) にて、`Company` 要素の説明を入力します。ここでは `Root element` と記述し

ます。

7. メニューオプション **「ファイル | 保存」** をクリックして、好きな名前 (例えば `AddressFirst.xsd`) でXMLスキーマを保存します。

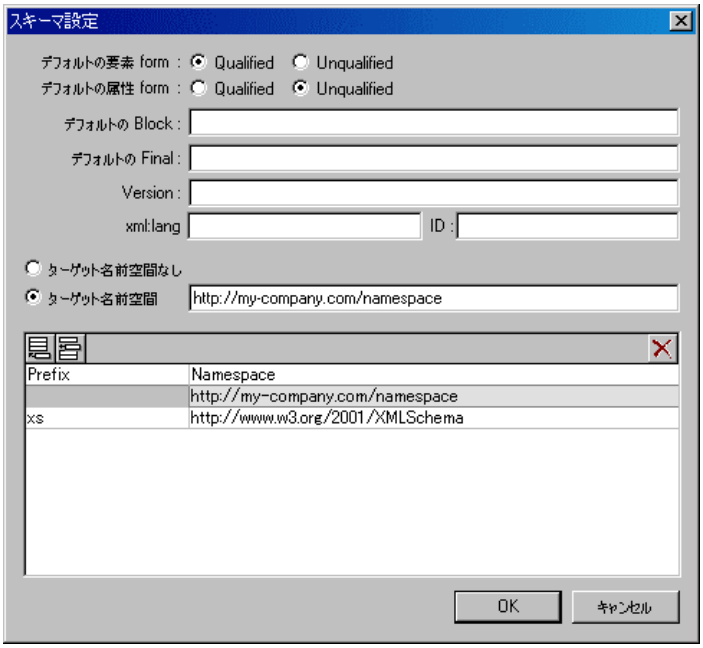
2.2 名前空間の定義

XML 名前空間はXML スキーマとXML ドキュメントにとって重要な課題です。XML スキーマドキュメントはXML スキーマ名前空間を参照する必要があり、XML ドキュメントインスタンスのためにターゲット名前空間を定義することができます。スキーマデザイナーとしては、(基本的には、どのプレフィックスを使用して)これらの名前空間をどのように定義するか決定する必要があります。

作成中のXML スキーマ内で、XML ドキュメントインスタンスのためにターゲット名前空間を定義します。(XML スキーマドキュメントを新規作成すると、XML スキーマ名前空間に対する必要なレファレンスは自動的にXMLSpy により作成されます。)

ターゲット名前空間の作成方法：

1. **「スキーマ設計 | スキーマ設定」**をメニューオプションから選択します。これによりスキーマ設定ダイアログが開かれます。



2. ターゲットの名前空間のラジオボタンをクリックして、`http://my-company.com/namespace` を入力します。XMLSpy では、ターゲットの名前空間として入力された名前空間が XML スキーマドキュメントのデフォルトの名前空間として作成され、ダイアログの下のペインの名前空間のリストに表示されます。
3. **「OK」**ボタンを押して、確認します。

注意点：

- XML スキーマの名前空間は、XMLSpy により自動的に作成され、`xs:` のプレフィックスが与えられます。
- XML ドキュメントインスタンスが作成されると、ドキュメントが妥当であるために、スキーマ内で定義されたターゲット名前空間が必要になります。

2.3 コンテンツモデルの定義

スキーマ概要には、既に作成された Company という名前のグローバル要素が表示されます。この要素は1つの Address 要素と、出現数に制限の無い Person 要素（コンテンツモデル）を持つことになります。グローバルコンポーネントの中の要素、複合型、そして要素グループはコンテンツモデルを持つことができます。

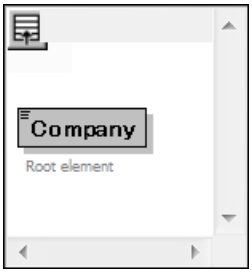
XMLSpy において、グローバルコンポーネントのコンテンツモデルは、スキーマビューのコンテンツモデルビュー内に表示されます（以下のスクリーンショットを参照）。グローバルコンポーネントのコンテンツモデルを編集するには、グローバルコンポーネントの左にあるダイアグラム表示アイコン をクリックします。

このセクションでは、Company 要素のコンテンツモデルを作成します。

基本的なコンテンツモデルの作成

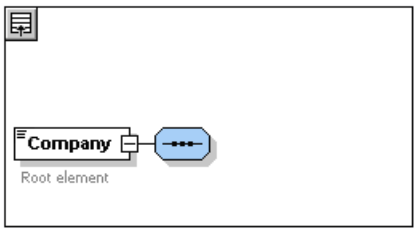
Company 要素のコンテンツモデルを作成するには：

1. スキーマ概要にて、要素のダイアグラム表示アイコン をクリックします。今のところは空の Company 要素のコンテンツモデルが表示されます（以下のスクリーンショットを参照）。その他に、コンポーネント入力ヘルパーにある Company エントリをダブルクリックしてコンテンツモデルを表示させる方法もあります。

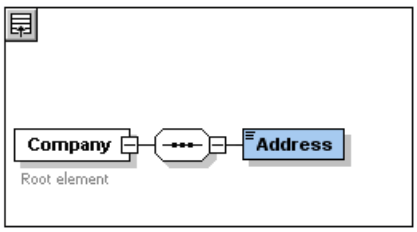


2. コンテンツモデルはコンポジッターとコンポーネントにより構成されます。コンポジッターは2つのコンポーネントの関係を指定します。Company コンテンツモデルでは、子要素を加えるために子コンポジッターを追加する必要があります。コンポジッターを追加するには、Company 要素を右クリックし、コンテキストメニューから「子を追加 | シーケンス」を選択します。（「シーケンス」、「選択」、そして「全て」は、コンテンツモデルで使用される3つのコンポジッターです）

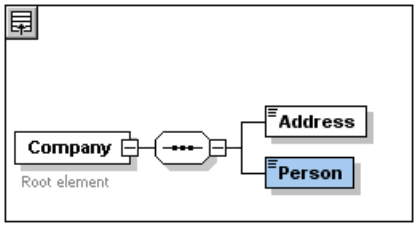
この操作によりシーケンスが指定され、そこから表示されるコンポーネントを定義するシーケンスコンポジッターが挿入されます。



3. シーケンスコンポジッターを右クリックして、**「子を追加 | 要素」**を選択します。名前の無い要素コンポーネントが追加されます。
4. 要素の名前に`Address` と入力して **「Enter」**で確定します。

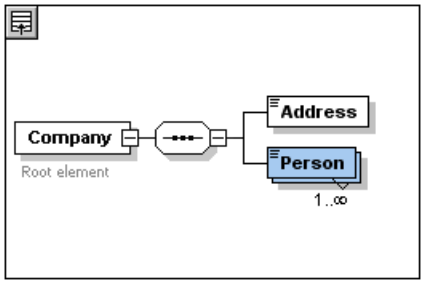


5. シーケンスコンポジッターを再度右クリックして、**「子を追加 | 要素」**を選択します。新たに作成された要素コンポジットの名前を`Person` にします。



これで1つの会社あたり、1つの住所と1人の人物の記述を許可するスキーマを定義しました。これからさらに Person 要素の数を増やす必要があります。

6. Person 要素を右クリックし、コンテキストメニューから無制限を選択します。ダイアグラムの Person 要素に許された出現回数として1から無限が表示されます。



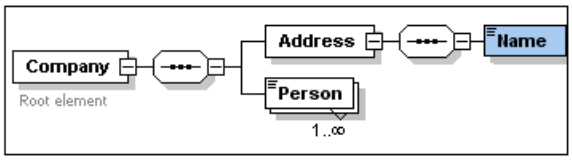
その他に、詳細入力ヘルパー内から minOcc と maxOcc フィールドを編集して、許可された出現回数を指定することができます。この場合なら、それぞれ1とunbounded になります。

コンテンツモデル構造に新たな階層を追加

これまでで作成した基本的なコンテンツモデルは、1つのレベル（Address と Person 要素というcompany 要素の子レベル）しか持ちませんでした。ここでは Address 要素の内容を定義します。Address 要素以下に Name、Street、と City 要素を配置します。これは第2のレベルです。再度子コンポジッターを Address 要素に加え、要素コンポーネントを加えることになります。

やり方：

1. Address 要素を右クリックしコンテキストメニューを開き、「子を追加 | シーケンス」を選択します。これでシーケンスコンポジッターが追加されます。
2. シーケンスコンポジッターを右クリックし、「子を追加 | 要素」を選択します。新たに作成された要素コンポーネントの名前を Name に変更します。



複合型、単純型、スキーマデータ型

ここまで、要素の型を明示的に定義はしませんでした。テキストタブをクリックして、スキーマをテキストビューに表示させてください（下記参照）。シーケンスコンポジッターが挿入された場所には xs:complexType 内に xs:sequence 要素が挿入されているのが分かると思います。簡単に説明すると、Company そして Address 要素は子要素を持っているため複合型になります。複合型要素は属性または要素を持ちます。

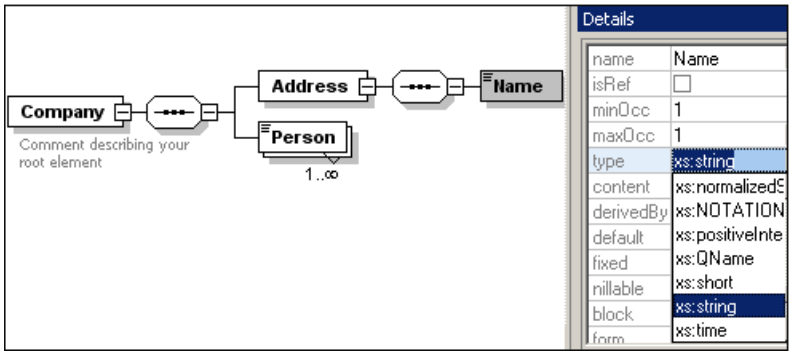
```
<xs:element name="Company">
  <xs:annotation>
    <xs:ドキュメント>Root element</xs:ドキュメント>
```

```
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Address">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Name"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Person"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

一方で、単純型要素は属性を持たずテキストだけを含みます。テキストとは、文字列や日付、数値などです。Address の子である Name にはテキストしか入って欲しくありません。つまり文字列だけに制限された単純型ということになります。XML スキーマデータ型を xs:string にすることで、このような設定を行うことができます。

Name 要素のデータ型を定義するには：

1. スキーマタブをクリックし、スキーマビューに戻ります。
2. Name 要素をクリックして選択します。
3. type 詳細入力ヘルパーにて、Name コンボボックスのドロップダウンメニューから xs:string を選択します。



minOcc と maxOcc は1の値を持っていて、この要素は1度しか現れないことに注意してください。

Name 要素のテキスト表記は以下のようになります：

```
<xs:element name="Name" type="xs:string"/>
```

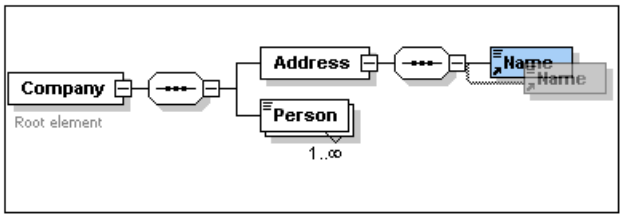
注意 : 単純型要素は幾つかある XML スキーマデータ型のどれでも取ることができます。全ての場合において、要素ボックスにはテキストコンテンツであることを示すアイコンが表示されます。

2.4 ドラッグアンドドロップによる要素の追加

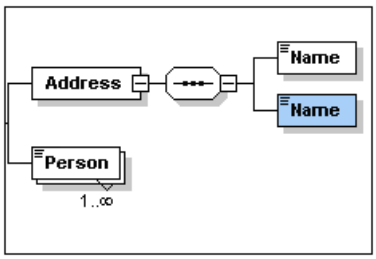
ここまでは、要素やコンポジッターを右クリックし、コンテキストメニューを使って要素を追加してきました。ドラッグアンドドロップを使うとコンテキストメニューよりも素早く要素を追加することができます。このセクションでは、ドラッグアンドドロップを使って Address 要素にさらに子要素を追加して定義を完成させます。

ドラッグアンドドロップを使って Address 要素の定義を完成させるには：

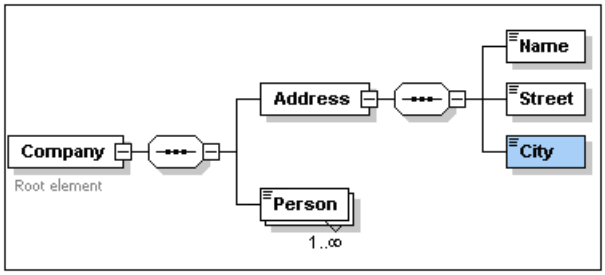
1. Address 要素の Name 要素をクリックして、「Ctrl」キーを押下したまま要素ボックスをマウスでドラッグします。要素ボックスに表示される小さな「+」アイコンはコピーが行われようとしていることを示しています。要素のコピーが接続線と一緒に表示され、要素がどこに作成されるのかを示しています。



2. マウスボタンを離して、Address シーケンスに新しい要素を作成します。新しい要素が間違った場所に作成された場合、Name 要素以下にドラッグしてください。



3. 要素ボックスをダブルクリックして、要素の名前を Street に変更します。
4. 同様に、City という名前でもう1つの要素を作成します。コンテンツモデルは以下のようになります。



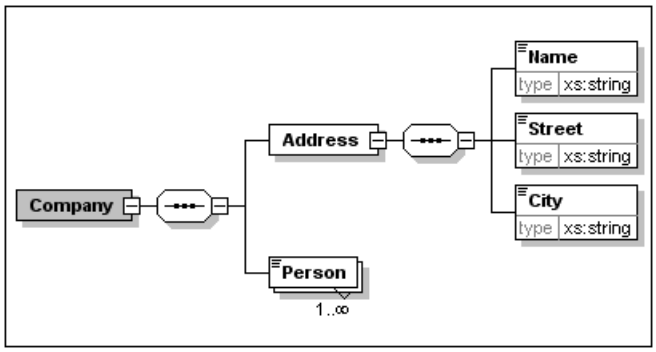
要素は Name、Street、そして City 要素からなるシーケンスを持つことになります。

2.5 コンテンツモデルビューの構成

そろそろコンテンツモデルビューの設定を行っても良いでしょう。コンテンツモデルビューを要素ボックスに、その要素が持つ型が表示されるように設定します。コンテンツモデルビューの設定を行うには：

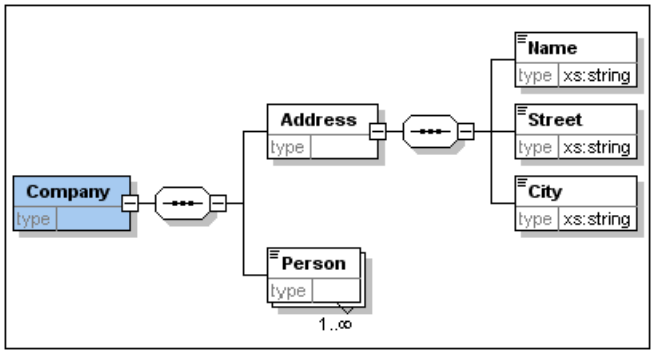
1. コンポーネントのコンテンツモデルビューを選択します（ダイアログ表示アイコン をクリックします）。
2. メニューオプション **「スキーマ設計 | 表示設定」** を選択します。スキーマ表示設定ダイアログが表示されます。

3. （Element タブにある）追加アイコン をクリックして、要素ボックスに対するプロパティの表示レコードを追加します。
4. ドロップダウンメニューから `type` を選択します（またはフィールドをダブルクリックして "`type`" と入力します）。これで各要素のデータ型がコンテンツモデルビューで表示されるようになります。
5. 記述行設定では、**「値の無い行を隠す」** を選択します。これで要素の値が無い場合、データ型の記述子が要素ボックスに表示されなくなります（例えば、要素が複合型の場合などです）。



`xs:string` という記述子が `Name`、`Street`、そして `City` 要素に表示されているのに対して、複合型の要素には無いことに注目してください。これは値の無い行を隠すオプションが有効になっているためです。

6. 記述行設定グループで、常に行を表示ラジオボタンを選択します。
7. 「OK」をクリックして変更を確定します。



データ型の記述子が常に (値を持たない複合型の要素ボックスでも) 表示されていることに注目してください。

メモ:

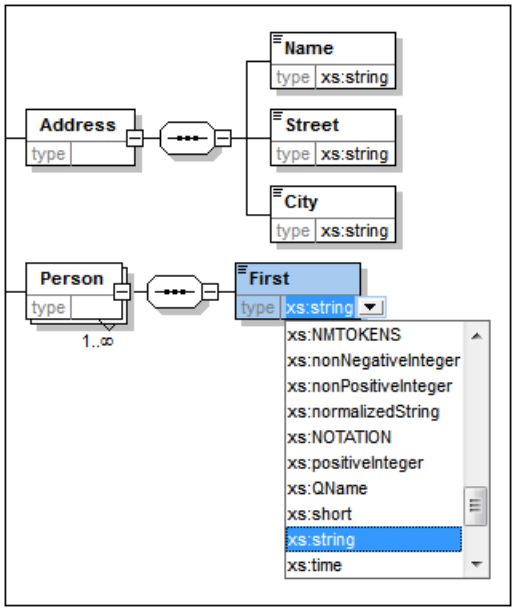
- プロパティの表示レコードは編集することができ、そこで入力された値が要素定義の一部になります。
- スキーマ表示設定ダイアログで行った設定はスキーマドキュメンテーション出力や印刷時の出力にも適用されます。

2.6 基本的なスキーマの完成

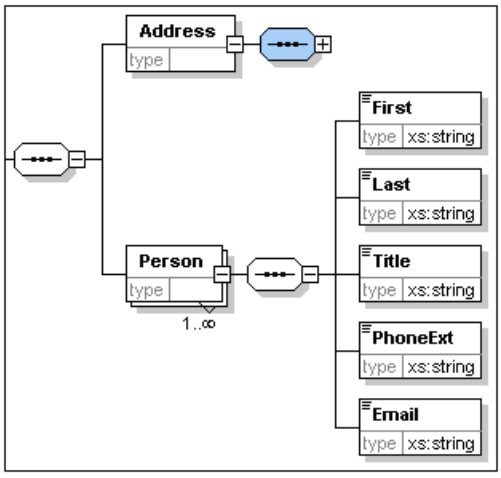
ここまでAddress 要素の定義を行いました。次にPerson 要素の内容を定義する必要があります。Person 要素は以下の子要素（全て単純型）を含む様にします：First、Last、Title、PhoneExtt、そしてEmail。Title を除くこれらの要素は全て必須で、設定された順に現れなければなりません。PhoneExt のデータ型は2桁までのxs:integer となり、それ以外は全てxs:string となります。

Person のコンテンツモデルを作成するには：

1. Person 要素を右クリックし、コンテキストメニューを表示します。**「子を追加 | シーケンス」**を選択します。これによりシーケンスコンポジッターが挿入されます。
2. シーケンスコンポジッターを右クリックし、**「子を追加 | 要素」**を選択します。
3. First を要素名として入力し、**「タブ」**キーを押します。これでカーソルがtype フィールドに移動します。



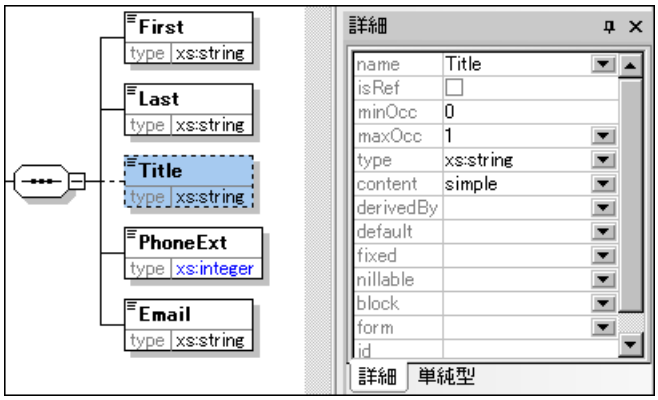
4. xs:string エントリをドロップダウンリストから選択するか、その値をtype フィールドに入力します。
5. ドラッグアンドドロップにより、その他4つの要素を作成します。それぞれLast、Title、PhoneExt そしてEmail と名前を付けます。



メモ: 「Ctrl」キーを押しながら必要な要素をクリックすることで複数の要素を選択することができます。この方法で、例えば複数の要素を一度にコピーすることができます。

要素を省略可能にする

Title 要素を右クリックして、コンテキストメニューから**省略可能**を選択します。要素ボックスの境界線が実線から、要素が省略可能であることを示す点線に変わります。

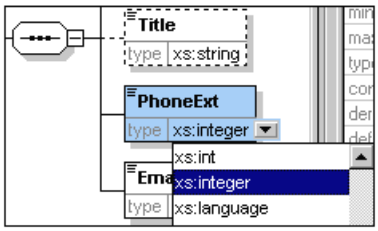


詳細入力ヘルパーにて、minOcc が 0、maxOcc が 1 と表示され、要素が省略可能であることを示します。コンテキストメニューを使う方法以外にも、minOcc に 0 をセットして要素を省略可能にすることができます。

要素の内容を制限する

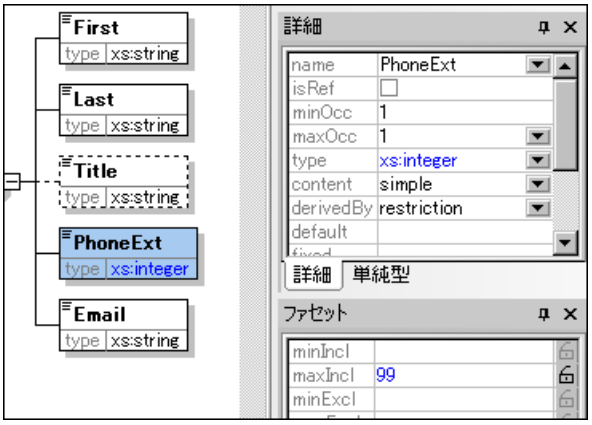
PhoneExt 要素が xs:integer 型で、最高 2 桁の数値だけを持てるようにするには:

1. PhoneExt 要素の Type フィールドをダブルクリックして、ドロップダウンリストから xs:integer を選択 (または入力) します。



この時点でファセット入力ヘルパーのアイテムが変化します。

2. ファセット入力ヘルパーにて、`maxIncl` フィールドをダブルクリックして、99 と入力し、「Enter」で確定します。



この定義により、99までの内線番号（Phone Extension）が妥当となります。

3. メニューオプション **「ファイル | 保存」** を選択し、スキーマに対する変更を保存します。

メモ:

- 単純型のXMLスキーマデータ型（例えば `xs:string` や `xs:date`）を選択することで、詳細入力ヘルパーのコンテンツモデルが自動的に単純型（content = simple）に変更されます。
- 要素にコンポジッター（シーケンス、選択、またはすべて）を追加することで、詳細入力ヘルパーのコンテンツモデルが自動的に単純型（content = complex）
- 上記で説明されたスキーマはXMLSpy アプリケーションフォルダーの `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\Tutorial` フォルダーにある `AddressFirst.xsd` という名前のファイル名で入手可能です。

3 XML スキーマ :高度な機能

今までの作業で基本的なスキーマが作成されました。これからスキーマの開発における高度な機能に進んでいきます。

目的
このセクションでは、以下のことを学習します：

- スキーマの要素としても使用できる複合型ならびに単純型の扱い方
- グローバル要素 を作成し、他の要素からの参照を行う
- 属性とその列挙値を含むプロパティーを作成する

このセクションでは、直前のパートで作成したAddressFirst.xsd スキーマからスタートします。

このセクションで使用されるコマンド
このセクションではスキーマビューだけを使用します。以下のコマンドが使われます：

ダイアグラム表示（または コンテンツモデルビュー表示）。このアイコンはスキーマ概要にある全てのグローバルコンポーネントの左側に配置されています。このアイコンをクリックすることで、対応するグローバルコンポーネントのコンテンツモデルを表示します。

全てのグローバルを表示。このアイコンはコンテンツモデルビューの左上に配置されています。このアイコンをクリックすることで、全てのグローバルコンポーネントを表示するスキーマ概要に移動します。

追加アイコンはスキーマ概要の左上に配置されています。このアイコンをクリックすることでグローバルコンポーネントの追加を行います。

3.1 複合型と単純型を扱う

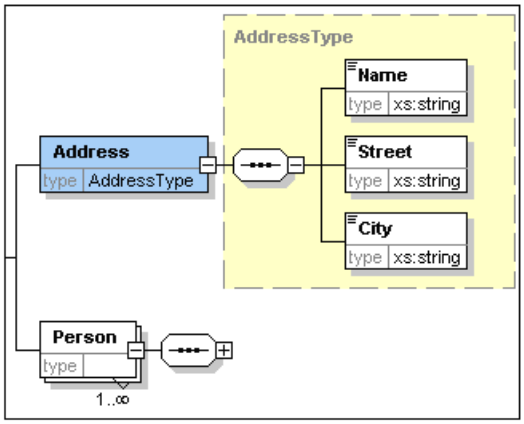
要素のコンテンツモデルの定義を行ったので、それをスキーマのどこか別の場所で再利用することにします。そのためには要素の定義をグローバル複合型、またはグローバル要素として作成する必要があります。このセクションではグローバル複合型を扱います。まずは複合型をグローバルなレベルで作成し、それをコンテンツモデル内で使用できるように拡張します。グローバル要素については後ほど学習します。

グローバル複合型の作成

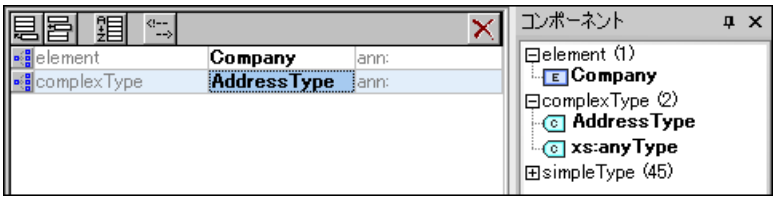
既に定義した（Name、Street、 City 要素を含む）基本的な Address 要素は住所の書式として色々な場所で利用できます。そこで、これら要素の再利用をすることができる複合型を定義することにします。

グローバル複合型を作成するには：

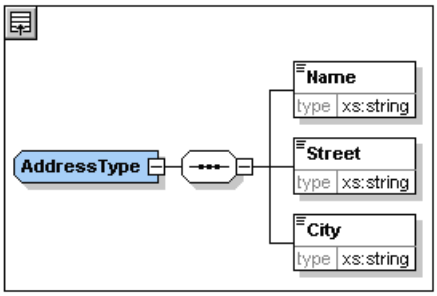
1. コンテンツモデルビューにて Address 要素を右クリックする。
2. 現れたコンテキストメニューで「グローバル化 | 複合型」を選択します。AddressType という名前のグローバル複合型が作成され、Company コンテンツモデルの Address 要素がこの型に割り当てられます。Address 要素の内容は、黄色のボックスで表示されている AddressType のコンテンツモデルになります。Address 要素のデータ型が AddressType になっていることに注目してください。



3. 全てのグローバルを表示アイコン をクリックします。メインウィンドウの表示が、スキーマ内の全てのグローバルコンポーネントを表示するスキーマ概要に移動します。
4. コンポーネント入力ヘルパーにて（＋で示される）**要素**と**複合型**エントリの展開アイコンをクリックして、対応するスキーマが表示されることを確認します。
スキーマ概要は2つのグローバルコンポーネント（Company 要素と複合型の AddressType）を表示しています。コンポーネント入力ヘルパーにも AddressType 複合型は表示されます。



5. AddressType のコンテンツモデルビュー表示アイコン をクリックして、（下のスクリーンショットにあるような）コンテンツモデルを表示します。複合型コンテナーの形に注目してください。



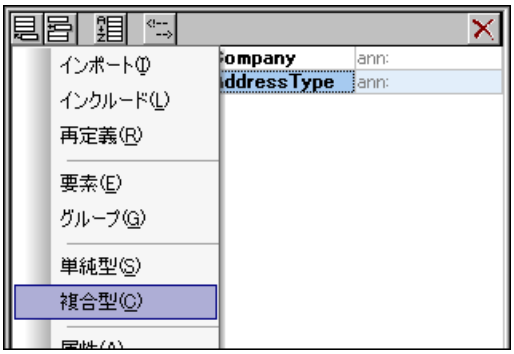
6. 全てのグローバルを表示アイコン をクリックして、スキーマ概要に戻ります。

複合型の拡張

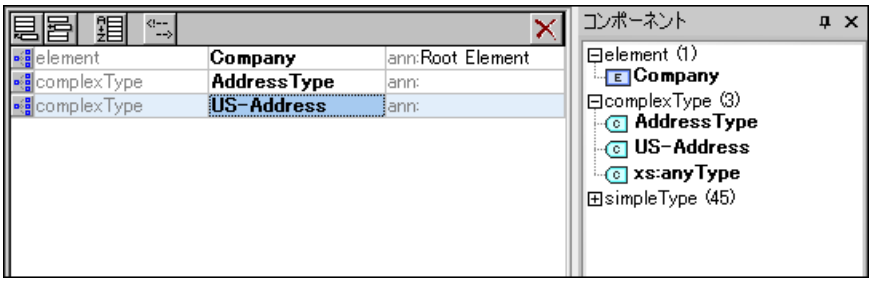
それでは、2つの国に特化した住所を表す為に、グローバル `AddressType` コンポーネントを使用したいと思います。このために、`AddressType` をベースにした新たな複合型を作成し、その定義を拡張します。

以下を行います :

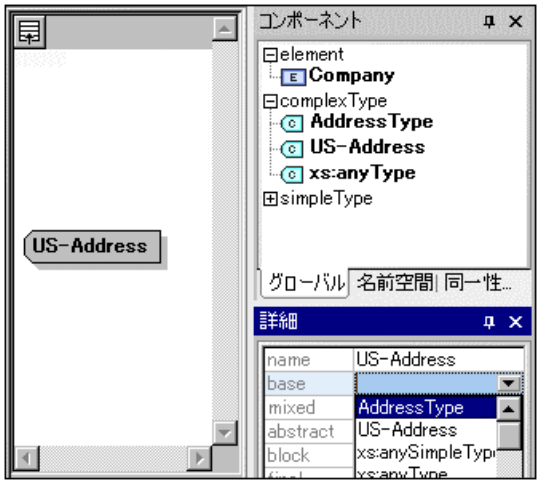
1. スキーマ概要に移動します (コンテンツモデルビューにいる場合、全てのグローバルを表示アイコン をクリックします)。
2. コンポーネントウィンドウの左上にある追加アイコン をクリックします。以下のメニューが開きます:



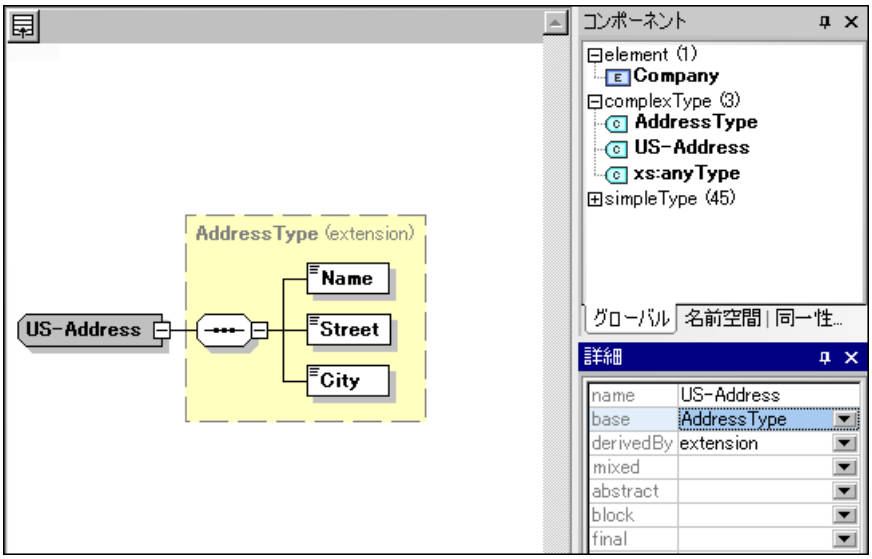
3. メニューから**複合型**を選択します。コンポーネントリストに新しいレコードが現れ、コンポーネントの名前を入力するようにカーソルがセットされます。
4. `US-Address` と入力し、**Enter**」で確定します。(ハイフン -() を入力する代わりに空白 (スペース) を入力した場合、要素名が赤くなり、不正な文字であることが示されます)



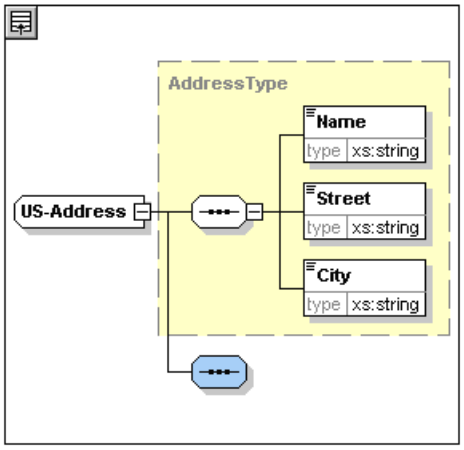
5. `US-Address` のコンテンツモデルビューアイコン をクリックし、新しい複合型のコンテンツモデルを表示します。コンテンツモデルは (以下のスクリーンショットにあるように) 空です。
6. 詳細入力ヘルパーにて、base コンボボックスをクリックし、`AddressType` エントリを選択します。



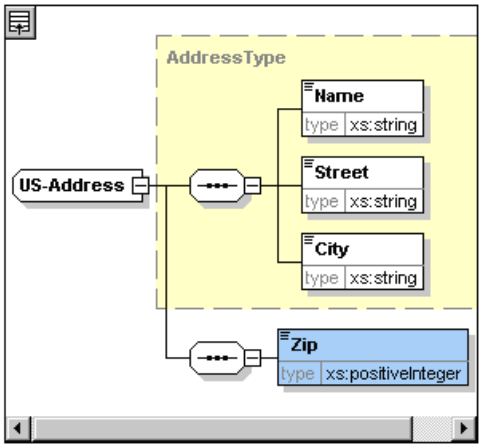
コンテンツモデルビューには `US-Address` のコンテンツモデルとして `AddressType` が表示されます。(*下のスクリーンショット*)。



7. これで `US-Address` 複合型のコンテンツモデルを、郵便番号 (ZIP Code) を持つように拡張することができます。`US-Address` コンポーネントを右クリックして、現れたコンテキストメニューから **子を追加 | シーケンス** を選択します。シーケンスコンポジッターが `AddressType` ボックスの外に表示されます (下のスクリーンショットを参照)。これは要素に対する拡張であることを表わしています。



8. 新しいシーケンスコンポジッターを右クリックして、**子を追加 | 要素**」を選択します。
9. 新たに作成された要素に`Zip` という名前を付け、**「タブ」** キーを押下し、カーソルをtype フィールドに移動させます。
10. ドロップダウンメニューから`xs:positiveInteger` を選択し、**「Enter」** キーにより確定します。



これで、複合型 `AddressType` をもとにした郵便番号（ZIP Code）の拡張を持つ複合型 `US-Address` が作成されました。

グローバル単純型

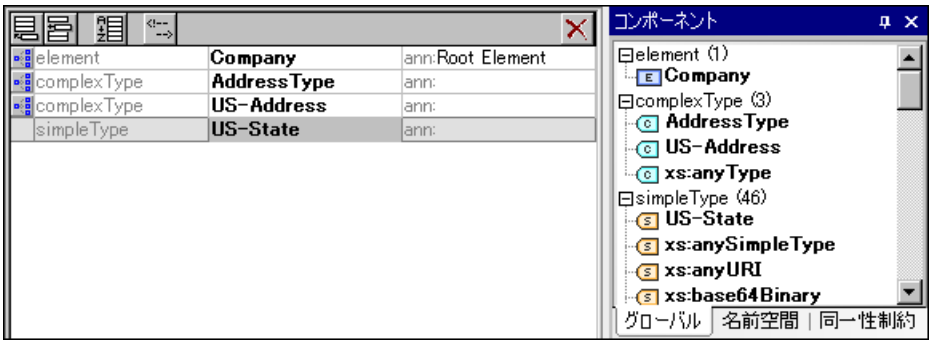
複合型の`US-Address` が複合型の`AddressType` をベースにしたように、単純型をベースにすることもできます。その利点はグローバル複合型の場合と同じで、単純型が再利用されることにあります。単純型を再利用するためには、その単純型をグローバルに定義しなければなりません。このチュートリアルではアメリカの州のコンテンツモデルを単純型として定義することにします。この単純型は他の要素のベースとして使用されることになります。

グローバル単純型の作成

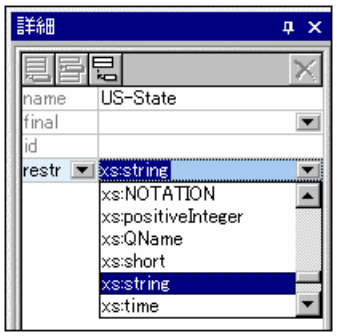
グローバル単純型の作成は、グローバルコンポーネントのリストに新しい単純型を追加し、名前を与え、データ型を定義することから構成されています。

グローバル単純型を作成するには以下をおこないます：

1. スキーマ概要に切り替えます。(コンテンツモデルビュー内で作業している場合は、すべてのグローバルアイコンの表示 をクリックします。)
2. アイコンの追加をクリックします。表示されるコンテキストメニューから**SimpleType** を選択します。
3. 新しく作成されたsimpleType の名前として`US-State` を入力します。
4. 「**Enter**」を押して確認します。単純型 `US-State` が作成され、コンポーネント入力ヘルパー内の単純型のリストに表示されます。 (simpleType エントリのアイコンの展開をクリックして確認してください)。



5. 詳細入力ヘルパー (*下のスクリーンショット*)内では、`restr` の値フィールドにカーソルをポイントして、`xs:string` を入力、または`restr` 値フィールドのドロップダウンメニューから`xs:string` を選択します。



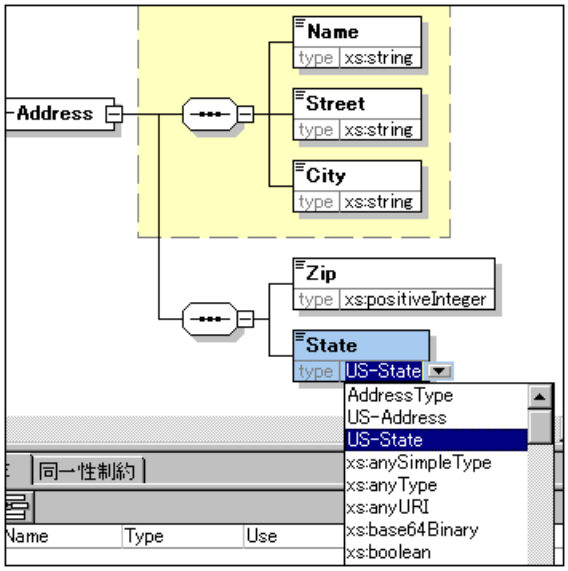
`xs:string` のデータ型である、`US-State` という名前の単純型が作成されます。このグローバル コンポーネントは、`US-Address` のコンテンツモデル内で使用することができます。

コンテンツモデル内でグローバル単純型を使用する

グローバル単純型 は、コンテンツモデル内で、コンポーネントの型を定義するために使用することができます。`State` という名前の要素を定義するために`US-Address` コンテンツモデル内の`US-State` を使用します。

次を行います :

1. スキーマ概要内の`US-Address` のコンポーネント モデルビューアイコン をクリックします。
2. 下のシーケンスコンポジターを右クリックして、**「子の追加 | 要素」**を選択します。
3. 要素名のために、`State` を入力します。
4. **「タブ」**キーを押して、カーソルを型詳細ラインの値フィールドにポイントします。
5. コンボボックスのドロップダウンメニューから、`US-State` を選択します。



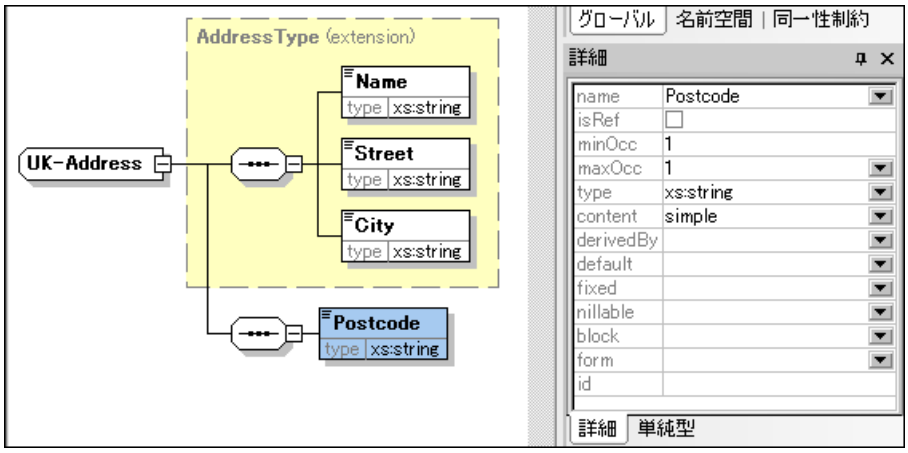
State 要素は、US-State 単純型をベースにします。

第 2の複合型をAddressType をベースにして作成する。
UK の住所が入力できるように、グローバル複合型を作成します。 複合型は、AddressType をベースにして、UK アドレスのフォーマットに一致するように拡張されています。

次を行います :

1. スキーマ概要内にAddressType (base=AddressType) をベースにする UK-Address という名前のグローバル複合型を作成します。
2. UK-Address の コンテンツモデルビュー 内に、Postcode 要素を作成し、xs:string の型を与えます。

K-Address コンテンツモデルは以下のようになります :

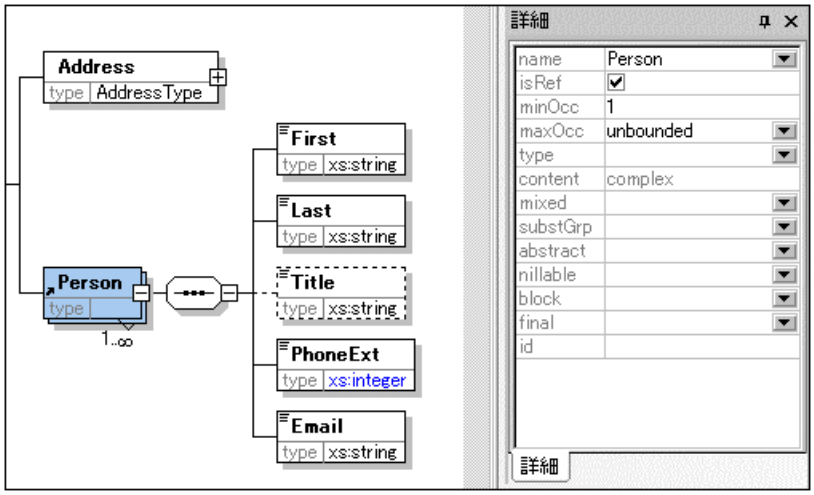


メモ: このセクションでは、コンテンツモデル定義で使用されるグローバル単純型 および複合型を作成しました。グローバル型の利点は、複数の定義で再利用できることです。

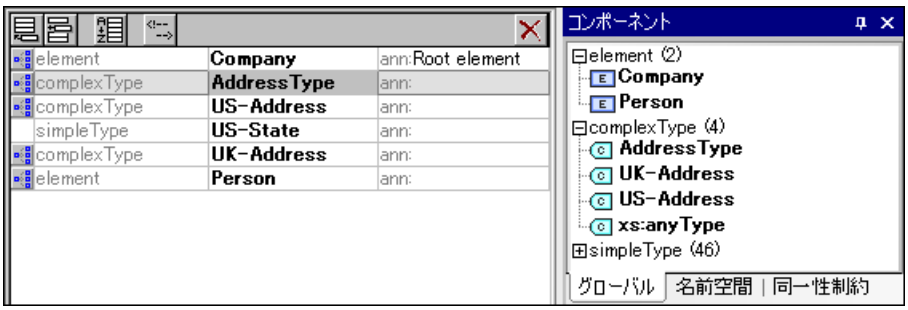
3.2 グローバル要素の参照

このセクションでは、既に定義した Person 要素をグローバル要素に変換し、その要素を Company 要素から参照します。

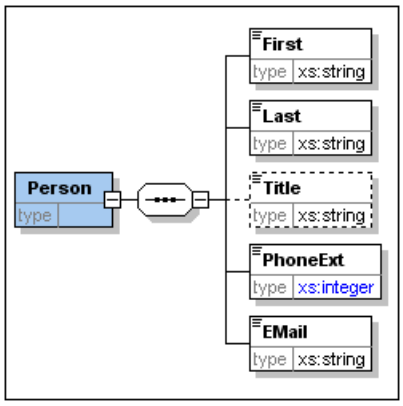
1. 全てのグローバルを表示アイコン をクリックして、スキーマ概要に移動します。
2. Company 要素のダイアグラム表示アイコン をクリックします。
3. Person 要素を右クリックし、**「グローバル化 | 要素」**を選択します。要素に小さな矢印アイコンが表示され、この要素がグローバルに宣言された Person 要素を参照していることを表します。詳細入力ヘルパーには isRef チェックボックスにチェックが入ります。



4. 全てのグローバルを表示アイコン をクリックしてスキーマ概要に戻ります。Person 要素がグローバル要素として表示されます。コンポーネント入力ヘルパーにも Person が表示されます。



5. コンポーネント入力ヘルパーにて、`Person` 要素をダブルクリックし、グローバル要素 `Person` のコンテンツモデルを開きます。



グローバル要素ボックスには矢印が**無い**ことに注目してください。これはその要素が参照されているものであり、参照を行っている要素では無いからです。参照を行っている要素のボックスに矢印は表示されます。

注意：

- グローバル要素を参照する要素は、参照するグローバル要素と同じ名前でなければなりません。
- グローバル定義はXML ドキュメントのどこでコンポーネントが使用されるかの記述は行わず、コンテンツモデルの記述だけを行います。グローバル定義が他のコンポーネントから参照された時に、XML ドキュメント内での場所が指定されます。

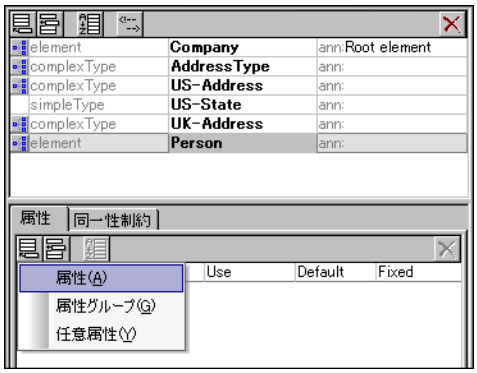
  グローバルに記述された要素は異なる場所で再利用することが可能です。グローバル複合型の場合とは違い、グローバル要素そのものを変更することなくコンテンツモデルの変更を行うことはできません。グローバル要素を参照する要素のコンテンツモデルを変更した場合、グローバル要素のコンテンツモデルも同様に変更され、そのグローバル要素を参照しているその他の要素のコンテンツモデルも変更されます。

3.3 属性と列挙型

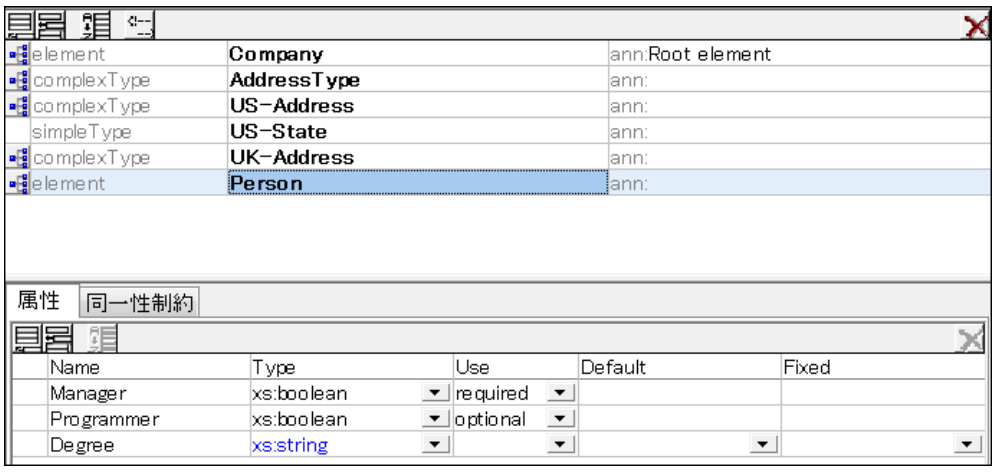
このセクションでは、属性と列挙型の属性の作成方法について学習します

要素属性の定義

1. スキーマ概要にて、Person 要素をクリックしてアクティブにします。
2. 属性 /同一性制約タブグループ (スキーマ概要ウィンドウ下部) の左上にある追加アイコン をクリックし属性エントリを選択します。



3. Name フィールドに Manager という属性の名前を入力します。
4. Type コンボボックスで xs:boolean を選択します。
5. コンボボックスで required を選択します。



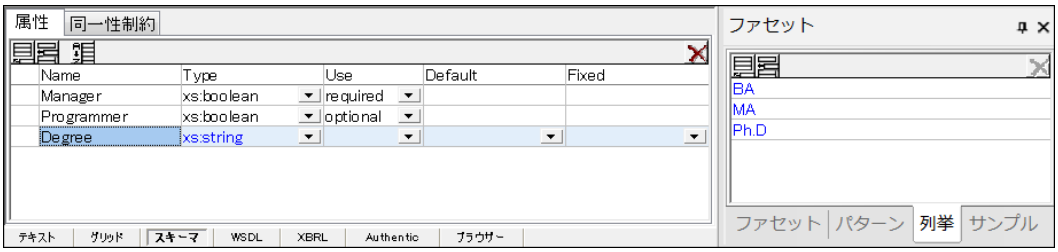
6. 同様の方法で Programmer という名前の、Type=xs:boolean で、Use=optional の属性を作成します。

属性の列挙型の定義

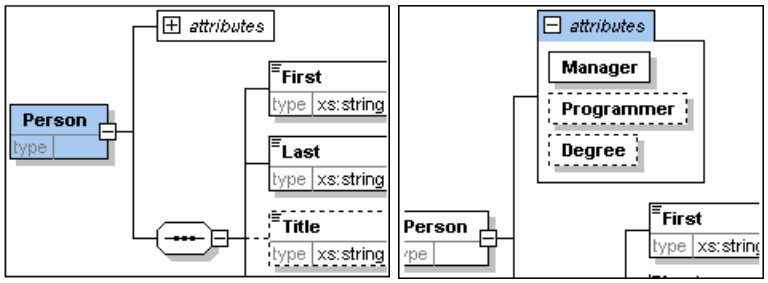
列挙型とはある属性で許される値の集合のことです。XML インスタンスドキュメント内の属性の値がXML スキーマで指定された列挙型の中にない場合、ドキュメントは不正となります。ここでは Person 要素の Degree 属性に対して列挙型を作成します。

次を行います :

1. スキーマ概要にて、Person 要素をクリックしてアクティブにします。
2. 属性ウィンドウの左上にある追加アイコン をクリックして、**「属性」**エントリを選択します。
3. 属性の名前に Degree と入力し、Type に xs:string を選択します。
4. Degree 属性を選択した状態で、入力ヘルパー内の **「列挙」**タブをクリックします（次スクリーンショットを参照）。



5. **「列挙」**タブにて追加アイコン をクリックします。
6. BA と入力して **「Enter」**で確定します。
7. 同様に MA と PhD という列挙を追加します。
8. Person のコンテンツモデルビューアイコン をクリックします。



先程定義した属性がコンテンツモデルビューに表示されます。展開アイコンでその要素で定義された全ての属性が表示されます。この表示モードはメニューオプションの **「スキーマ設計 | 表示設定」**の **「図に表示」**ペインにある **「属性」**チェックボックスのチェックを入れることで有効になります。

9. 全てのグローバルを表示アイコン をクリックして、スキーマ概要に戻ります。

完成したスキーマの保存

注意 : XMLSpy によるスキーマファイルを保存する前に、既にある AddressLast.xsd ファイルの名前を変更し（例えば AddressLast_original.xsd）、上書きしないようにします。

完成したスキーマファイルを好きな名前で保存します（**「ファイル | 名前を付けて保存」**）。チュートリアルの次のパートで使用されるスキーマファイルの名前は AddressLast.xsd であるため、ここでも AddressLast.xsd と名前をつけることをお勧めします。

4 XML スキーマ :XMLSpy の機能

XML スキーマを完成させた後は、ナビゲーションショートカット と XMLSpy から生成することができるスキーマドキュメンテーションの学習をしましょう。

このセクションで使用されるコマンド

このセクションではスキーマビューだけを使用します。以下のコマンドが使われます：

ダイアグラム表示（または コンテンツモデルビュー表示）。このアイコンはスキーマ概要にある全てのグローバルコンポーネントの左側に配置されています。このアイコンをクリックすることで対応するグローバルコンポーネントのコンテンツモデルを表示します。

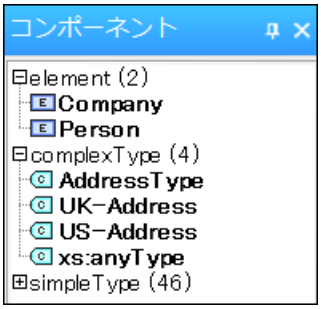
4.1 スキーマナビゲーション

このセクションでは効果的にスキーマビューをナビゲートする方法を学習します。

グローバルコンポーネントのコンテンツモデルを表示する

コンテンツモデルを持つことのできるグローバルコンポーネントは複合型、要素、そして要素グループです。これらのコンポーネントのコンテンツモデルビューは以下のようにして開くことができます。

- スキーマ概要にて、コンポーネント名の左にあるダイアグラム表示アイコン をクリックします。
- スキーマ概要かコンテンツモデルビューにてコンポーネント入力ヘルパーの複合型、要素、または要素グループをダブルクリックします (以下のスクリーンショットを参照)。そのコンポーネントのコンテンツモデルが表示されます。



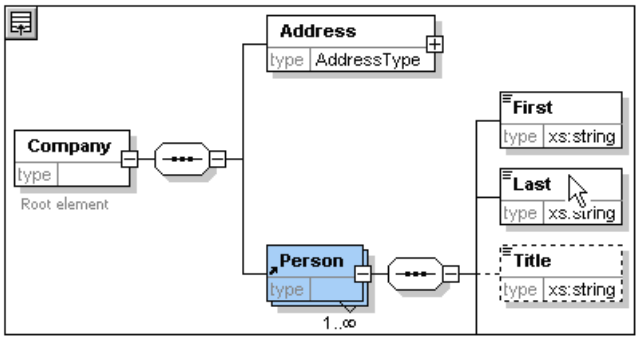
コンポーネント入力ヘルパーにあるその他のグローバルコンポーネント (単純型、属性、属性グループ) をダブルクリックすると、そのコンポーネントはスキーマ概要でハイライトされます (そのようなコンポーネントはコンテンツモデルを持たないため)。

コンポーネント入力ヘルパーにある、グローバルや名前空間タブでもダブルクリックによるメカニズムが使用できます。

参照している要素からグローバル要素に移動する

グローバル要素を参照する要素がコンテンツモデルに含まれている場合、「Ctrl」キーを押しながら必要な要素をダブルクリックすることで、そのグローバル要素またはその要素のコンポーネントに移動することができます。

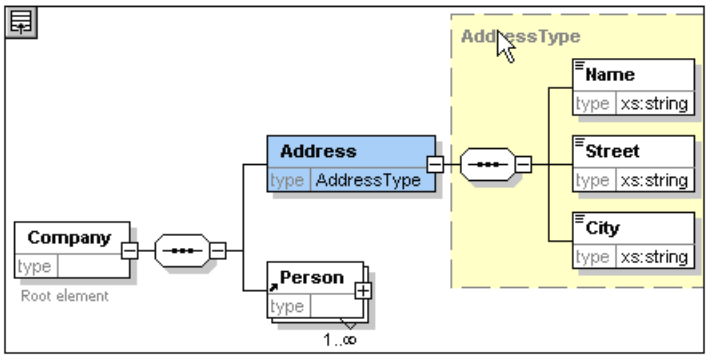
例えば、`Company` コンテンツモデルを参照している際、「Ctrl」キーを押下しながら、`Last` をダブルクリックすることで `Person` コンテンツモデルが開かれ、その中にある`Last` 要素がハイライトされます。



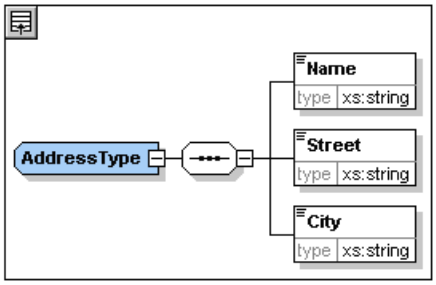
`Last` 要素がハイライトされたとき、関連する情報が入力ヘルパーならびに情報ウィンドウに表示されます。

複合型の定義に移動する

複合型はコンテンツモデル内の要素の型としてよく使用されます。コンテンツモデル内から直接複合型の定義に移動したい場合は、黄色のボックスにある複合型の**名前**をダブルクリックします（以下のスクリーンショットのマウスポインターに注目）。.



この操作で複合型のコンテンツモデルビューに移動することができます。



メモ： 参照されているグローバル要素の場合、参照を行っているコンテンツモデル内の要素を **「Ctrl」** キーを押下しながらダブルクリックすることで、対応する複合型定義内の要素に移動することができます。

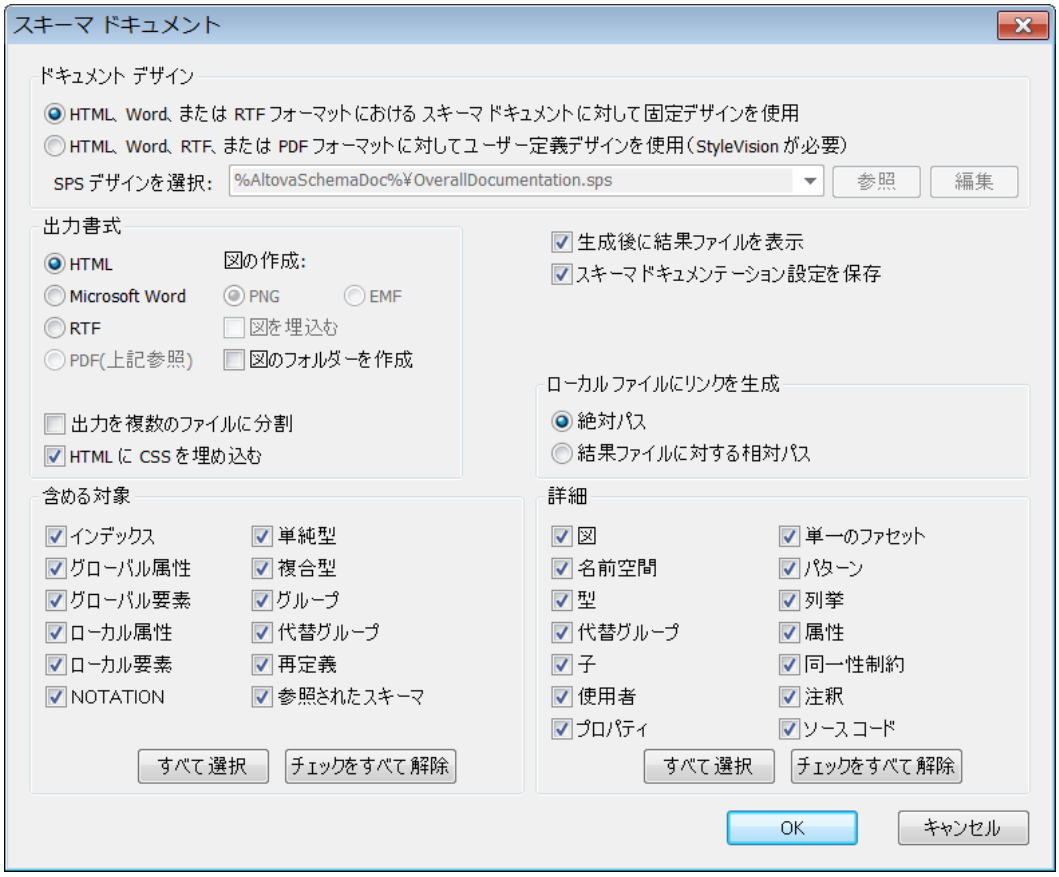
5 スキーマドキュメンテーション

XMLSpy を使って XML スキーマの詳細な文章をHTML や Microsoft Word（MS Word）の形式で生成することができます。ドキュメントに収めたいコンポーネントと希望する詳細度を選択します。関連するコンポーネントはHTML や MS Word ドキュメントなどでハイライトされます。MS Word ドキュメントを生成するためには、お使いのコンピューター またはネットワーク）にMS Word がインストールされている必要があります。

このセクションでは、`AddressLast.xsd` XML スキーマのドキュメンテーションを生成します。

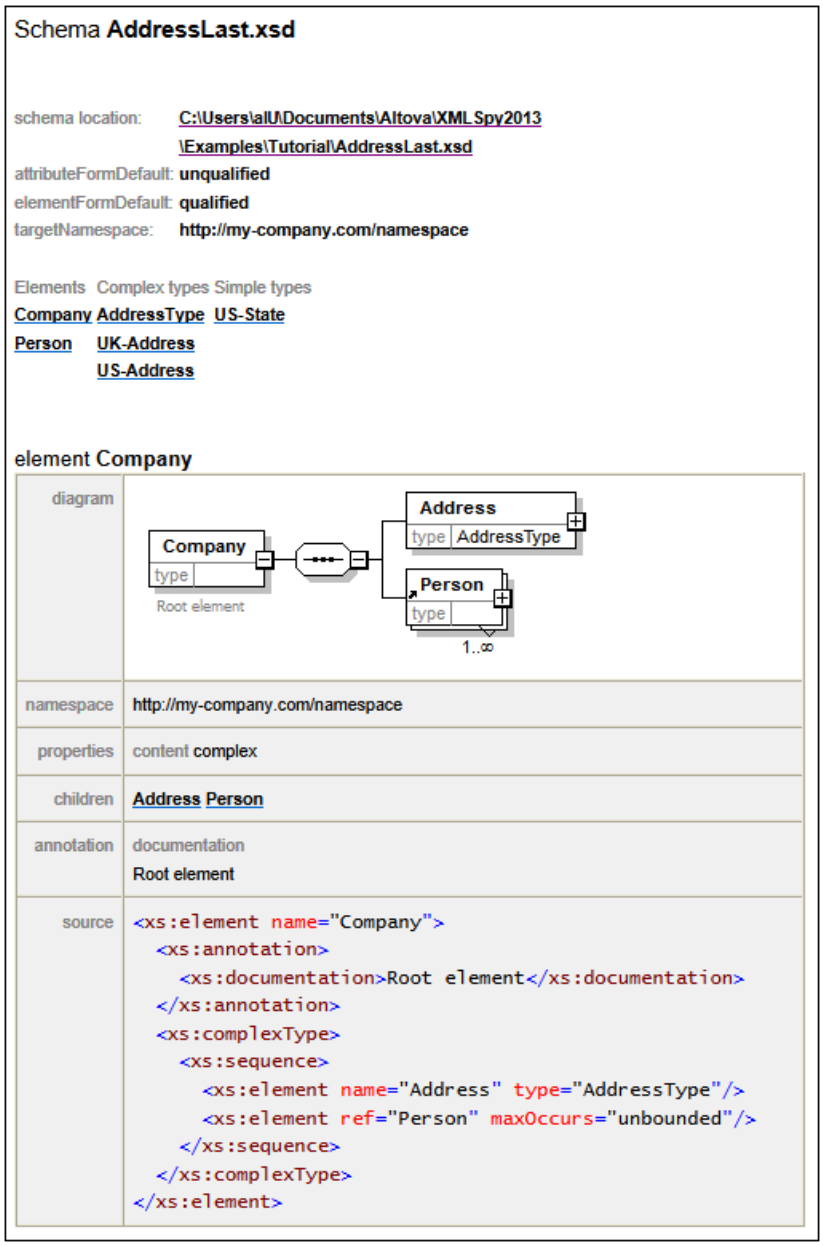
以下を行います：

1. メニューオプションの **「スキーマ設計 | ドキュメンテーションの生成」** を選択し、スキーマドキュメントダイアログを表示します。



2. 出力書式にHTML を選択し、**「OK」** をクリックします。
3. 「名前を付けて保存」ダイアログにて、ファイルが保存される場所を選択し、ファイルの名前を入力（例えば `AddressLast.html`）します。**「保存」** ボタンを押してファイルを保存します。

XMLSpy のブラウザービューにHTML ドキュメントが表示されます。表示されているリンクをクリックして、対応するコンポーネントに移動します。



Schema **AddressLast.xsd**

schema location: C:\Users\alU\Documents\Altova\XMLSpy2013\Examples\Tutorial\AddressLast.xsd
attributeFormDefault: **unqualified**
elementFormDefault: **qualified**
targetNamespace: **http://my-company.com/namespace**

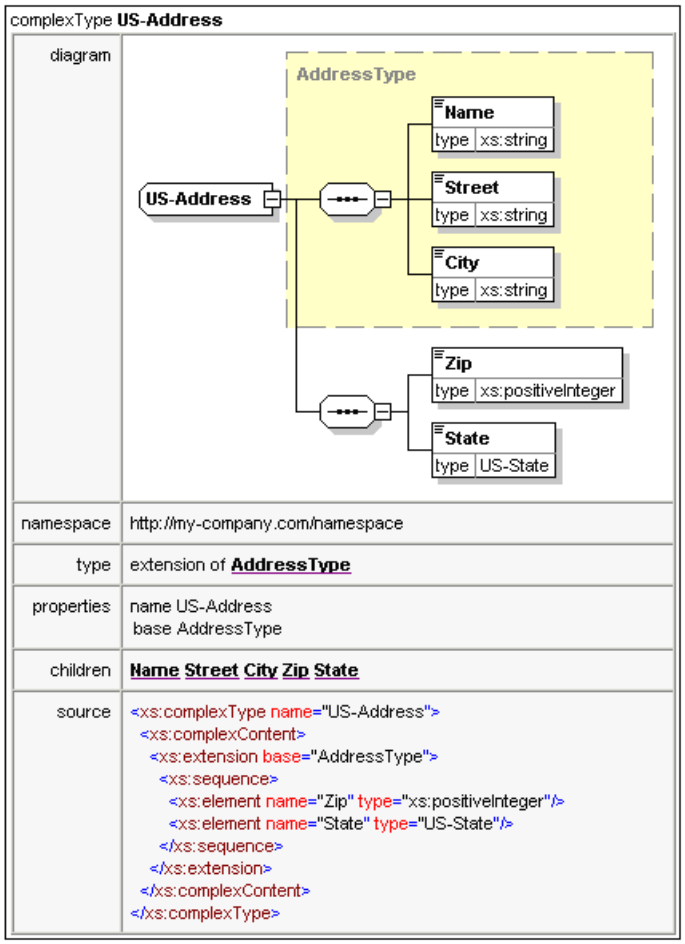
Elements Complex types Simple types
Company AddressType US-State
Person UK-Address
US-Address

element **Company**

diagram	
namespace	http://my-company.com/namespace
properties	content complex
children	Address Person
annotation	documentation Root element

source	`<xs:element name="Company">`
	`  <xs:annotation>`
	`  <xs:documentation>Root element</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:complexType>`
	`  <xs:sequence>`
	`  <xs:element name="Address" type="AddressType"/>`
	`  <xs:element ref="Person" maxOccurs="unbounded"/>`
	`  </xs:sequence>`
	`  </xs:complexType>`
	`</xs:element>`

上の図はHTML フォームの**最初のページ**を示しています。他のスキーマのコンポーネントが含まれている場合、それらのスキーマもドキュメントの中に加えられることになります。



上の図は複合型がどのように表示されるか示しています。

element **US-Address/Zip**

diagram	Zip type | xs:positiveInteger
namespace	http://my-company.com/namespace
type	**xs:positiveInteger**
properties	name Zip isRef 0 content simple
source	<xs:element name="Zip" type="xs:positiveInteger"/>

element **US-Address/State**

diagram	State type | US-State
namespace	http://my-company.com/namespace
type	**US-State**
properties	name State isRef 0 content simple
source	<xs:element name="State" type="US-State"/>

simpleType **US-State**

namespace	http://my-company.com/namespace
type	**xs:string**
properties	name US-State
used by	element **US-Address/State**
source	<xs:simpleType name="US-State"> <xs:restriction base="xs:string"/> </xs:simpleType>

上の図は要素や単純型がどのように表示されるか示しています。

MS Word 出力オプションも試してみてください。Word ドキュメントがMS Word で開かれます。MS Word ドキュメントでハイパーリンク機能を使いたい場合、「Ctrl」キーを押下しながらリンクをクリックしてください。

6 XML ドキュメント

このセクションではXMLSpy を使用してXML ドキュメントを作成し、作業を行う方法を学習します。併せてXMLSpy のインテリジェント編集機能を使用する方法も学習します。

目的

このセクションでは以下のことを学習します：

- `AddressLast.xsd` スキーマをベースにした新しいXML ドキュメントを作成する。
- 要素の型を指定して、その要素に対応するコンテンツモデルを検証に利用できるようにする
- 要素と属性を挿入し、グリッドビューならびにテキストビューにてインテリジェント入力ヘルパーを使ってコンテンツを挿入する
- XMLSpy からMicrosoft Excel にXML データをにコピーする;MS Excel でデータを修正し、そのデータをMS Excel から XMLSpy にコピーしなおす。この機能はグリッドビューにあるデータベーステーブルビューで使用できます。
- データベーステーブルビューの並べ替え機能を使用してXML 要素の並べ替えを行う
- XML ドキュメントを検証する。
- 3桁の内線番号に対応するためにスキーマの修正を行う。

このセクションで使用されるコマンド

このセクションでは主にグリッドビューならびにテキストビュー、そして一部スキーマビューを使用します。以下のコマンドが使われます：

ファイル | 新規作成 新しい種類のXML ファイルを作成します。

表示 | テキストビュー テキストビューに切り替えます。

表示 | グリッドビュー 拡張グリッドビューに切り替えます。

XML | テーブル | テーブルとして表示 1つの要素型の複数の出現をフラットなテーブルで表示します。このビューはデータベーステーブルビュー または単純にテーブルビュー) といいます。アイコンはグリッドビューで一般的なビューとテーブルビューを切り替えるために使用されます。

F7. 整形式のチェックを行います。

F8. 割り当てられたDTD または スキーマに対してXML ドキュメントの検証を行います。

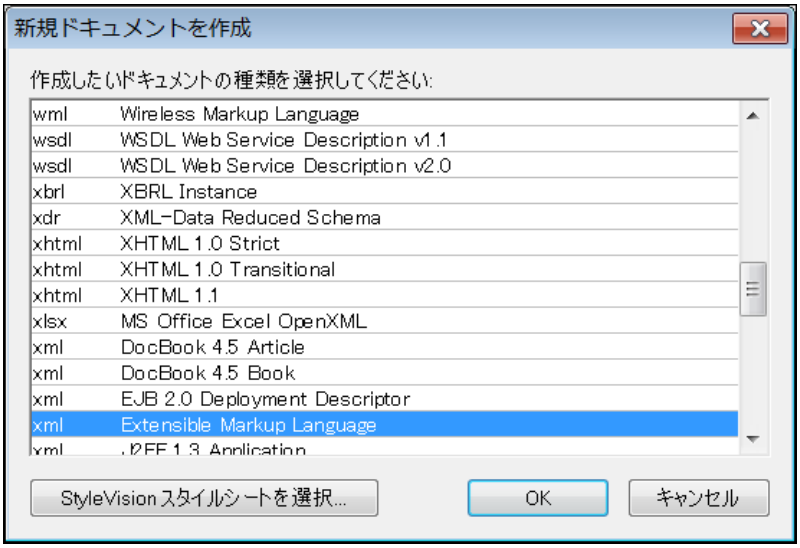
割り当てられたDTD または スキーマファイルを開きます。

6.1 新規 XML ファイルの作成

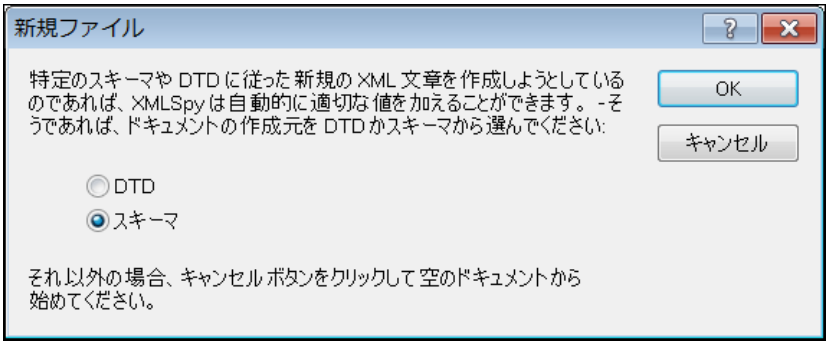
XMLSpyにて新しいXML ファイルを作成する際、スキーマ (DTD またはXML スキーマ) をベースにしたものにするかどうか決定することができます。このセクションでは、既に作成したAddressLast.xsd スキーマをベースにした新しいファイルを作成します。

新しいXML ファイルを作成するには：

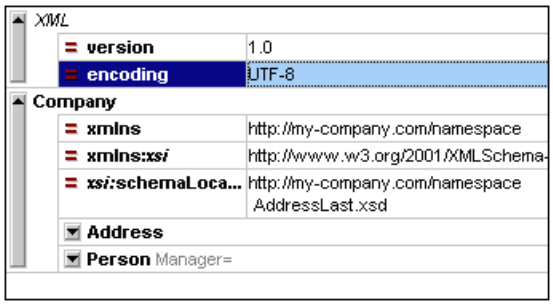
1. メニューオプションから **「ファイル | 新規作成」** を選択します。新規ドキュメントを作成ダイアログが開かれます。



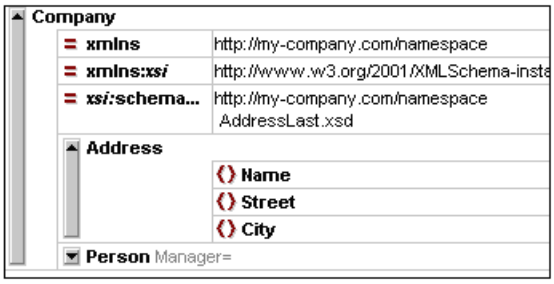
2. ダイアログからExtensible Markup 言語 Extensible Markup 言語 エントリを選択し、 **「OK」** で確定します。ポップアップが表示され、XML ドキュメントのベースにDTD またはスキーマを使用するか尋ねられます。



3. **スキーマ** ラジオボタンをクリックし、 **「OK」** で確定します。次のダイアログが表示され、XML ドキュメントのベースとなるスキーマファイルを選ぶように促されます。
4. 参照もしくはウィンドウボタンを使用して、スキーマファイルを見つけます。ウィンドウボタンを使うことでXMLSpy にある全てのファイルとプロジェクトを閲覧することができます。AddressLast.xsd を選択し (チュートリアルのイントロダクションを参照ください)、 **「OK」** により確定します。スキーマで定義された要素を含むXML ドキュメントがメインウィンドウで開かれます。
5. **「グリッド」** タブをクリックし、グリッドビューを選択します。
6. グリッドビューに表示されるドキュメントの構造に注目してください。要素をクリックして選択を狭めてみてください。ドキュメントは以下のように見えているはずです。

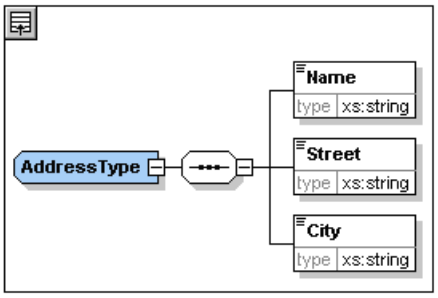


7. Address の横にある アイコンをクリックして、Address の子要素を表示します。ドキュメントは以下のようになるはずです。

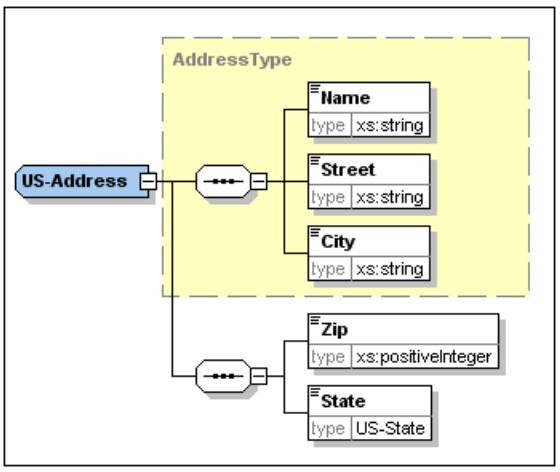


6.2 要素の型の指定

Address の子要素はグローバル複合型 AddressType として定義されています (XML スキーマ AddressLast.xsd 内で定義されているスキーマビューのコンテンツモデルが以下のスクリーンショットでは示されます)。



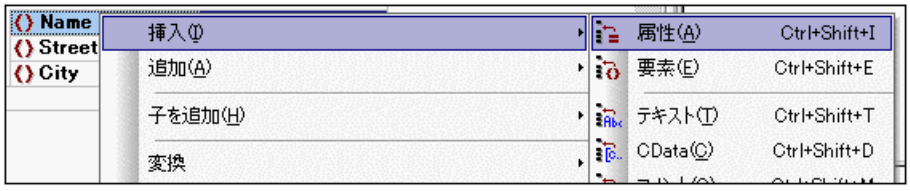
しかしここでは一般的な住所の型よりも、US または UK の住所を使用します。AddressLast.xsd スキーマで US-Address と UK-Address というグローバル複合型を作成したのを覚えているでしょうか? US-Address のコンテンツモデルを以下に示します。



XML ドキュメントにて Address 要素が (一般的な AddressType 型ではなく) 拡張された Address 型 (US-Address か UK-Address) の要件を満たすためには、必要な複合型を Address 要素の属性として指定しなければなりません。

次を行います :

1. XML ドキュメントにて Address 要素を右クリックして、**追加 | 属性**」をメニューから選択します。



   Address 要素に属性フィールドが追加されます。
2. xsi:type が属性の名前として入力されていることを確認します (以下のスクリーンショットを参照)

3. **「タブ」** キーを押して隣の (値) フィールドに移動します。

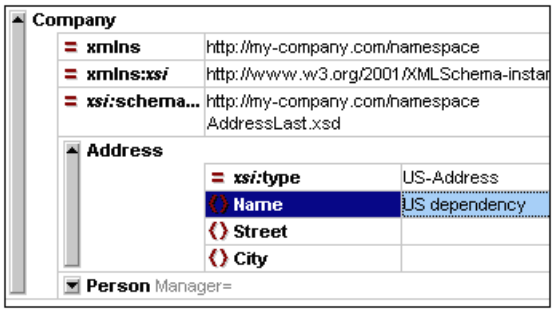
4. 属性の値に`US-Address` と入力します。

メモ: xsi *プレフィックス*を使うことでXML ドキュメント*インスタンス*に関連した特別なXML *スキーマ*を使えるようになります。*スキーマ*をXML ファイルに関連付けたときにxsi *プレフィックス*の名前空間が、自動的にドキュメントに追加されたことに注目してください。上の場合はAddress 要素の型を指定しました。さらに詳しい情報については[XML スキーマの仕様](#) (英文) を参照ください。

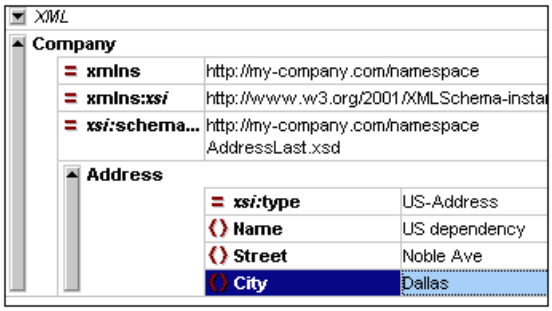
6.3 グリッドビューを使ったデータの入力

XML ドキュメントにデータを入力することができます。データの入力を行うには：

1. Name の値フィールドをダブルクリックし（または矢印キーを使い）、US dependency と入力し、「Enter」により確定します。

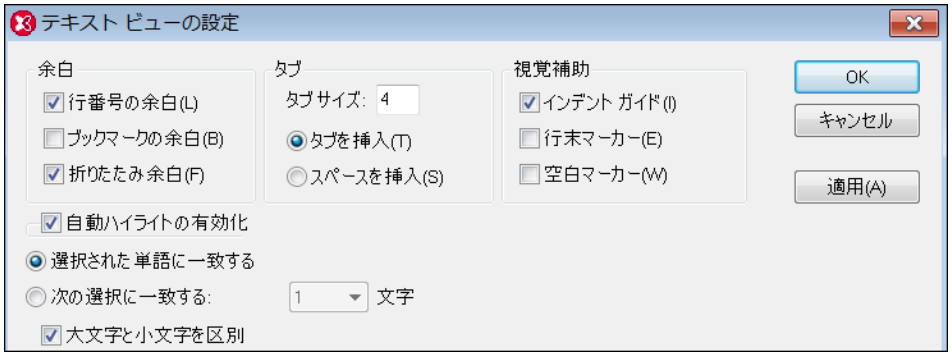


2. 同様の方法で、Street と City にも値を入力します（例えば Noble Ave と Dallas）。
3. Person 要素をクリックし、「Delete」キーを押下して Person 要素を削除します（次のセクションで追加し直します）。この作業が終わると、Address 要素全てがハイライトされます。
4. Address 要素の子要素をクリックして、それ以外の Address の子要素が選択されていない状態にします。XML ドキュメントは以下のようになっているはずです：

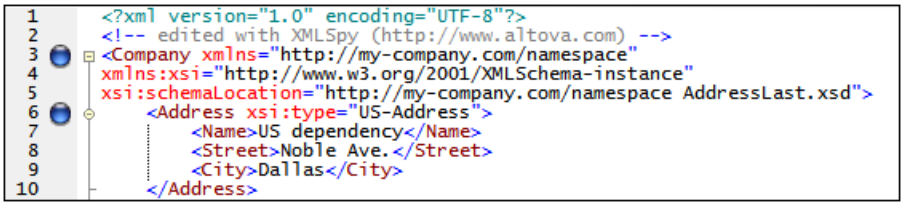


6.4 テキストビューを使ったデータの入力

上記に加えて、テキストビューにはテキストの編集作業を楽にする多くの構造的な編集機能があります。これらの機能はテキストビュー設定ダイアログにて有効/無効の変更を行うことができます（**表示 | テキストビュー設定**」、下のスクリーンショットを参照）。



下のスクリーンショットは、上のダイアログの設定に合わせたテキストビュー内の現在のファイルを表示しています。



左側に 3 つのマージンがあります：(i) 行番号のマージン、(ii) ブックマークマージン（2 つの青いブックマークを含みます）、(iii) (XML 要素の表示を展開/折りたたみすることのできる) ソース折りたたみマージン。

更に、インデントガイド、行の終わりマーカー、空白マーカーなどの視覚補助は、チェックおよびチェックの解除によりオン/オフすることができます。テキストビュー設定ダイアログの視覚補助ペインでオン/オフすることができます。(上のスクリーンショットを参照)。上のスクリーンショットは、インデントガイドが切り替えられており、1 つのインデントガイドを `Address` 要素で表示しています。

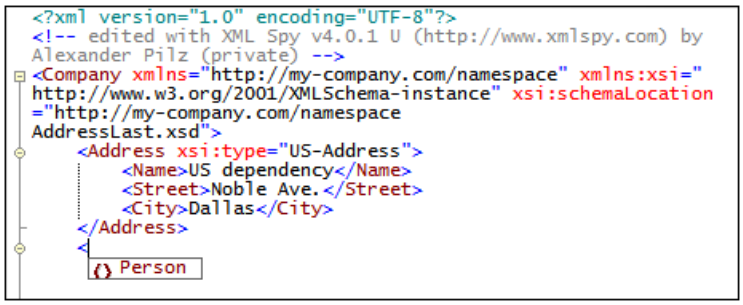
メモ： テキストビューに関連した印刷およびブックマークに機能はこのチュートリアルの テキストビューの設定 のセクションで説明されています。

テキストビューでの編集

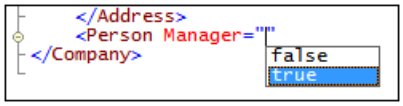
テキストビューに表示される以下の余白（マージン）の有効/無効を設定することができます。

以下を行います：

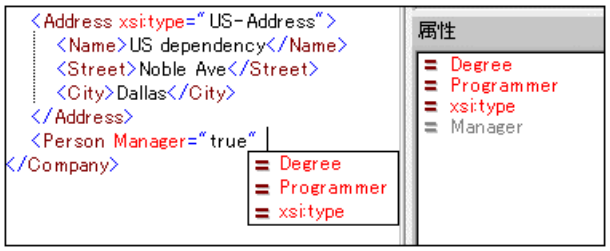
1. メニューオプションの **表示 | テキストビュー**」を選択するか、**テキスト**タブをクリックします。これで XML ドキュメントがテキストの形で構文カラー強調表示と共に表示されます。
2. テキストカーソルを `Address` 要素の終了タグの直後に移動させ、「**Enter**」を押して新しい行を追加します。
3. この位置で小なりかっこ < を入力します。（スキーマの定義により）この位置で許されている全ての要素がポップアップで表示されます。この位置では `Person` 要素だけが許されているので、それだけが表示されます。



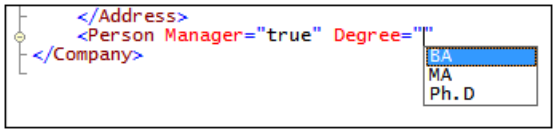
4. Person エントリを選択します。Person 要素がManager 属性と共に挿入され、Manager 属性の値フィールドの内部にカーソルが移動します。
5. Manager 属性のポップアップからtrue を選択します。



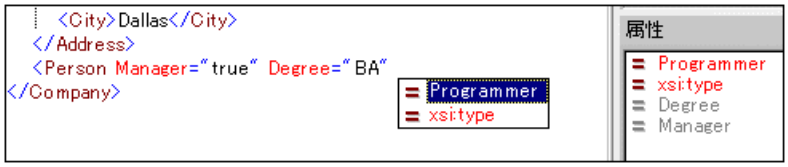
6. カーソルを行末に移動し（例えば「End」キーを使うことができます）、スペースバーを押します。新しいポップアップが開かれ、この位置で許されている属性のリストが表示されます。同様に、属性入力ヘルパーにて使用することのできる属性が赤色で表示されます。Manager 属性はグレーになっていますが、それはその属性が既に使われたためです。



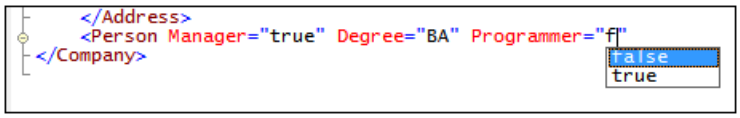
7. 矢印キーを使ってDegree を選択し、「Enter」を押します。これでまた別のポップアップが表示され、既に定義された列挙（BA、MA、またはPhD）から選択を行うことができます。（列挙はXML スキーマにより許可されている値です。）



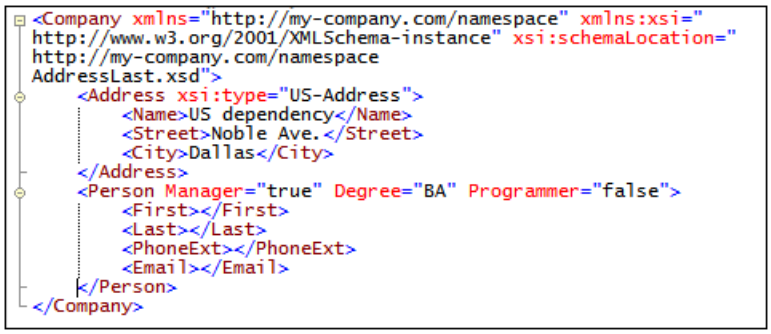
8. 矢印キーを使ってBA を選択し、「Enter」で確定します。さらにカーソルを（「End」キーを使うなどして）行末に移動し、スペースバーを押します。今度はManager とDegree が属性入力ヘルパーでグレーになります。.



9. 矢印キーでProgrammer を選択し、「Enter」を押します。



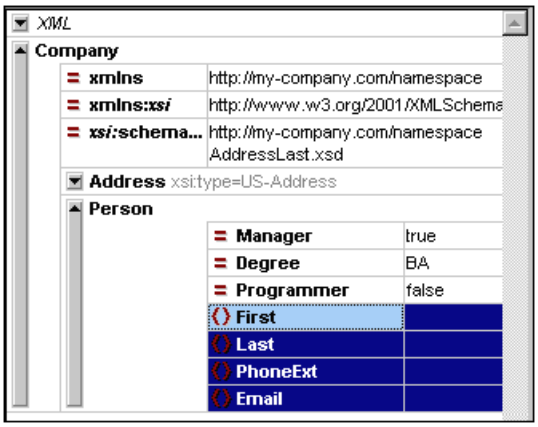
10. "f" キーを押して **「Enter」**を押します。
11. カーソルを行末まで移動し、大なり記号 **>** を入力します。XMLSpy が Person に必要な子要素を自動的に挿入します（省略可能な Title 要素が挿入されていないことに注目してください）。全ての要素には開始と終了タグがありますが、内容は記述されていません。



これでテキストビューにて Person にデータを入力することができますが、グリッドビューに移動して、ドキュメントの編集時にどれだけ簡単にビューを変更できるか見てみましょう。

グリッドビューへ変更

グリッドビューへ変更するには、メニューオプションの **「表示 | 拡張グリッドビュー」**を選択するか、**「グリッド」**タブをクリックします。新たに加えられた Person の子要素がハイライトされます。



それでは、ドキュメントの検証を行い、見つかったエラーを修正していきましょう。

6.5 ドキュメントの検証

XMLSpy ではXML ドキュメントを 2 つの方法で評価できます：

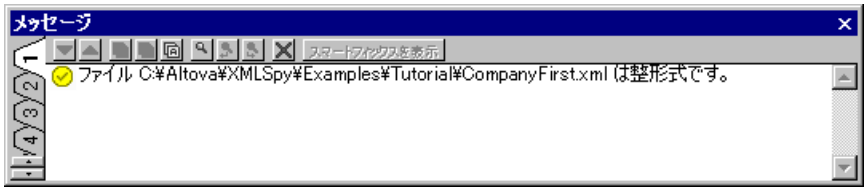
- 整形式のチェック
- 検証

上のうちのどちらかが失敗した場合、ドキュメントを正しく修正しなければなりません。

整形式のチェック
開始タグと終了タグがマッチし、要素が正しくネストされ、間違った文字があったり必要な文字（例えばセミコロンの無いエンティティなど）がなかったりしなければ、そのXML ドキュメントは整形式であると言います。

どのビューでも整形式のチェックを行うことができます。テキストビューでやってみましょう。整形式のチェックを行うには、メニューオプションの **「XML | 整形式のチェック」** を選択するか、**「F7」** キーを押すか、 をクリックします。メインウィンドウの下部にあるメッセージウィンドウにドキュメントは整形式であるというメッセージが表示されます。

メッセージウィンドウの出力には9つのタブがあることに注目してください。評価された結果の出力はアクティブなタブに対して行われます。従って、タブ1で整形式のチェックをしておきながら、その結果を残したまま 検証を行う前に、タブ2に移ることができます（タブ1に残ったままだと検証の結果で上書きされてしまいます）。

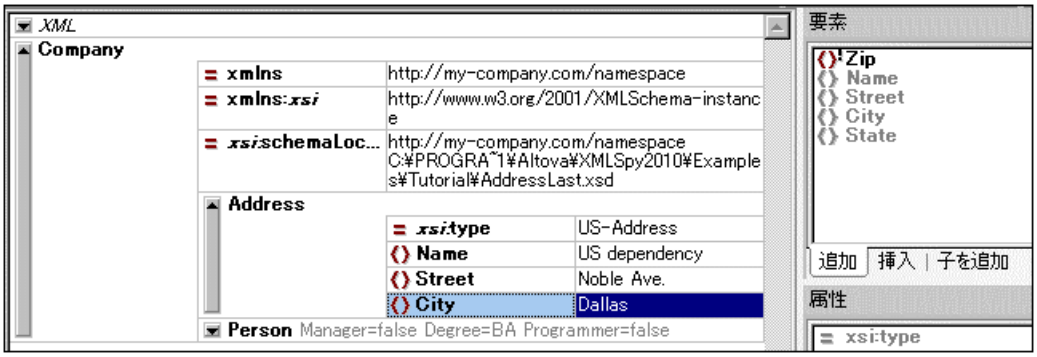


注意：このチェックではスキーマとの整合性を考慮したXML の構造はチェックされません。スキーマとの整合性の評価は検証にて行われます。

検証
XML構造が指定されたスキーマの記述に従ったものであれば、XML ドキュメントは妥当となります。

XML ドキュメントの妥当性をチェックするには、まずグリッドビューを選択し、メニューオプションから **「XML | 検証」** を選択するか、**「F8」** キーを押すか、または をクリックします。ファイルが妥当でないというエラーメッセージがメッセージウィンドウに表示されるはずです。これはAddress 内のCity 要素の後に必須要素が必要だからです。スキーマをチェックしてみると、（xsi:type 属性を使ってAddress 要素から指定した）US- Address 複合型の中にはCity 要素の後にはZip 要素とState 要素が続かなければならないと記述されているのが分かるでしょう。

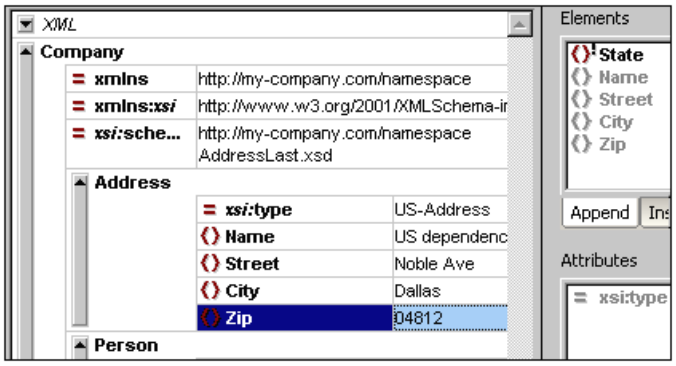
不正なドキュメントの修正
ドキュメントが不正になる原因箇所（この場合は`City` 要素）がハイライトされます。



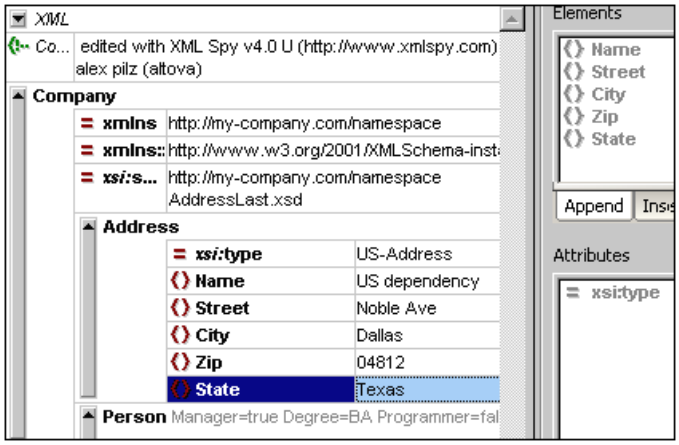
それでは画面右上にある要素入力ヘルパーを見てください。Zip 要素が感嘆符 (!) と共に表示されています。これはその要素が現在のコンテキストで必須であることを示しています。

検証エラーを修正するには：

1. 要素入力ヘルパーにて、Zip 要素をダブルクリックします。これによりZip 要素がCity 要素の後に挿入されます (ダブルクリックするのは **追加** タブの要素であることに注意してください)。
2. **タブ** キーを押し、州の郵便番号 (04812) を入力し、**Enter** で確定します。これで要素入力ヘルパーが State 要素が必須であることを示すようになります (感嘆符と一緒に表示されます)。*以下のスクリーンショットを参照してください。*



3. 要素入力ヘルパーにて、State 要素をダブルクリックします。その後 **タブ** キーを押し州の名前 (Texas) を入力し、**Enter** で確定します。これで要素入力ヘルパーにはグレーの要素しか表示されなくなります。これは Address にはもう必須の子要素が無いことを表しています。

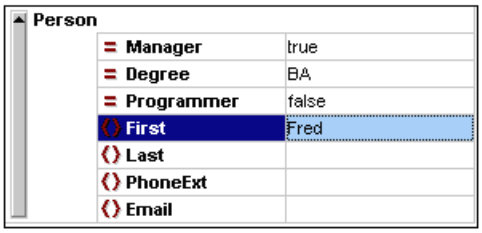


ドキュメントを完成させ、再度検証する

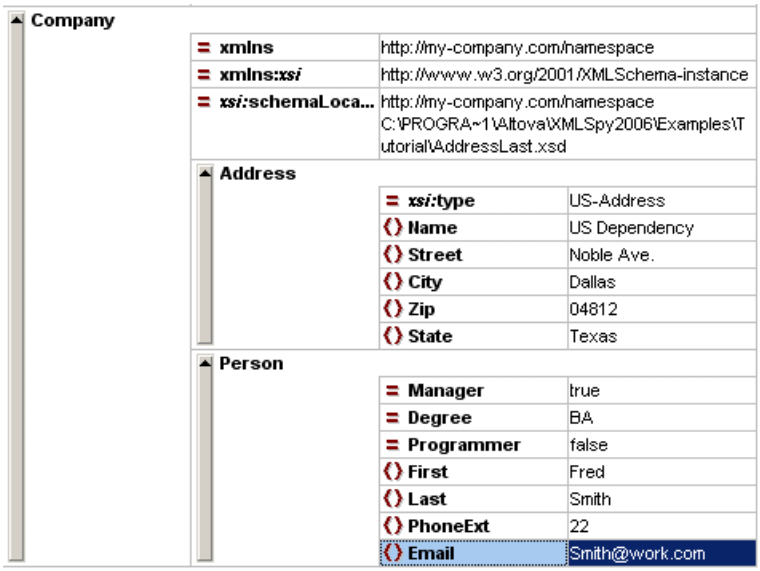
それでは再度検証を行う前にPerson 要素の情報を入力し、ドキュメントを完成させましょう。

以下を行います：

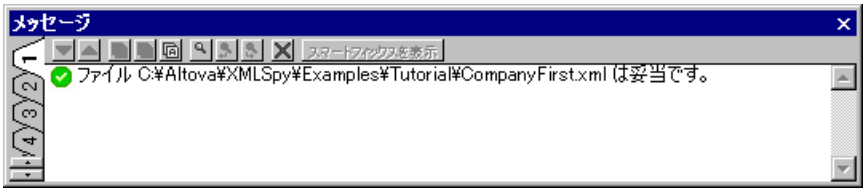
1. First 要素をクリックして、名前（例えばFred）を入力し、「Enter」で確定します。



2. 同様の方法で、Person の子要素全て（Last、PhoneExt、そしてEmail）のデータを入力してください。PhoneExt の値は（Xキーマで指定既に指定したように）99以下の整数でなければならないことに注意してください。XML ドキュメントはグリッドビューで以下のようになっているはずです。



3. を再度クリックして、ドキュメントが妥当かどうかのチェックを行ってください。メッセージウィンドウにファイルが妥当であると言うメッセージが表示されます。



4. メニューオプションから「ファイル | 上書き保存」を選択し、XML ドキュメントに適切な名前（例えばCompanyFirst.xml）を与えます。Tutorial フォルダー内には既にCompanyFirst.xml ファイルがあることに注意してください。作成したファイルを保存する前に、そちらのファイルの名前を変更する必要があるかも知れません。

メモ：保存する際にXML ドキュメントが妥当である必要はありません。不正なドキュメントを保存しようとすると、その旨の警

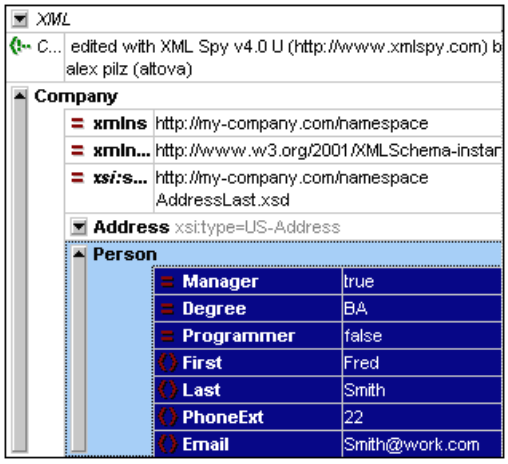
告が表示されますが、不正のままの状態で保存したいのであれば、**保存することは可能です**。

6.6 要素と属性の追加

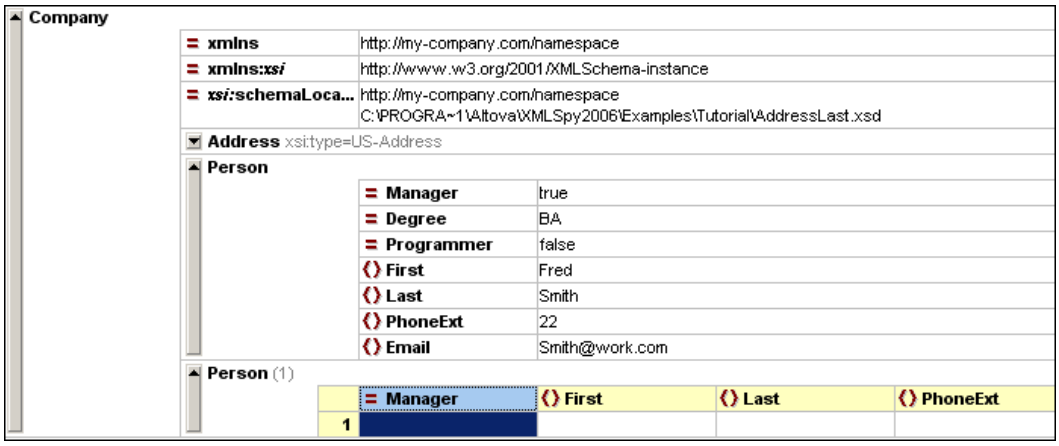
Person この段階ではドキュメント内にPerson 要素は1つしかありません。

新たにPerson 要素を追加するには：

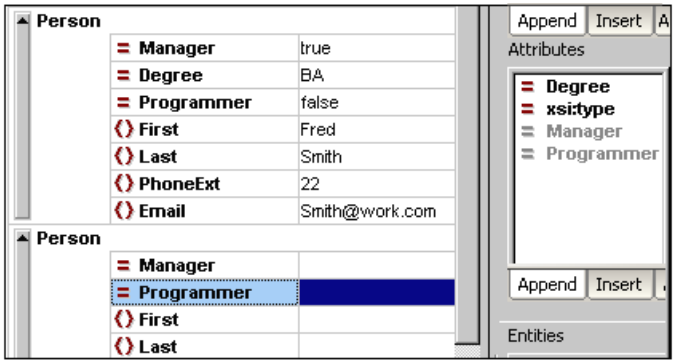
1. Address 要素の左にあるグレーのサイドバーをクリックしてAddress 要素を折りたたみます。これによりビュー内で使える空間を増やせます。
2. グリッドビューのPerson 要素をクリックしてPerson 要素全体を選択します。要素入力ヘルパーの **「追加」** タブにてPerson 要素が使えるようになっていることに注目してください。



3. 要素入力ヘルパーのPerson 要素をダブルクリックします。新しいPerson 要素が必須子要素と一緒に追加されます（下のスクリーンショットを参照）。Person 要素で省略可能なTitle 子要素が追加されていないことに注目してください。



4. Person 開始タグの大なりかっこ> の直前にカーソルを移動させ、属性入力ヘルパーの **「追加」** タブにて、Programmer エントリをダブルクリックします。これによりManager 属性の後にProgrammer 属性が追加され、属性入力ヘルパーではProgrammer がグレーになります。
5. 新しいPerson 要素のManager 属性をクリックします。属性入力ヘルパーを見てください。既に入力されたManager エントリがグレーになっています。Manager 属性に関する情報が表示されている情報ウインドウにも注目してください。Manager 属性は必須の属性であるため自動的に追加されています。Programmer 属性は追加されていません。
6. 属性入力ヘルパーの **「追加」** タブにて、Programmer エントリをダブルクリックします。これで空のProgrammer 属性がManager 属性の後に挿入されます。



これで属性入力ヘルパーの`Programmer`属性がグレーになります。

このビューでも要素の情報を入力することは可能ですが、`Person`の様な複数回出現する構造の入力に適したグリッドビューのデータベースデーブルビューに切り替えましょう。

6.7 データベース /テーブルビューでの編集

グリッドビューの中には複数回出現する要素の編集に適したデータベース/テーブルビューという特別なビューが存在します（これ以降はテーブルビューと表記します）。個々の要素の型がテーブルとして表示されます。要素の型がテーブルとして表示されたとき、その子供（要素ならびに属性）は列として表示され、複数回出現する情報が行（レコード）として表示されます。

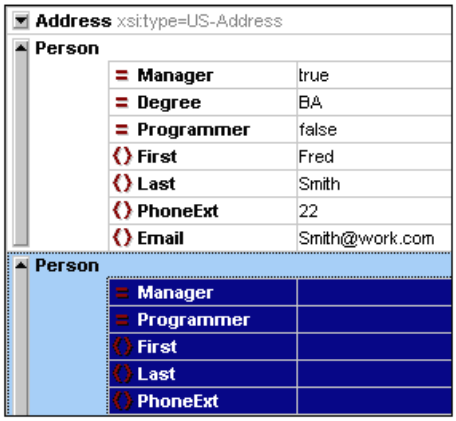
要素の型をテーブルとして表示するには、（複数回出現する）任意の要素を選択し、ツールバーのテーブルとして表示アイコン をクリックします（メニューオプションの **「XML | テーブル| テーブルとして表示」**を選択しても同様の操作が可能です）。この操作で要素の型がテーブルとして表示されます。複数回出現する子要素もテーブルとして表示されます。テーブルビューは拡張グリッドビューで提供され、XML 形式のファイル全て（XML、XSD、XSL など）の編集で使用できます。

テーブルビューの利点
テーブルビューには以下のような利点があります：

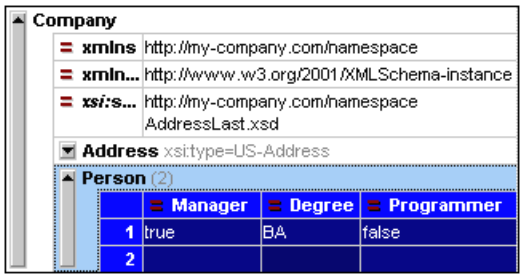
- ドラッグアンドドロップで列のヘッダーを変更することができます。つまり、テーブルに対応する全ての出現に対して、XML ドキュメント内での子要素または子属性の位置を修正することができるということです。
- テーブルは **「XML | テーブル| 昇順に並べ替え」**または **「XML | テーブル| 降順に並べ替え」**を使うことによりソートを行うことができます。
- 新しい行（例えば新しい要素）を **「XML | テーブル| 行を挿入」**により追加することができます。
- サードパーティ製品から、またはサードパーティ製品へコピーと貼付けを行うことができます。
- テーブルでもインテリジェント編集機能が使用できます。

テーブルに要素の型を表示する
Person 要素をテーブルとして表示するには：

1. グリッドビューにて、`Person` テキストの1つの近くをクリックして `Person` 要素を選択します。



2. メニューオプションから「XML | テーブル| テーブルとして表示」を選択するか、テーブルとして表示アイコン をクリックします。2つの `Person` 要素が1つのテーブルに統合されます。子要素と子属性の名前が列のヘッダになり、それらの値がテーブルの行（レコード）になります。

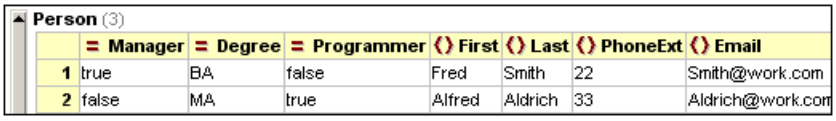


3. メニューオプションから 表示 | 最適な幅」を選択するか、最適な幅アイコン をクリックし、テーブルの列幅を最適化します。

注意 : テーブルを選択 (テーブルの要素名をクリック) し、テーブルとして表示アイコン をクリックすることで、テーブルビューを停止し、ドキュメント内にある個々の要素を表示することができます。

テーブルビューでコンテンツを入力する

2つ目の `Person` 要素のコンテンツ入力を行うには、2行目にあるそれぞれのテーブルセルをダブルクリックし、何らかのデータを入力します (下のスクリーンショットを参照)。しかしファイルを妥当なものにするためには `PhoneExt` は 99 までの整数でなければならないことを忘れないでください。テーブルのセル内でもインテリジェント編集機能は使用できます。従ってオプションがある箇所 (Boolean コンテンツと `Degree` 属性の列挙) ではドロップダウンリストから値を選択できます。



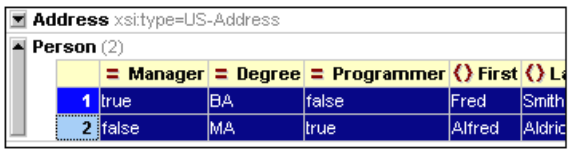
注意 : テーブルとして表示されている要素や属性に対しても入力ヘルパーは使用できます。例えば要素入力ヘルパー内の `Person` エンティをダブルクリックすることで、テーブルに新しい行 (新しい `Person` 要素) を追加することができます。

サードパーティ製品とのXMLデータのコピー

スプレッドシートで表示されたデータをXMLSpy のXMLドキュメントとサードパーティの製品間でコピーすることができます。このデータはXMLSpy 内でXML データとして扱うこともできますし、サードパーティ製品にてネイティブフォーマットとして使用することもできます。このセクションではExcel データシートとデータのやり取りの方法を学習します。

以下を行います :

1. 行ラベル 1をクリックします。 **「Ctrl」** キーを押下しながら行ラベル 2をクリックします。これでテーブル内の両方の行が選択されます。



2. メニューオプションから **「編集 | 構造化テキストとしてコピー」** を選択します。このコマンドで要素が画面に表示されたままの形式でクリップボードにコピーされます。
3. Excel に切り替え、XML データをExcel ワークシートに貼りつけます (Ctrl+V)。

A	B	C	D	E	F	G	H
TRUE	BA	FALSE	Fred	Smith	22	Smith@work.com	
FALSE	MA	TRUE	Alfred	Aldrich	33	Aldrich@work.com	

4. Excel にて新しい行にデータを入力します。`PhoneExt` 要素に3桁の数字（例えば 444）を入力してください。

A	B	C	D	E	F	G	H
TRUE	BA	FALSE	Fred	Smith	22	Smith@work.com	
FALSE	MA	TRUE	Alfred	Aldrich	33	Aldrich@work.com	
TRUE	PhD	FALSE	Colin	Coletti	444	Coletti@work.com	

5. Excel でテーブルのデータをマークし、**「Ctrl」**+C によりデータをクリップボードにコピーします。
6. XMLSpy に戻ります。
7. XMLSpy のテーブル内の左上にあるデータセルをクリックして、**「編集 | 貼付け」**を選択します。

Address xsi:type=US-Address
Person (3)

	= Manager	= Degree	= Programmer	{} First	{} Last	{} Pho
1	TRUE	BA	FALSE	Fred	Smith	22
2	FALSE	MA	TRUE	Alfred	Aldrich	33
3	TRUE	PhD	FALSE	Colin	Coletti	444

8. テーブルに新しいデータが表示されます。
9. メニューオプションの **「編集 | 置換」**を使用して、大文字で表示された`TRUE` と`FALSE` をそれぞれ小文字の `true` と`false` に変更します。

列のコンテンツによりテーブルをソートする

テーブルビュー内のテーブルは列に対して昇順または降順でソートを行うことができます。ここでは苗字（Last）に従い `Person` テーブルをソートしてみましょう。

列の内容に対してテーブルのソートを行うには：

1. `Last` 列のヘッダをクリックして列を選択します。

Address xsi:type=US-Address
Person (3)

	= Manager	= Degree	= Programmer	{} First	{} Last	{} Phone
1	true	BA	false	Fred	Smith	22
2	false	MA	true	Alfred	Aldrich	33
3	true	PhD	false	Colin	Coletti	444

2. メニューオプションから「XML | テーブル | 昇順に並べ替え」を選択するか、昇順ソートアイコン をクリックすることで、テーブル全てがソートされます。列はハイライトされたままになります。

Address xsi:type=US-Address
Person (3)

	= Manager	= Degree	= Programmer	{} First	{} Last	{} Phone
1	false	MA	true	Alfred	Aldrich	33
2	true	PhD	false	Colin	Coletti	444
3	true	BA	false	Fred	Smith	22

)ソート操作によりテーブルの表示が変わっただけではなく、もともとのXMLドキュメントも変更されています。つまり Person 要素の順序が Last 要素の内容に従ってソートされたのです (テキストタブをクリックしてテキストビューで変更を確認してみてください)。

3. メニューオプションから「XML | 検証」を選択するか、「F8」キーを押します。'444'という値は PhoneExt 要素では許されないという内容のエラーメッセージが表示されます (スクリーンショットを参照)。不正な PhoneExt 要素がハイライトされます。

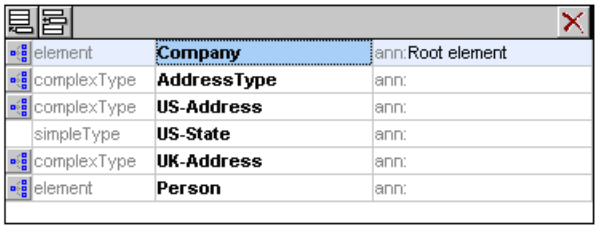
「詳細」を展開して、値が 99 以下でないため PhoneExt が不正であることを確認してください。エラーメッセージのリンクをクリックすることで XML ファイル内のエラーが発生した場所にジャンプすることができます。内線番号の為に設定した値のレンジではこの値はカバーされません。この値を使って妥当性を確保するために スキーマを修正する必要があります。次のセクションではスキーマの修正を行います。

6.8 スキーマの修正

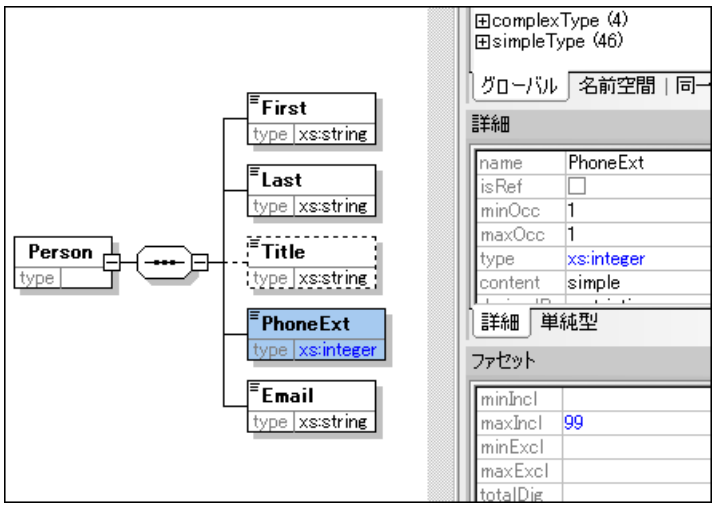
2桁の内線番号では、全ての番号をカバーできないことが分かりました。それでは有効な番号を3桁まで増やすことにしましょう。そのためにはスキーマの修正が必要になります。XML スキーマの修正はXML ドキュメントを閉じることなく行うことができます。

以下を行います：

1. メニューオプションから **「DTD/スキーマ| 定義へ移動」** を選択するか、定義へ移動アイコン をクリックします。割り当てられたスキーマ、この場合は `AddressLast.xsd` が開かれます（以下のスクリーンショットを参照）。必要に応じてスキーマビューへ移動します。オプションダイアログ（**「ツール| オプション」**）のファイルタイプタブでは、各ファイルタイプに対応したデフォルトのビューを変更することができます。



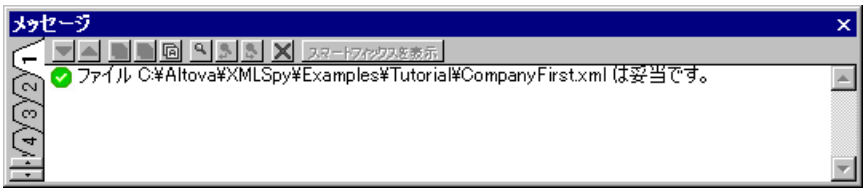
2. スキーマビューに表示されたグローバル `Person` 要素のダイアグラム表示アイコン をクリックし、`Person` 要素のコンテンツモデルにて `PhoneExt` 要素を選択します。ファセットタブ内にファセットデータが表示されます。



3. ファセットタブ内にある `maxIncl` の値フィールドをダブルクリックし、値を 99 から 999 に変更します、**「Enter」** で確定します。

4. スキーマドキュメントを保存します。
5. **「Ctrl」** + **「タブ」** キーを押してXML ドキュメントに移動します。

6. をクリックしてXML ドキュメントを再度検証します。



メッセージウィンドウにファイルが妥当であるというメッセージが表示されます。XML ドキュメントは修正されたスキーマに適合することが分かりました。

7. メニューオプションから「ファイル| 名前を付けて保存」を選択し、`CompanyLast.xml` という名前でファイルを保存します（XMLSpy に付随しているオリジナルの`CompanyLast.xml` ファイルを例えば `CompanyLast_orig.xml` の様な名前に変更するのを忘れないでください）。

メモ: XMLSpy に付随している`CompanyLast.xml` ファイルは`Tutorial` フォルダーの中に収められています。

7 XSLT 変換

目的
XML ファイルを変換するXSL スタイルシートを使ってXML ファイルからHTML ファイルを生成します。変換と言いますが、XML ファイルを変更するのではなく、新しいファイルが作成されます。「変換」という単語は慣例的な表現です。

方法
変換を行うために必要な方法は以下の通りです：

- 既に定義されたXSL ファイル、`Company.xsl` をXML ドキュメントに割り当てます。
- XMLSpy のインターフェースにある2つのAltova XSLT 内蔵エンジンのうち1つを使用して変換を実行します（下記参照）。

このセクションで使用されるコマンド
以下の XMLSpy コマンドがこのセクションでは使用されます：

「XSL/XQuery | XSLの割り当て」：アクティブなXML ドキュメントにXSL ファイルを割り当てます。

「XSL/XQuery | XSLへ移動」：XML ドキュメントにより参照されているXSL ファイルを開きます。

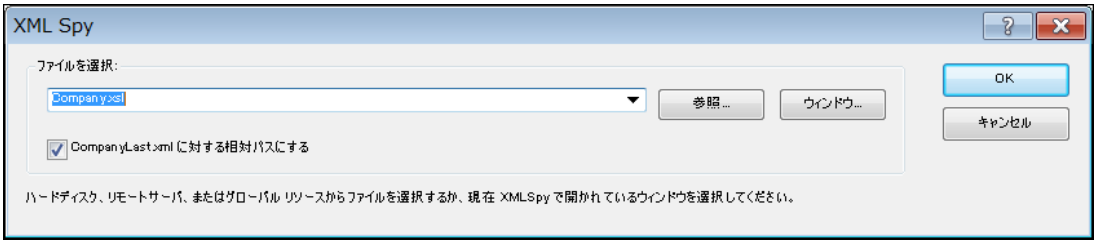
「XSL/XQuery | XSL変換（F10）」、または対応するツールバーアイコン：アクティブなXML ドキュメントをそのXML ファイルに割り当てられたXSL スタイルシートを使って変換します。XSL ファイルがまだ割り当てられていない場合、割り当てを行うプロンプトが表示されます。

注意：XMLSpy は 1.0、2.0、および 3.0XSLT エンジンを搭載しています。正しいエンジンは`xsl:stylesheet` か`xsl:transform` 要素のバージョン情報をもとに、XMLSpy により自動的に選択されます。このチュートリアルの変換では、XSLT 1.0 スタイルシートが使用されます。XSL 変換 コマンドが起動されたときには、変換のためのXSLT 1.0 エンジンが自動的に選択されます。

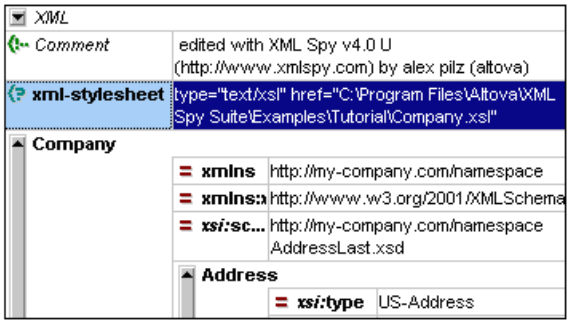
7.1 XSLT ファイルの割り当て

CompanyLast.xml ファイルにXSLT ファイルを割り当てるには：:

1. メインウィンドウのCompanyLast.xml タブをクリックしてCompanyLast.xml をアクティブにし、テキストビューに切り替えます。
2. メニューオプションの **「XSL/XQuery | XSLの割り当て」** を選択します。
3. 参照ボタンをクリックして、Tutorial フォルダーからCompany.xsl ファイルを選択します。XML ドキュメント内に記述されるXSL ファイルへのパスを相対パスにしたい場合は、**「CompanyLast.xml からの相対パスにする」** にチェックを入れます。



4. **「OK」** をクリックしてXSL ファイルをXML ドキュメントに割り当てます。
5. グリッドビューに切り替えて割り当てを確認します（下のスクリーンショットを参照）。

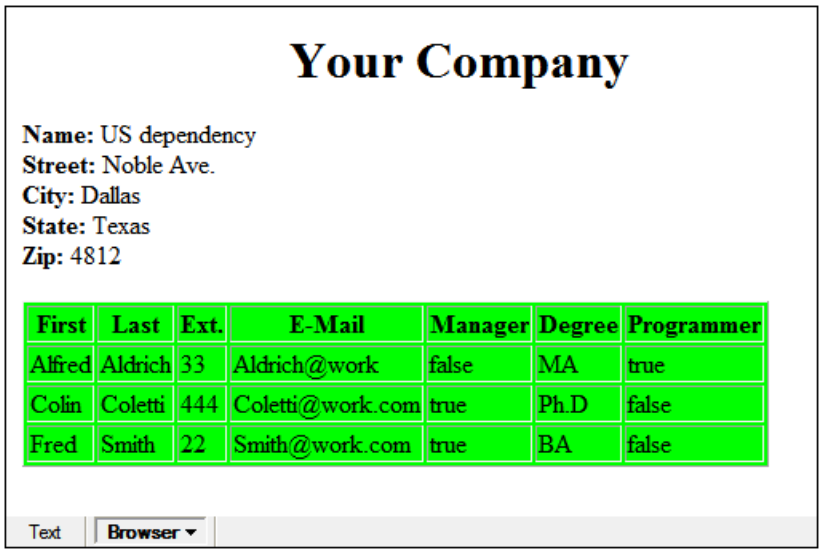


XSL ファイルを参照するXML-stylesheet 処理命令がXML ドキュメントファイルに挿入されます。**「CompanyLast.xml からの相対パスにする」** チェックボックスを有効にした場合、表示されるパスが相対パスになり、そうでない場合は絶対パスが使用されます。

7.2 XMLファイルの変換

割り当てを行ったXSLファイルを使用してXMLドキュメントを変換するには：

1. XMLファイルがアクティブになっていることを確認します。
2. メニューオプションから**「XSL/XQuery | XSL変換 (F10)」**を選択するかアイコン をクリックします。この操作によりXMLドキュメントで参照されているXSLスタイルシートを使用した変換が開始されます（`Company.xsl`ファイルはXSLT 1.0ドキュメントなので、変換にはAltova XSLT 1.0エンジンが自動的に選択されます）。出力されたドキュメントが`XSL output.html`というファイル名でブラウザービューに表示されます（HTML出力ファイルが生成されない場合、オプションダイアログ（ツール| オプション）のXSLタブにて、**出力ファイルのデフォルト拡張子**」を`html`に設定してください）。HTMLドキュメントには会社（Company）の情報と、人物（Person）の情報が表示されます。



メモ:出力ファイルの内容にテーブルのヘッダだけが表示され、その内容が表示されない場合、スキーマのターゲット名前空間を、このチュートリアルのはじめにある名前空間の定義で記述された通りになっているか確認してください。名前空間はすべてのファイル（スキーマ、XML、そしてXSL）で一致していなければなりません。

7.3 XSLファイルの修正

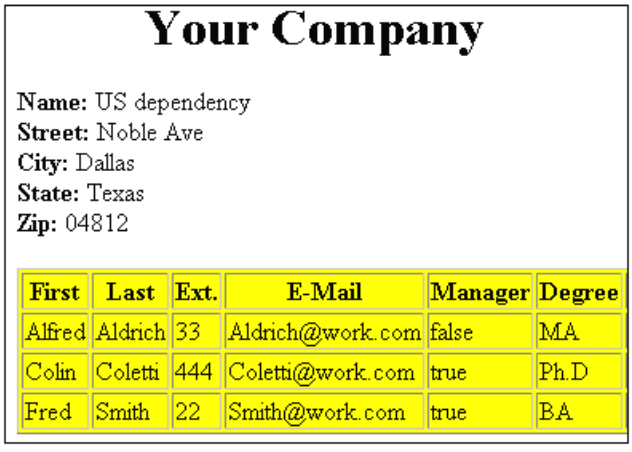
XSL ドキュメントを修正することで、出力ファイルを変更することができます。例えば、HTML 出力のテーブルの背景色をライムから黄色に変更してみましょう。

以下を行います：

1. CompanyLast.xml タブをクリックしてドキュメントをアクティブにします。グリッドビューにいることを確認してください。
2. メニューオプションから **「XSL/XQuery | XSLへ移動」** を選択します。このコマンドでXML ドキュメントにより参照されているCompany.xsl ファイルが開かれます。

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3  xmlns:xsi ="http://www.w3.org/2000/10/XMLSchema-instance"
4  xmlns:my="http://my-company.com/namespace">
5
6  <xsl:template match="/">
7      <html>
8          <head>  <title>Your company</title></head>
9              <body>
10                 <h1><center>Your Company</center></h1>
11                 <xsl:apply-templates select="//my:Address"/>
12                     <table border="1" bgcolor="lime">
13                         <thead align="center">
14                             <td><strong>First</strong></td>
15                             <td><strong>Last</strong></td>
16                             <td><strong>Ext.</strong></td>
17                             <td><strong>E-Mail</strong></td>
18                             <td><strong>Manager</strong></td>
19                             <td><strong>Degree</strong></td>
20                             <td><strong>Programmer</strong></td>
21                         </thead>
22                         <xsl:apply-templates select="//my:Person"/>
23                     </table>
24             </body>
25     </html>
26 </xsl:template>
```

3. <テーブル border="1" bgcolor="lime"> という行を見つけ出し、bgcolor="lime" をbgcolor="yellow" に変更します。
4. メニューオプション **「ファイル | 保存」** を選択して、XSL ファイルの変更を保存します。
5. CompanyLast.xml タブをクリックしてXML ファイルをアクティブにします。その後 **「XSL/XQuery | XSL変換」** を選択するか **「F10」** を押します。新しいXSL Output.html ファイルがXMLSpy のブラウザービューに表示されます。テーブルの背景色が黄色になっています。



Your Company

Name: US dependency
Street: Noble Ave
City: Dallas
State: Texas
Zip: 04812

First	Last	Ext.	E-Mail	Manager	Degree
Alfred	Aldrich	33	Aldrich@work.com	false	MA
Colin	Coletti	444	Coletti@work.com	true	Ph.D
Fred	Smith	22	Smith@work.com	true	BA

6. メニューオプションから **「ファイル | 保存」** を選択し、Company.html という名前でドキュメントを保存します。

8 プロジェクト管理

このセクションではXMLSpy のプロジェクト管理機能を紹介します。XML ファイルをプロジェクトにより管理する利点を学習した後、作成したファイルを簡単なプロジェクトで管理することにします。

8.1 プロジェクトの利点

XML ファイルをプロジェクトにより管理すると以下のような利点があります：

- ファイルの拡張子やその他の条件により、複数のファイルや URL をフォルダーにまとめることができます。
- フォルダーまたはプロジェクト全体に対してバッチ処理を行うことができます。
- DTD または XML スキーマを指定したフォルダーに割り当て、フォルダー内のファイルに対して検証を行うことができます。
- XSLT ファイルを指定したフォルダーに割り当て、割り当てた XSLT を使ってそのフォルダー内にある XML ファイルの変換を行うことができます。
- XSL 変換の出力ファイルの変換先を指定することができます。

上のプロジェクト設定は、全てメニューオプションの **「プロジェクト | プロパティ」** にて行うことができます。次のセクションでは、プロジェクトメニューを使ってプロジェクトを作成します。

- メニューオプションの **「プロジェクト | ソース管理 | ソース管理に追加」** により XML ファイルをソース管理に置くことができます（詳しくはオンラインマニュアルの ソース管理セクション を参照してください）。
- パーソナルネットワーク ならびに ウェブフォルダー もプロジェクトに追加し、バッチ検証を行うことができます。

8.2 プロジェクトの作成

これまでの過程で、メインウィンドウ内には様々なファイルが開かれているはずです。これらのファイルをチュートリアルプロジェクトの中にまとめることができます。まずは新しいプロジェクトを作成し、その後チュートリアルのファイルを対応するプロジェクトのサブフォルダーに追加していきます。

基本的なプロジェクトの作成

新しいプロジェクトを作成するには：

1. メニューオプションから **「プロジェクト | 新規プロジェクトを作成」** を選択します。新しいプロジェクト という名前の新しいプロジェクトがプロジェクトウィンドウに作成されます。新しいプロジェクトには典型的な XML ファイルのカテゴリフォルダーが作成されます。

2. CompanyLast.xml タブをクリックし、メインウィンドウの CompanyLast.xml ファイルをアクティブにします。
3. メニューオプションから **「プロジェクト | プロジェクトにアクティブなら びに関係するファイルを追加」** を選択します。CompanyLast.xml と AddressLast.xsd の2つのファイルがプロジェクトに追加されます。処理命令で参照されるファイル（例えばXSLT ファイル）は関係するファイルとは見なされないことに注意してください。
4. メニューオプションから **「プロジェクト | プロジェクトを保存」** を選択します。プロジェクトを Tutorial という名前で保存します。

プロジェクトにファイルを追加する

プロジェクトには他のファイルを追加することもできます：

1. CompanyLast.xml 以外の開かれている任意のXML ファイル（.xml の拡張子がついたファイル）を選択して、アクティブにします（XML ファイルが他に開かれていない場合は、XML ファイルを開くか作成してください）。
2. メニューオプションから **「プロジェクト | プロジェクトにアクティブなファイルを追加」** を選択します。その .xml ファイルの種類をもとに、XML ファイルがプロジェクトの XML ファイル フォルダーに追加されます。
3. 同様に、HTML ファイルとXSD ファイル（例えば Company.html と AddressFirst.xsd ファイル）をプロジェクトに追加します。これらのファイルはそれぞれ HTML File フォルダーとDTD/Schemas フォルダーに追加されます。
4. メニューオプションの **「プロジェクト | プロジェクトを保存」** を選択するか、プロジェクトウィンドウ内のファイルまたはフォルダーを選択し、ツールバーの保存アイコン（または **「ファイル | 保存」**）をクリックしてプロジェクトを保存します。

メモ： プロジェクトフォルダーを右クリックして「アクティブなファイルを追加」を選択することで、そのフォルダーにアクティブなファイルを追加することができます。

その他の便利なコマンド

その他にも良く使われる以下のプロジェクトコマンドがあります：

- 新しいフォルダーをプロジェクトに保存するには、**「プロジェクト | プロジェクトにプロジェクトフォルダーを追加」** を選択して、プロジェクトフォルダーの名前を入力します。

- プロジェクトのフォルダーを削除するには、フォルダーを右クリックして、コンテキストメニューから **「削除」**を選択します。
  プロジェクトのファイルを削除するには、ファイルを右クリックして **「Delete」** キーを押下します。

9 おしまい

ここまでくれば、おめでとう。そしてありがとう!

このチュートリアルはXMLSpy の基本を学習するのに役立ったでしょうか? より詳しい情報を知りたいときには、オンラインヘルプを参照するか、XMLSpy アプリケーションフォルダーにあるPDF 版のチュートリアル (`tutorial.pdf`) を印刷してください。

Altova XMLSpy 2017 Enterprise Edition

ユーザーマニュアル

ユーザーマニュアル

このドキュメンテーションのユーザーマニュアルパートはXMLSpy を使い始めるにあたって必要な全ての情報と、XMLSpy の様々な機能について記述されています。主に4つのパートで構成されています：(i) イントロダクション；(ii) チュートリアル(iii) 使用方法；そして (iv) ユーザーリファレンスです。

まずはイントロダクションから読み始め、GUI と重要なアプリケーションの設定について理解するべきでしょう。初めてXML に触れるのであれば、XMLSpy チュートリアルでXMLSpy の機能を学習するだけではなく、XML ドキュメントが簡単に作成、使用できることを学習することができます。その後、もっと重要な編集ビューセクションならびに使用方法に進んで下さい。ユーザーリファレンスのパートはリファレンスとして使用することができます。

イントロダクション

イントロダクションでは、GUI、オプションダイアログでの重要な設定、そしてアプリケーション環境に関する解説を行ないます。

チュートリアル

XMLSpy チュートリアルでは最もよく使用されるXMLSpy の機能についての解説が行われ、初めての操作に慣れ親しむことができます。

使用方法

使用方法に関するパートでは、XML やXML に関係する様々な技術を使うにあたりXMLSpy が提供する機能について説明します。まずはXMLSpy の複数の編集ビューの説明が行われ、次に個々のテクノロジーの機能についての解説が行われます。使用方法パートの最後では、アプリケーション全般のXMLSpy の様々な機能についての説明が行われます。使用方法のセクションは以下の様になっています：

- 編集ビュー：XMLSpy の様々な編集ビューについて記述します。
- XML：XMLSpy でXML ドキュメントを扱う際に利用可能な、様々な機能について説明します。
- DTD ならびにXML スキーマ：XMLSpy を使ったスキーマ (DTD ならびにXML スキーマ) の編集と利用方法について記述します。
- XSLT とXQuery：XSLT とXQuery の開発に使用することのできる機能の説明を行います。
- Authentic：Altova のグラフィカルなXML エディターと、その使用方法について説明します。
- HTML とCSS: XMLSpy のHTML とCSS に対するサポートについて説明します。
- JSON とJSON スキーマ: JSON 編集サポートについて説明します：テキストビューおよびグリッドビュー内のJSON ドキュメント、およびスキーマビュー内のJSON スキーマについて説明します。
- Avro、Avro スキーマ: Avro データファイルとAvro スキーマのためのサポートについて説明します。
- WSDL とSOAP：これらのプロトコルを使用するにあたりXMLSpy のメカニズムについて説明します。
- XBRL：XMLSpy のXBRL ビューにおけるタクソノミエディターの扱い方について説明します。
- Office Open XML ファイルとZIP ファイル：これらのスタンダードを使用したドキュメントの編集やマージを行うことができるArchive View について説明します。
- データベース：データベースとXML をつなぎ合わせるXMLSpy の様々な機能について説明します。
- グローバルリソース：Altova 社製品、特にパッケージソフトを使用した開発の効率を高めるために利用することのできるAltova のユニークなメカニズムについて解説します。
- プロジェクト：開発効率を高め、付随する開発オプションを有効にするXMLSpy のプロジェクトメカニズムについて説明します。
- ファイルディレクトリの比較：XMLSpy に内蔵されている比較エンジンの紹介を行います。
- ソース管理: XMLSpy と作業するためにどのようにソース管理を設定するか説明します。
- Visual Studio 内のXMLSpy：Visual Studio 開発プラットホームへの統合について説明します。
- Eclipse 内のXMLSpy：Eclipse 開発プラットホームへの統合について説明します。
- コード生成：スキーマ定義に従ったプログラムコードを生成する内蔵のコード生成機能について説明します。

メニューコマンド

メニューコマンドはレファレンスのセクションの一部で、XMLSpy のメニューに従い整理されており、各メニューコマンドの詳細を説明しています。

Windows XP、Windows Vista、Windows 7、 Windows 8 および Windows 10 でのファイル パス

このドキュメントのファイル パスはすべてのオペレーティングシステムで同じではありません。以下の通信に注意してください：

- *(マイ)ドキュメント フォルダー*：デフォルトでは以下の場所に存在します。Example ファイルはこのフォルダのサブフォルダに存在します。

Windows XP	`C:\Documents and Settings\<username>\My Documents`
Windows Vista、Windows 7/8/10	`C:\Users\<username>\Documents`

- *アプリケーション フォルダー*：アプリケーション フォルダーには Altova アプリケーションが存在します。デフォルトでの、アプリケーションフォルダのパスは以下の通りです。

Windows XP	`C:\Program Files\Altova\`
Windows Vista、Windows 7/8/10	`C:\Program Files\Altova\`
64-bit OS 上での32bit バージョン	`C:\Program Files (x86)\Altova\`

メモ： XMLSpy は、Windows Server 2003/2008/2012/2016 でもサポートされています。

1 インターフェイスと環境

このセクションでは以下の説明を行ないます:

- アプリケーションGUI
- アプリケーション環境

GUI のセクションでは、最初にGUI の概要を説明した後、各 GUI ウィンドウの詳細について解説していきます。ウィンドウならびにGUI のサイズ変更、移動、そして操作方法などについても説明します。

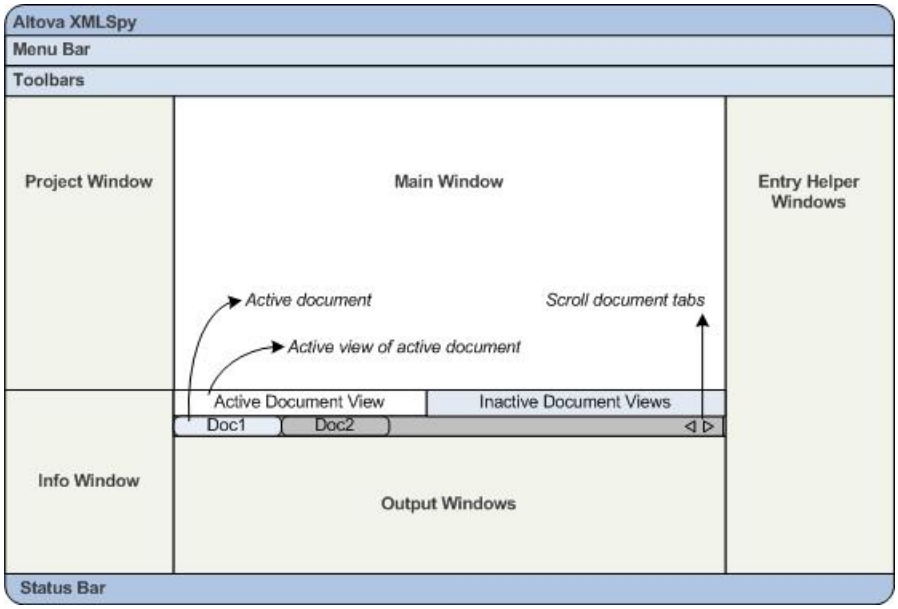
アプリケーション環境のセクションでは、ファイルの管理を行うために必要な、様々な設定の表示ならびに変更方法について説明します。アプリケーションのカスタマイズを行う場所と、その方法についても記述されます。このセクションでは、あなたのコンピューター内にインストールされている、重要なサンプルファイルやチュートリアルファイルの場所についても言及されています。セクションの最後にはAltova Web サイトへのリンクが掲載されており、アプリケーションが持つ機能の一覧や、様々な形式のユーザーマニュアル、利用可能なサポートオプション、そしてAltova が提供するその他の製品に関する情報を参照することができます。

1.1 グラフィカルユーザインターフェース (GUI)

グラフィカルユーザインタフェース (GUI) は、メインウィンドウと、幾つかのサイドバーにより構成されています (以下の図を参照)。デフォルトでは、サイドバーはメインウィンドウの周りに配置されており、以下の様にグループ分けされています：

- プロジェクトウィンドウ
- 情報 ウィンドウ
- 入力ヘルパ (要素、属性、エンティティなど 現在アクティブなドキュメントの種類により変化)
- 出力ウィンドウ (メッセージ、、XPath、XSL Outline、ファイル内の検索、スキーマ内の検索、、XBRL 内の検索、チャート

このセクションのサブセクションではメインウィンドウならびにサイドバーについて説明します。



GUI にはメニューバーやステータスバー、ツールバーなども含まれており、このセクションにて全て説明されます。

サイドバー表示の切り替え

「**ウィンドウ**」メニューのコマンドを使用することで、サイドバーグループのウィンドウ (プロジェクトウィンドウ Info ウィンドウ 入力ヘルパ 出力ウィンドウ) の表示を切り替えることができます。表示されているサイドバー (またはタブによりグループ化されたサイドバー) のタイトルバーを右クリックして、「**隠す**」コマンドを選択することでも、サイドバーを非表示にすることができます。

サイドバーのフロートとドッキング

各サイドバーウィンドウは、GUI から自由な状態のフロート、または GUI に結合したドッキング状態にセットすることができます。フロート状態のウィンドウがドッキング状態になると、最後にドッキングしていた場所にドッキングすることになります。ウィンドウはタブとして別のウィンドウにドッキングすることもできます。

以下の方法を使うことで、ウィンドウをフロートまたはドッキング状態にすることができます：

- ウィンドウのタイトルバーを右クリックして、必要なコマンド (**フロート**または**ドッキング**) を選択する。
- ウィンドウのタイトルバーをダブルクリックする。ウィンドウがドッキングしている状態であれば、フロ

ート状態になり、フロート状態であれば、最後にドッキングしていた位置にドッキングします。
- (タイトルバーを使って) ドッキング状態のウィンドウをドラッグすることで、フロート状態に切り替える。フロート状態のウィンドウをドラッグすることで、目的の位置にドッキングすることもできます。2つのセットの矢印が表示されます。外側に表示される4つの矢印により、アプリケーションウィンドウに対して相対的なドッキングを行うことができます。内側に表示される矢印を使うことで、現在カーソルが位置しているウィンドウに対して相対的なドッキングを行うことができます。内側に表示されている矢印の中央でボタンをドロップすると、ドラッグしていたウィンドウが、ドロップされたウィンドウのタブとしてドッキングされます。

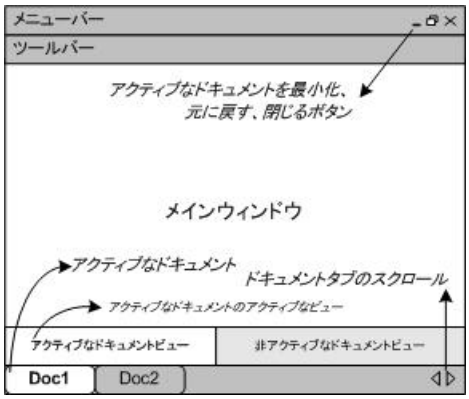
タブ化されたウィンドウをフロート状態にするには、そのタブをダブルクリックしてください。タブをドラッグすることで、タブ化されたウィンドウをグループから切り離すことができます。

サイドバーを自動的に隠す
自動的に隠す機能により、ドッキング状態のサイドバーがアプリケーションウィンドウの端に寄せられ、サイズを最小限に抑えることができます。この機能により、メインウィンドウやその他のサイドバーを、より大きな空間で使用することができるようになります。最小化されたサイドバーにマウスを移動させることで、サイドバーが再度表示されます。

サイドバーを自動的に隠したり、その設定を解除するには、サイドバーウィンドウのタイトルバーにある自動的に隠すアイコンをクリックするか、タイトルバーを右クリックして **自動的に隠す**」を選択します。

1.1.1 メイン ウィンドウ

(以下の図に表示される) メインウィンドウではドキュメントの編集を行ないます。



メインウィンドウのファイル

- 開いたり編集を行うファイルの数に制限はありません。
- 開かれた各ドキュメントは、各自ウィンドウに表示され、その名前がメインウィンドウの下部にあるタブに表示されます。ドキュメントをアクティブにするには、そのドキュメントのタブをクリックしてください。
- 複数のファイルが開かれている場合、ドキュメントタブバーに収まりきらず、一部のドキュメントタブが表示されないことがあります。目的のドキュメントタブは、(i) ドキュメントタブバーの右にあるスクロールボタンを使うことで、または (ii) **ウィンドウ** メニューから目的のドキュメントを選択することでアクティブにすることができます。
- アクティブなドキュメントが最大化されると、**最小化**」**元に戻す**」、**閉じる**」ボタンがメニューバーの右側に配置されます。ドキュメントが重ねて表示、または最小化されると、**最大化**」、**元に戻す**」、**閉じる**」ボタンがドキュメントウィンドウのタイトルバーに配置されます。
- 1つのファイルを最大化すると、開かれている他のファイルも全て最大化されます

- **ウィンドウ** メニューのコマンドを使用することで、開かれているドキュメントを重ねて、または上下 左右) に並べて表示することができます。
- 開かれているファイルは、 「Ctrl+Tab」 または 「Ctrl+F6」により、開かれた順序でアクティブに切り替えることができます。
- ドキュメントタブを右クリックすることで表示されるコンテキストメニューから、 **印刷**」や **閉じる**」といったコマンドを実行することができます。

メインウィンドウ内のビュー
アクティブなドキュメントを表示、編集する方法は何通りかあります。使用することのできるビューは、 上の図にあるように) ドキュメントタブ上のバーに表示されており、アクティブなビューがハイライトされます。目的のビューボタンをクリック または **表示** メニューのコマンドを使用することでビューの切り替えを行うことができます。

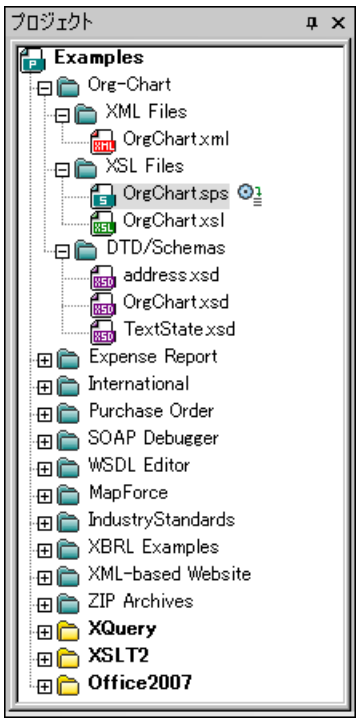
利用可能なビューは編集またはブラウザビューです：

- **テキストビュー**: 構文による色分けなどの機能を備えたソースコードレベルの編集を行うビュー
- **グリッドビュー**: 構造化された編集を行うビュー。グラフィカルな編集を行うことのできる構造化されたグリッドの形式で、ドキュメントは表示されます。このビューには **テーブルビュー** が内蔵されており、繰り返し出現する要素をテーブル形式で表示することができます。
- **スキーマビュー**: XML スキーマの表示や編集を行ないます。
- **WSDL ビュー**: WSDL ドキュメントの表示と編集を行ないます。
- **Authentic ビュー**: StyleVision Power Stylesheet をベースにしたXML ドキュメントの編集をグラフィカルなインターフェイスで行ないます。
- **ブラウザビュー**: CSS ならびにXSL スタイルシートをサポートする統合されたブラウザビューです。

メモ: 各ファイル拡張子のデフォルトビューは、 **ツール | オプション**」ダイアログの、ファイルタブにあるデフォルトビューペインにてカスタマイズすることができます。

1.1.2 プロジェクトウィンドウ

プロジェクトとは、何らかの形で関係するファイル同士を集めた集合のことです。例えば、以下のスクリーンショットでは、Examples という名前のプロジェクトに、様々なサンプルファイルが、適当なフォルダーに収められており、各フォルダーにはサブフォルダーを作成することもできます。例えば、Examples プロジェクト内のOrgChart サンプルフォルダーにはXML、XSL、そしてスキーマファイルを管理するためのサブフォルダーがあります。



プロジェクトを使用することで、一緒に使用されるファイルをまとめて、ファイルへのアクセスを素早く行うことができます。更に、各フォルダーに対してスキーマや XSLT ファイルを定義することもでき、フォルダーに対するバッチ処理を行うこともできます。

プロジェクトの操作

フォルダーの操作を行うコマンドは、**プロジェクト**メニュー、またはプロジェクトやフォルダーの (右クリックにより表示される) コンテキストメニューからアクセスすることができます。

- プロジェクトウィンドウで同時に開くことのできるプロジェクトは 1つだけです。新規プロジェクトが作成、または既存のプロジェクトが開かれると、現在プロジェクトウィンドウにて開かれているプロジェクトは置き換えられます。
- プロジェクトに変更がなされた場合、そのプロジェクトを保存する必要があります。プロジェクトの保存は「**プロジェクト | プロジェクトを保存**」メニューコマンドにより行うことができます。
- プロジェクトはフォルダー、ファイル、その他のリソースなどにより構成されるツリー構造により表示されます。リソースを追加する場所や階層に制限はありません。
- プロジェクトフォルダーは、論理的にファイルのグループ化を行う*論理* フォルダーです。ハードディスク上にある階層構造などに対応する**必要はありません。**
- フォルダーは、実際のファイルシステムに対して、直接対応するように作成することもできます。このようなフォルダーは*外部フォルダー* と呼ばれ、プロジェクトウィンドウでは黄色のフォルダーアイコンにより表示されます (通常のプロジェクトフォルダーは緑色のフォルダーアイコンで表示されます)。外部フォルダーは、**更新**コマンドにより、明示的に同期を行う必要があります。
- フォルダー内に収められるファイルの種類やその組み合わせに制限はありません。また、フォルダーのプロパティダイアログから各フォルダーにファイルの拡張子を定義することで、共通するファイルを1つの場所に収めることができます。ファイルが親フォルダーに追加されると、そのファイル拡張子が定義されたサブフォルダーに、ファイルが自動的に追加されます。

- プロジェクトウィンドウでは、フォルダーをドラッグすることで、他のフォルダーや、同じ親フォルダー内の異なる場所に、そのフォルダーを移動することができます。ファイルをドラッグすることで他のフォルダーに移動することはできますが、フォルダー内部のファイルはアルファベット順に表示されており、同じフォルダー内での移動を行うことはできません。また、Windows エクスプローラーなどからファイルやフォルダーをプロジェクトウィンドウにドラッグすることで、プロジェクトへの追加を行うこともできます。

- 各フォルダーには、フォルダーのプロパティダイアログにて定義されるプロパティがあります。これらのプロパティには、フォルダーのファイル拡張子、XML ファイルの検証を行うスキーマファイル、XML ファイルの変換を行うXSLT ファイルなどが記述されます。
- フォルダー内にあるファイルのバッチ処理を行うには、フォルダーを右クリックし、コンテキストメニューから選択することのできるコマンドを選択します (例：**XML の検証**または**整形式のチェック**)。

更に詳細なプロジェクトの使用方法に関しては、プロジェクトのセクションを参照ください。

メモ: プロジェクトウィンドウの表示は**ウィンドウ**メニューから切り替えることができます。

1.1.3 情報ウィンドウ

情報ウィンドウには、現在カーソルが位置している要素や属性に関する情報が表示されます (以下のスクリーンショットを参照)。情報はテキストビュー、グリッドビュー、XBRL ビュー、およびAuthentic ビュー情報ウィンドウで確認することができます

情報ウィンドウの表示は**ウィンドウ**メニューから切り替えることができます。.

以下の点に注意してください：

- XSLT ファイルがアクティブな場合、情報ウィンドウの XSLT タブには XSLT 固有の情報やコマンドが表示されます (以下のスクリーンショットの XSLT タブを参照)。XSLT タブに表示される情報やコマンドの読み方については、XSLT とXQuery | XSLT | XSL アウトラインのセクションを参照ください。
- XML スキーマドキュメント (xsd ファイル) がアクティブな場合、情報 ウィンドウのスキーマタブにて、拡張スキーマ検証を有効にすることができます。拡張検証の設定方法に関しては、スキーマルールのセクションを参照ください。

1.1.4 入力ヘルパー

入力ヘルパーは、妥当なXML ドキュメントを素早く作成するための、インテリジェントな編集機能です。ドキュメントの編集中、現在カーソルがある位置にて利用可能なオプションが入力ヘルパーには表示されます。入力ヘルパーに表示される情報は、DTD、XML スキーマ、またはStyleVision Power Stylesheet から取得されます。例えば、XML データドキュメントの編集を行っている場合、現在カーソルのある位置で挿入可能な要素、属性、エンティティが、それぞれの入力ヘルパーでにて表示されます。

利用可能な入力ヘルパーは以下の条件により変化します：

1. *編集中のドキュメントの種類* :例えば、XML ドキュメントには要素、属性、エンティティ入力ヘルパーが表示されるのに対して、XQuery ドキュメントでは XQuery キーワード、変数、そして関数入力ヘルパーが表示されます。各ドキュメントの種類と利用可能な入力ヘルパーについては、ユーザーマニュアルに関するセクションセクションで記述されます。
2. *現在のビュー* :編集メカニズムはアクティブなビューにより変化するため、入力ヘルパーも使用されているビューの編集メカニズムに適合するようにデザインされています。例えば、テキストビューにおいて、要素はカーソルがある位置にしか挿入することができず、入力ヘルパーの要素がダブルクリックされると、カーソル位置に要素が挿入されるようになっています。グリッド

ビューの場合は、選択された要素の前、または後に新たな要素を挿入、または追加することができるばかりか、子ノードとして追加することもできます。そのため、グリッドビューの入力ヘルパーには、挿入、追加、子を追加タブが表示されます。

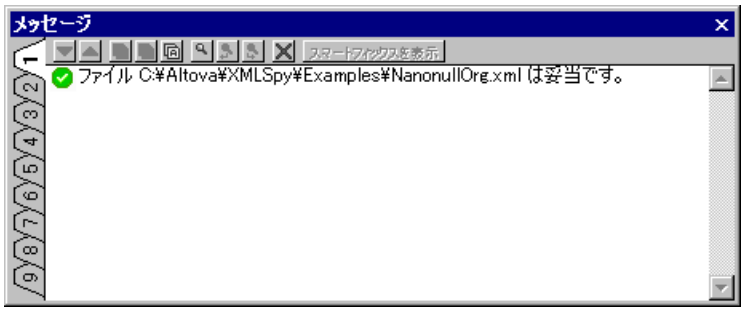
各ビューにおける入力ヘルパーの詳細については編集ビューのセクションを参照ください。異なるドキュメントの種類と入力ヘルパーの関係については、例えば、XML 入力ヘルパーや XQuery エントリヘルパーといった各ドキュメントの記述を参照ください。

以下の点に留意してください:

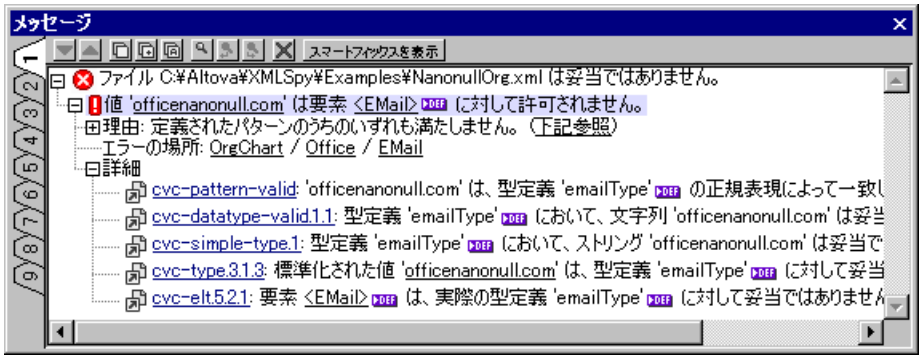
- 入力ヘルパーの表示は、メニューオプションの「**ウィンドウ | 入力ヘルパー**」から切り替えることができます。
- Visual Studio、.NET では、入力ヘルパーにはアプリケーション名のプレフィックスが与えられます。

1.1.5 出力ウィンドウ :メッセージ

メッセージウィンドウには XMLSpy にて行われた操作や処理の結果や、エラーやその他の出力などが表示されます。例えば、XML、XML スキーマ、DTD、または XQuery ドキュメントが検証され、そのドキュメントが妥当である場合、そのドキュメントが妥当であるというメッセージが表示されます (以下のスクリーンショットを参照)。



ドキュメントが妥当でない場合、(以下のスクリーンショットにあるように) エラーメッセージが表示されます。以下のスクリーンショットにある黒色で示されるリンクは XML ドキュメント内にあるノードへのリンクで、青色で表示されているリンクはインターネット上の関連する仕様とその説明へのリンクになります。紫色の DEF ボタンをクリックすることで、関連するスキーマ定義がスキーマビューにて開かれます。



メッセージウィンドウは全てのビューで有効になっており、XML ドキュメントにあるコンテンツへのリンクをクリックすることで、XML ドキュメントのノードがテキストビューでハイライトされます。XML スキーマがスキーマビューにて検証される際には、DEF ボタンをクリックしても、ビューの切り替えは行われません。

XML 検証スマートフィックス
テキストビューまたは**グリッドビュー**にて検証が行われると、スキーマ内にある情報をベースに、修正の候補 (スマートフィックス) が表示されます。スマートフィックスのオプションを参照するには、**スマートフィックスを表示**ボタンをクリックしてください (上のスクリーンショットを参照)。スマートフィックスの候補がメッセージウィンドウ内のペインに表示されます (以下のスクリーンショットを参照)。

メッセージウィンドウのエラーは一度に1つずつ表示されます。また、開始タグと終了タグがマッチしないなどの整形式に関するエラーは、検証エラーが表示される前に表示されます。そのため**スマートフィックスを表示**ボタンは、整形式に関するエラーが全て修正され、検証エラーの修正が行われる段階で有効になります。

スマートフィックスペインにて提示されたスマートフィックスの候補の1つを選択し、**フィックス + 検証**ボタンまたは**フィックス**ボタンをクリックしてください (上のスクリーンショットを参照)。XML ドキュメント内にある不正なテキストが選択されたスマートフィックスに置き換えられます。また、スマートフィックスをダブルクリックすることで、**オプション**のドロップダウンリストにて選択された内容に従うかたちでフィックスが適用、またはフィックスを適用した後に検証が行われます。**フィックス + 検証**コマンドを使用することにより、フィックスが行われた後に検証が再度行われ、別の検証エラーが存在するか確認することができます。

スマートフィックスペインを隠すには、**スマートフィックスを隠す**ボタンをクリックしてください (上のスクリーンショットを参照)。編集ビュー | スキーマビュー | 検証とフィックス。

検証フォルダーとプロジェクトウィンドウ内のファイル

XML メニュー以下にある**検証**コマンドは、通常アクティブなドキュメントに対して適用されますが、アクティブなプロジェクト内部にあるファイルやフォルダー、ファイルのグループに対しても適用することができます。目的のファイルやフォルダーをプロジェクトウィンドウにて (クリックにより) 選択し、メニューオプションの「**XML | XML の検証**」を選択するか **F8** キーを押下してください。ファイルが妥当でない場合、そのファイルがメインウィンドウにて開かれ、ファイルが妥当では無い旨のメッセージが表示されます。

メモ: プロジェクトウィンドウからは、整形式のチェック **整形式のチェック**コマンドまたは **F7**) を行うこともできます。

1.1.6 出力ウィンドウ :XPath/ XQuery

XPath/XQuery ウィンドウ (下のスクリーンショット) によりXML ドキュメントに対してXPath とXQuery 条件式 を評価しデバッグすることができます。ウィンドウは、以下の2つのモードで動作します :

- 評価モードでXPath またはXQuery 条件式が1つ以上のXML ドキュメントに対して評価されます。
- デバッグモードで現在アクティブなXML ドキュメントに対して適用し、XPath/XQuery 条件式をデバッグすること

ができます。ブレークポイントとトレースポイントを設定し、評価をステップバイステップで行うことができます。

2つのモードを切り替えるには、(ウィンドウのツールバーの左隅にある、*下のスクリーンショット参照*) **デバッグモード** ボタンを選択/選択解除します。モードの使用方法に関してはこのセクションのサブセクションで説明されています。XPath/XQuery ウィンドウ内でのXPath/XQuery 条件式の構文とサポートに関しては、XPath とXQuery 仕様情報を参照してください。

メモ: デバッグモード ボタンへの切り替えは、オプションリスト内で現在のファイルオプションが選択されている場合のみ使用することができます (*下のスクリーンショット参照*)。

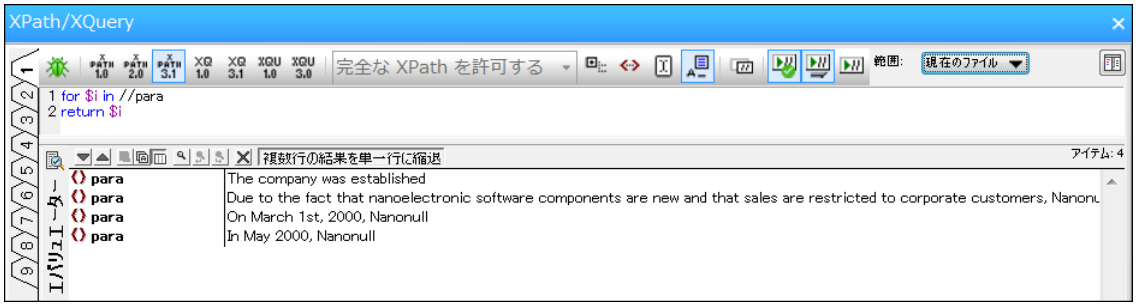
このセクションのボタン

デバッグモード

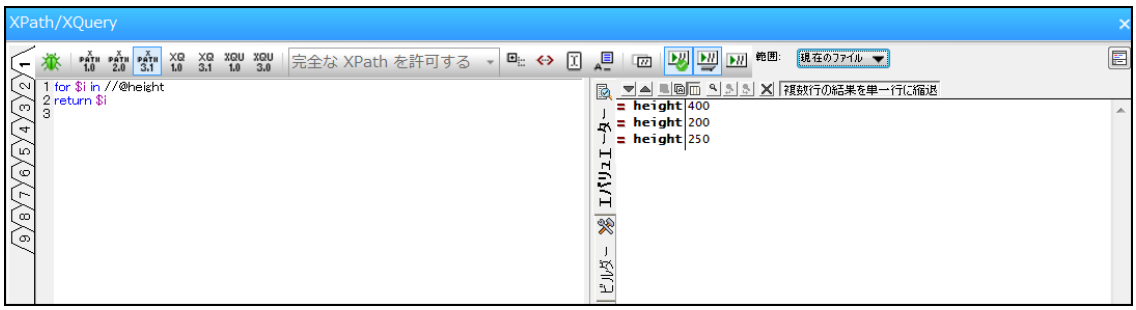
左右と上下のレイアウト

左右と上下のレイアウト

(両方のモードの) ツールバーの右端に、(*下のスクリーンショットでハイライトされている*) ボタンがあります。このボタンを使用して、左右と上下のレイアウトを切り替えることができます。レイアウトはいつでも切り替えることが可能です。下のスクリーンショットは、上下のレイアウトが表示されており、このレイアウトは広範囲におよぶ上下のラインにより構成された結果を表示するために役にたちます。



上下例合ウ戸 (*下のスクリーンショット*) は、複数ラインにおよぶXPath/XQuery 条件式 (左側のペイン) を表示するために役にたちます。



複数のタブ

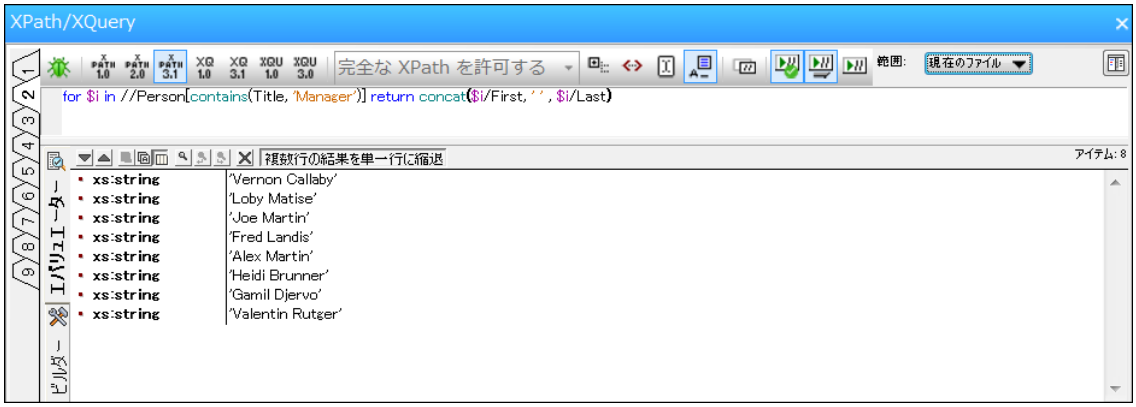
XPath/XQuery ウィンドウには9つのタブがあり、9つの異なる条件式と同時に作業することができます。タブに条件式を入力した後、タブのモードが保存されます。評価モード で条件式が入力された場合、タブの条件式と評価設定が保存されま

す。この結果、各タブの設定は保存されます。**新規** タブに切り替える場合、前に選択されたタブの設定が新しいタブにパスされます。

デバッガーが現在のタブで実行されていない限り、1つのタブから別のタブへいつでも切り替えることができます。通常、異なる XPath または XQuery 条件式を、各条件式を個別に評価またはデバッグし、結果を比較するために、異なるタブに入力します。

評価モード

XPath/XQuery ウィンドウにより、範囲オプション内で設定することのできる、*下のスクリーンショット参照*) それぞれの評価コンテキスト内で、一度に9つまで XPath または XQuery 条件式をビルドまたは評価することができます。評価は、アクティブな XML ドキュメント、と以下のコンテキスト内のカーソルに対して相対します。: (i) 現在開かれている XML ファイル、(i) アクティブな XMLSpy プロジェクトの XML ファイル、と (ii) 選択されたフォルダーの XML ファイル。



このセクションのボタン

	デバッグモード	デバッグと評価モード間を切り替えます
	左右と上下のレイアウト	左右と上下のレイアウト間を切り替えます

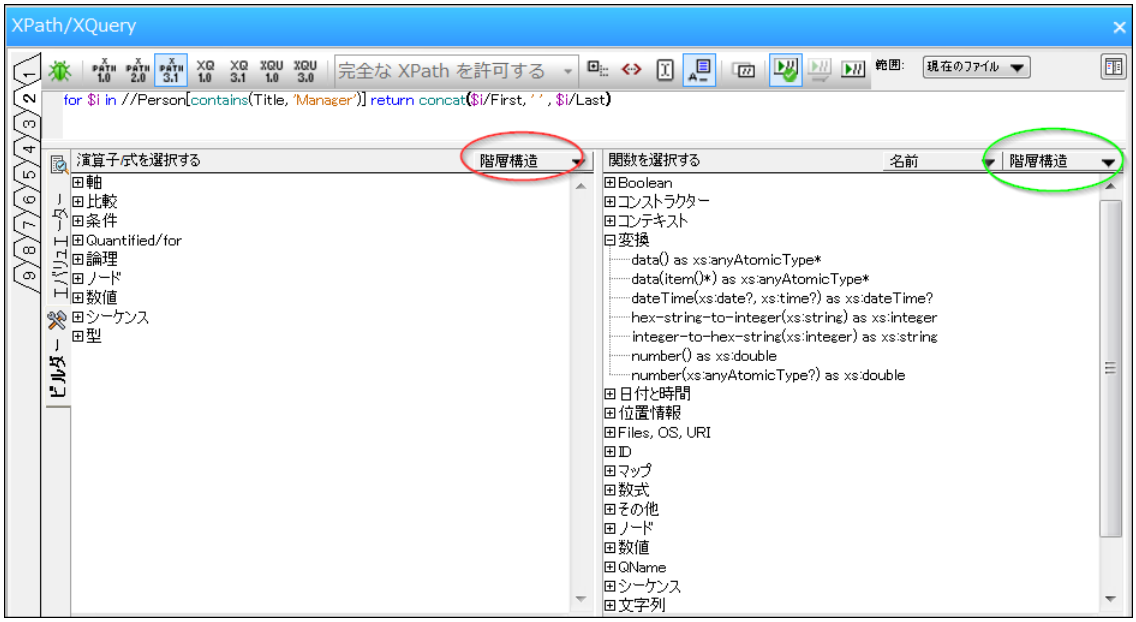
評価モードに関して

- 評価モードに切り替えるには、ウィンドウのツールバーの左端の**デバッグモード**ボタンをクリックします。(*上のスクリーンショット参照*).
- ウィンドウのツールバーの右端の**左右と上下のレイアウト**ボタンをクリックすることにより左右と上下のレイアウトを切り替えることができます (*上のスクリーンショット参照*)。
- 評価モードには、以下のタブがあります: (i) 式の構築を手伝う入力ヘルパーを搭載したビルダータブと、(i) 式の評価の結果を表示するエバリュエータータブ (*上のスクリーンショット参照*)。
- 下のペインが、2つのタブの名前を表示するには小さすぎる場合、名前は非表示になり、タブのアイコンのみが表示されます。マウスをアイコンにポイントするとタブの名前が表示されます。
- 評価モード内の XPath/XQuery ウィンドウには9つのタブがあります。タブに条件式を入力した後、タブのモードが保存されます。他のタブに切り替えた後、タブに戻ると、そのタブの設定は変更されていません。**新規** タブに切り替える場合、前に選択されたタブの設定が新しいタブにパスされます。
- 複数の XML ドキュメントのコンテキストで条件式が評価される場合 (下の範囲オプションに関する詳細を参照してください) 結果リスト内からファイル名をクリックすると、XMLSpy 内のファイルを開き、そのファイルをアクティブにします。

XPath 評価に関しては以下で説明されています。XQuery 評価 に関する詳細は、XQuery 評価を参照してください。(XQ アイコンは XQuery 評価のためです。XQU アイコンは XQuery Update 実行のためです。) XPath/XQuery ウィンドウ内で、XPath/XQuery 条件式を作成するには、XPath と XQuery 仕様情報を参照してください。

XPath/XQuery 式ビルダー

ビルダータブ (下のスクリーンショット) には、2つの入力ヘルパーペインがあります : (i) 演算子と条件式のためのペインと (ii) 関数のためのペイン。単一の関数名に1つ以上の署名が存在する場合、各署名は異なる関数としてリストされることに注意してください。 (これらは、関数名のオーバーロードとして知られています。) 両方のペインのアイテムは階層的またはフラットリストとしてグループ化することができます。各ペインの右上のドロップダウンリストからオプションを選択します (下のスクリーンショット内で赤線で囲まれています)。スクリーンショットでは、両方のペインはアイテムを階層的なグループで表示されています。



ビルダーの機能 :

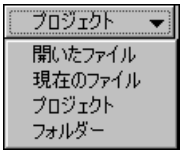
- 入力ヘルパーペイン内でアイテムの詳細を確認するには、アイテムにマウスをポイントします。
- 各関数は、署名と共にリストされています (すなわち、それぞれの引数、引数のデータ型、関数の出力のデータ型と共にリストされます。上のスクリーンショット内でハイライトされている *dateTime* 関数を参照してください)。
- 引数がある場合は、名前順またはデータ型別にリストされます。 (上のスクリーンショットで緑で縁取りされている) 関数ペインのタイトルバーのドロップダウンリストからオプションを選択します。A
- ペインのアイテム (演算子、条件式、または関数) をダブルクリックすると、条件式内のカーソルの場所にそのアイテムを挿入します。関数はプレースホルダー (# シンボル) により示された引数と共に挿入されます。
- (i) テキストが条件式の編集フィールド内で選択された場合、 (ii) プレースホルダーを含む演算子、条件式または関数がダブルクリックされ挿入された場合、選択されたテキストがプレースホルダーの代わりに挿入されます。これは、演算子、条件式、または関数に長いテキスト (パス条件式など) を挿入する素早い代替方法となります。

条件式に関数を入力した後、関数名にカーソルをポイントすると関数の署名と関数の説明テキストが表示されます。関数に1つ以上の署名がある場合、表示の下のオーバーロードファクターと共に表示されます。関数のかっこ内にカーソルを合わせ、**[Ctrl+Shift+Spacebar]** を押すと、その関数名の異なるオーバーロードの署名を確認することができます。

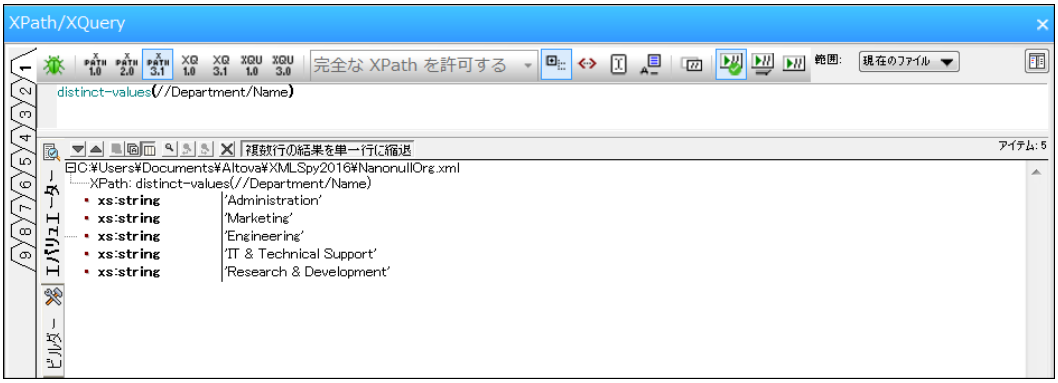
条件式の評価

以下にXPath 条件式の評価の主な手順が説明されています。XQuery 条件式評価の手順も同じです。 a dedicated description of XQuery 評価に関する詳細は、XQuery 評価のセクションを参照してください。 (The XQ アイコンare for XQuery 評価; the XQU アイコンare for XQuery Update 実行.)

1. XPath 条件式の評価される箇所により、範囲オプションリストからオプションを選択します (下のスクリーンショット): *現在のファイル*、*ファイルを開き* (*XMLSpy*)、*プロジェクト*、または*フォルダー*。



   現在のファイルが選択されている場合、現在アクティブなファイルが使用されています。 ファイルを開くを選択すると、XPath 条件式がXMLSpy内で現在開かれている全てのファイルに対して評価されます。プロジェクトは、現在アクティブな XMLSpy プロジェクトを参照します。 XMLSpy プロジェクト内の外部フォルダーは、外部フォルダーをスキップするチェックボックスをチェックすることで除外することができます。フォルダーオプションを使用して、必要なフォルダーを参照することができます。XPath 条件式はこのフォルダー内のXML ファイルに対して評価されます。
2. 使用するXPath バージョン (1.0、2.0、または3.1) を出力ウィンドウのツールバー内の適切なアイコンをクリックして選択します (下のスクリーンショット参照)。
3. ドロップダウンリストからXPath 条件式の型を選択します。このコンボボックスの他のオプションは、特定のXPath 1.0 ケースの狭義のサブセットのために使用することのできる一意のID 制約 XML スキーマがスキーマ内で定義された場合役に立つ セレクターとXML スキーマフィールドです。どちらかのオプションが選択されている場合、名前テスト (とワイルドカード *) のみがXPath 条件式 内での使用を許可されており、述語とXPath 関数を使用できない可能性があります。また、XML スキーマセレクターオプションに関しては、子の軸上の条件式のみが許可されています。XML スキーマフィールドオプションに関しては、子の軸と属性の軸上の条件式が許可されています。更に詳しい情報に関しては、W3C のXML スキーマ:構造の推奨 を参照してください。選択するXPath バージョンがXPath 1.0 でない場合、コンボボックスは無効化されます。完全なXPath の許可がデフォルトで選択されており、変更することはできません。
4. 入力中に評価する場合は、入力中にXPath 条件式を評価アイコン をオンに切り替えます。このアイコンがオフに切り替えられている場合、XPath 条件式を評価アイコン をクリックしている場合のみ条件式は評価されます。
5. XPath 条件式とXML ファイルの場所を表示するためには、出力内のヘッダーの表示アイコン をオンに切り替えます (下のスクリーンショットに表示されるとおり)。
6. XPath 条件式が要素または属性などのノードを返した場合、選択されたノードのコンテンツ全体が表示されるかを選択することができます。これは結果全体を表示 アイコン をスイッチオンし行います。



7. XPath 条件式 をXML ドキュメント内の選択に対応させるために設定するには、現在の選択をオリジナルとして設定 アイコン をオンに切り替えます。

8. *XML ファイルの検証* アイコン をクリックして、XPath/XQuery 条件式のために評価されている XML ファイルを検証します。
9. XPath 条件式を入力します。複数に及ぶ条件式を (読みやすさのために) 作成する場合、**[Return]** キーを使用します。
10. 条件式フィールドのテキストのサイズを拡大または縮小することができます。これを行うには、条件式フィールド内をクリックして、**Ctrl** を押しながらスクロールホイールで調整します。
11. ノードの式のロケーターパスを手動で入力する代わりに、XMLSpy が入力することができます。以下を行ってください : (i) ロケーターパスを入力する XPath 条件式の箇所にカーソルを合わせます。 (ii) ターゲットとするノードの開始タグにカーソルを合わせます。 (iii) In XPath/XQuery ウィンドウ内で、条件式の選択されたノードのロケーターパスを挿入するために編集フィールドに現在の選択の *XPath* をコピーするアイコン をクリックします。ドキュメントのルートノードから開始する絶対パスとして、ロケーターパスは入力されます。
12. *XPath 自動補完の表示* アイコン がオンに切り替えられている場合、この時点で挿入することが有効な XPath 関数、軸、ドキュメント要素、属性のリストを表示したポップアップが表示されます。
13. 条件式を評価するには、 (タイプ中に評価がオフに切り替えられており、新しい XML ドキュメントがアクティブに設定された場合) *XPath 条件式の評価* アイコンをクリックします。

エバリュエータータブの結果ペイン

エバリュエータータブには、結果ペイン (*上のスクリーンショット参照*) があります。ペインには、次の機能があります :

- (*範囲オプションで指定されている*) 複数のファイルで評価が行われる場合、各ファイルの結果はそのファイルのパスの下に個別にリストされます (*上のスクリーンショット参照*)。
- 結果が (テキストノードを含む) ノードを含む場合、 ? 条件式により生成された *テラル*に対し? 結果ペイン内でそのノードをクリックすると、メインウィンドウ内の XML ドキュメント内の対応するノードをハイライトします。
- 結果は 2 行のカラムから構成されます : (i) ノード名またはデータ型、 (ii) ノードの値。結果サブラインまたは値カラムのみのカラムをコピーすることができます。サブラインを右クリックして、**[全てのカラムを含みコピーする]** の切り替えをオン/オフします。 (代わりに、ツールバー内のアイコンをしようしてコマンドのオン/オフを切り替えることができます)。コピーするサブラインを右クリックして、**[サブラインをコピー]** または **[全てコピーする]** を選択します。

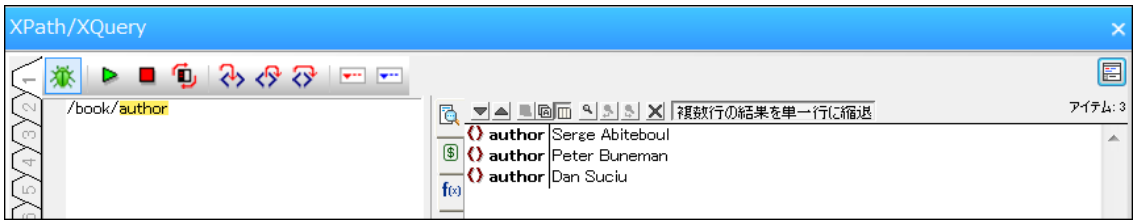
結果タブ (*上のスクリーンショット参照*) は、ナビゲート、検索、およびコピー機能を与えるアイコンを含みます。これらのアイコンは、左から始まる順序で下のテーブルで説明されています。結果リストアイテムのコンテキストメニュー内で対応するコマンドを使用することができます。

アイコン	動作
次、前	結果リストからそれぞれ次のおよび前のアイテムを選択します。
クリップボードに選択されたテキストラインをコピーする	クリップボードに選択された結果アイテムの値カラムをコピーします。全てのカラムをコピーするには、全てのカラムを含みコピーするコマンドに切り替えてください (*下を参照*)。
全てのメッセージをクリップボードにコピーする	空の値を含む値カラムのすべての結果アイテムをクリップボードにコピーします。各アイテムは異なるラインにコピーされます。
全てのカラムを含みコピーする	(i) 全てのカラム、または (i) 値カラムのコピー間を切り替えます。カラムの区切りは単一のスペースです。
検索する	結果リスト内で特別文字を含む文字列を検索する検索ダイアログを開きます。
前を検索する	検索ダイアログ内に最後に入力された用語の前の発生を検索します。
次を検索する	検索ダイアログ内に最後に入力された用語の次の発生を検索します。
クリア	結果のリストをクリアする
複数のラインの結果を	結果アイテムの値カラムが複数ラインのテキストを含む場合、 (新しいラインまたは文

単一のラインに折りたたむ	字を含む場合)、複数ラインと単一ラインの表示を切り替えることができます。

デバッグモード

XPath/XQuery ウィンドウのデバッグモードで、アクティブなファイルのコンテキスト内でXPath/XQuery 条件式をデバッグすることができます。



- **デバッグモード** ボタンをクリックしてデバッグモードに切り替えます。評価モード内では、範囲オプションが現在のファイルに設定されている場合のみ、**デバッグモード** ボタンが有効化されます。これは、アクティブなファイルのコンテキストでのみデバッグが実行されるからです。
- デバッグモードには2つのペインがあります : (i) 条件式が入力されるペイン、と (ii) 結果ペイン。これらのペインは、上下または左右に分割することができます。これらのレイアウト間を切り替えるには、**左右と上下のレイアウト** ボタンを使用します。
- 結果ペインには、以下のタブがあります :結果、変数、とコールスタック。結果ペインの高さが3つのタブの名前をすべて表示するには、小さすぎる場合、名前は非表示になり、タブのアイコンのみが表示されます。アイコンにマウスをポイントするとタブの名前が表示されます。

このセクションのボタン

	デバッグモード	デバッグと評価モード間を切り替えます
	左右と上下のレイアウト	左右と上下のレイアウト間を切り替えます
	結果	結果ペインの結果タブに切り替えます
	変数	結果 ペインの変数タブに切り替えます
	コールスタック	結果ペインのコールスタックタブに切り替えます

デバッグステップ

XPath/XQuery 条件式をデバッグするおおまかなステップは、以下のとおりです :

1. 条件式を実行する対象のXML ファイルをアクティブなファイルにします。
2. 条件式ペインにXPath/XQuery 条件式を入力します。
3. ブレークポイントまたはトレースポイントを設定します。ブレークポイントは、評価が停止される場所です。トレースポイントでは、ブレークポイントノードの評価結果が記録され、評価内でトレースすることのできるパスが与えられます。
4. デバッガーを開始し、評価をステップバイステップで実行するステップイン/アウト/オーバー機能を使用します。

これらのステップの詳細は下で説明されています。

デバッガーの開始、停止、および再起動

デバッガーが停止されている場合、**デバッガーの開始**ボタンのみが有効化されます。デバッガーが開始され、停止するまで、**デバッガーの停止** と**デバッガーの再起動** ボタンが有効化されます。

このセクションのボタン

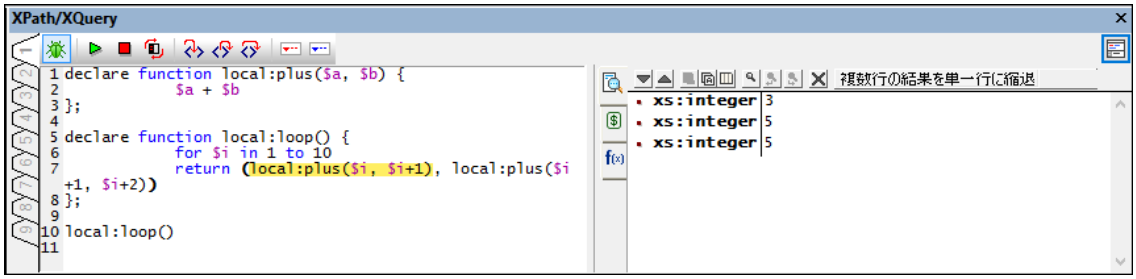
	デバッガーの開始 (Alt+F11)	デバッガーを開始します。デバッガーは条件式を最後まで評価します。デバッガーはブレークポイントのみで一時停止します。
	デバッガーの停止	評価を終了し、デバッガーを停止します。
	デバッガーの再起動	評価が停止すると (例えばブレークポイントなどで) 評価をはじめから行います。

ステップイン、アウト、オーバー評価ステップ

ステップ機能により評価をステップバイステップで実行することができます。それぞれの機能をクリックすることにより、対応する評価のステップを実行することができます。

このセクションのボタン

	ステップイン (F11)	評価をステップごとに行います。評価を詳しく行うステップを踏みます。
	ステップアウト (Shift+F11)	現在の評価ステップからステップアウトし、[親] ステップに移動します。
	ステップオーバー (Ctrl+F11)	[子孫] ステップをステップオーバーします。



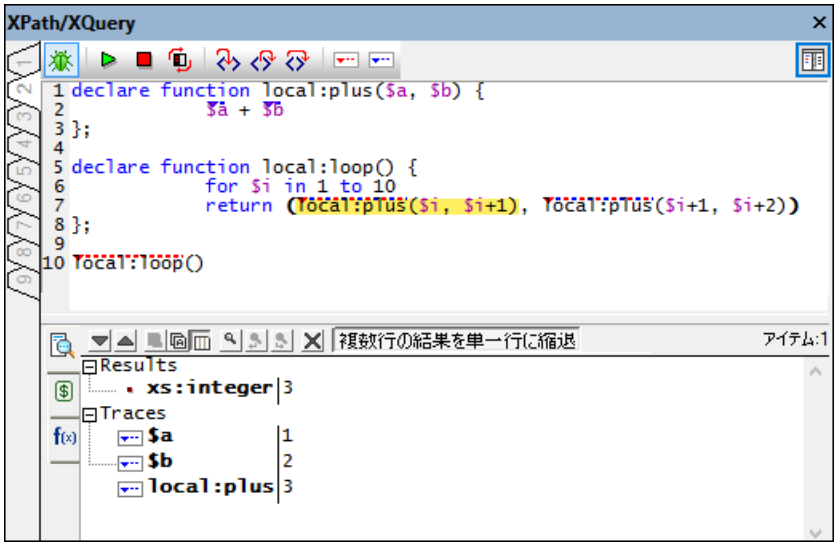
評価が進行すると、現在処理されているステップが条件式内でハイライトされ、結果タブ内に対応する結果が表示されます (*上のスクリーンショット参照*)。**[ステップイン]** は、停止するたびに最も詳しいデバッグ情報を提供しますが、**[ステップアウト]** と**[ステップオーバー]** は、条件式の評価の素早く確認できるデバッグ情報を与えます。

ステップイン、ステップアウト、ステップオーバー機能を試すための XQuery 条件式

```
declare function bcal:plus($a, $b) {
        $a + $b
};

declare function bcal:loop() {
        for $i in 1 to 10
        return (bcal:plus($i, $i+1), bcal:plus($i+1, $i+2))
};

bcal:loop()
```

ブレークポイントとトレースポイント

ブレークポイントとは、デバッガーが**[デバッガーの開始]**を使用して開始された後、停止するポイントです。デバッガーがブレークポイントで停止すると、結果を確認して、**[ステップイン]**機能を使用して、評価の次のステップの結果を表示します。ブレークポイントを設定するには、ブレークポイントを設定する場所にカーソルを合わせ、**[ブレークポイントの挿入 削除 (F9)]** ツールバーボタンをクリックします。ブレークポイントは赤い上線でマークされます (*下のスクリーンショット参照*)。ブレークポイントを削除するには、選択して、**[ブレークポイントの挿入 削除 (F9)]** をクリックします。



トレースポイントは、結果タブのトレースツリー内に結果が記録され表示されるブレークポイントです (*上のスクリーンショット参照*)。これにより、条件式の特定の部分の評価結果を確認することができます。例えば、上のスクリーンショットでは、トレースポイントは、`$a`, `$b`, `local:plus($i, $i+1)` と`local:plus($i+1, $i+2)`に設定されており、トレースポイントは、青い上線が引かれています。When スクリーンショット内でハイライトされている条件式の部分にデバッガーがある場合、そして、`$i` の値が`1` の場合、デバッガーが条件式の部分、結果タブでは、トレースポイント`$a`, `$b`, `local:plus($i, $i+1)` の条件式ノードの値がこの`$i` の値として表示されます。トレースポイントを設定するには、カーソルを条件式内の箇所に合わせツールバーボタン**[トレースポイントの挿入 削除 (F9)]** をクリックします。トレースポイントは青い点の上線で表示されます (*下のスクリーンショット参照*)。トレースポイントを削除するには、選択した後、**[トレースポイントの挿入 削除 (F9)]**をクリックしてください。 .

ブレークポイントとトレースポイント条件式の一部に対して設定されている場合、オーバーラインは赤と青の交互のダッシュにより構成されています (*上のスクリーンショット内で* `local:plus($i, $i+1)` と`local:plus($i+1, $i+2)` 関数の呼び出しを参照してください)。

結果ペインの結果タブ

結果タブ (*上のスクリーンショット参照*) は、ナビゲート、検索、およびコピー機能を与えるアイコンを含みます。これらのアイコンは、左から始まる順序で下のテーブルで説明されています。結果リストアイテムのコンテキストメニュー内で対応するコマンドを使用することができます。

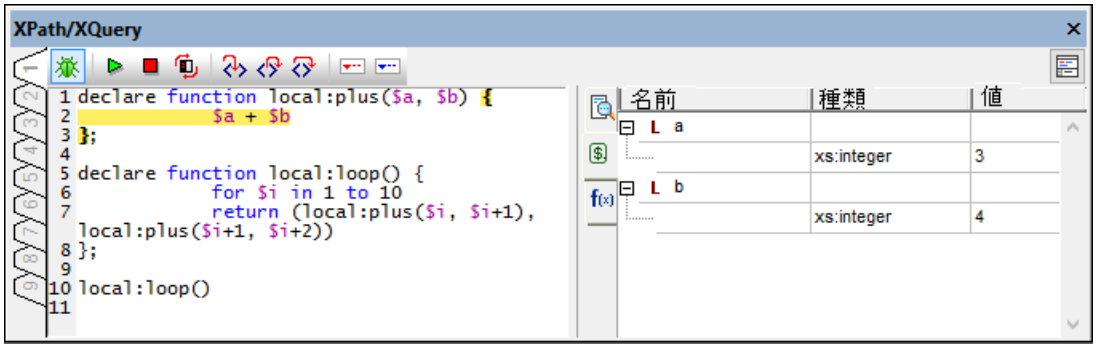
アイコン	動作
次、前	結果リストからそれぞれ次のおよび前のアイテムを選択します。

クリップボードに選択されたテキストラインをコピーする	クリップボードに選択された結果アイテムの値カラムをコピーします。全てのカラムをコピーするには、全てのカラムを含みコピーするコマンドに切り替えてください (*下を参照*)。
全てのメッセージをクリップボードにコピーする	空の値を含む値カラムのすべての結果アイテムをクリップボードにコピーします。各アイテムは異なるラインにコピーされます。
全てのカラムを含みコピーする	(i) 全てのカラム、または (i) 値カラムのコピー間を切り替えます。カラムの区切りは単一のスペースです。
検索する	結果リスト内で特別文字を含む文字列を検索する検索ダイアログを開きます。
前を検索する	検索ダイアログ内に最後に入力された用語の前の発生を検索します。
次を検索する	検索ダイアログ内に最後に入力された用語の次の発生を検索します。
クリア	結果のリストをクリアする
複数のラインの結果を単一のラインに折りたたむ	結果アイテムの値カラムが複数ラインのテキストを含む場合、(新しいラインまたは文字を含む場合)、複数ラインと単一ラインの表示を切り替えることができます。

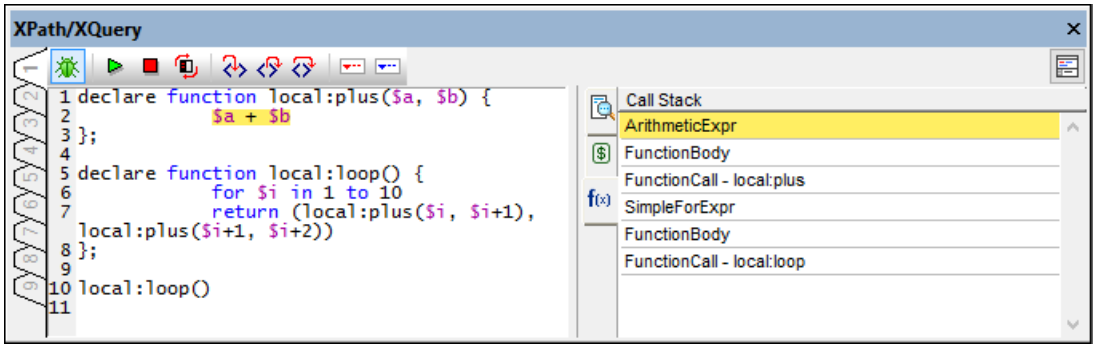
変数とコールスタック

結果ペインは結果タブ、*変数タブ*と*コールスタックタブ*などを含みます (*下のスクリーンショット*)。

デバッグ中、変数タブは、現在の*ステップ*とその値の変数を表示しています (*下のスクリーンショット参照*)。



コールスタックタブは、デバッグ内の箇所を呼び出す異なるプロセッサーを表示します。一番最近の呼び出しが上で、現在の呼び出しは双方のペインでハイライトされています (*下のスクリーンショット参照*)



XPath と XQuery 仕様情報

XPath 1.0 条件式

- XPath 1.0 関数は名前空間プレフィックス無しで入力する必要があります。
- 4つのノードテストがサポートされています：node()、text()、comment()、processing-instruction()。

XPath 2.0 と 3.1 条件式

- 文字列 (例：'Hello') や数値リテラル (例：256) がサポートされます。XML スキーマ型をベースにしたその他のリテラルを作成するには、名前空間プレフィックスによるコンストラクター (例：xs:date('2004-09-02')) を使用します。XML スキーマ型に使用する名前空間プレフィックスは、XML 名前空間 http://www.w3.org/2001/XMLSchema に関連付けられている必要があり、その名前空間もXML ファイル内で宣言されている必要があります。
- XPath の評価で使用されるXPath 2.0 およびXPath 3.1 関数は、名前空間 http://www.w3.org/2005/xpath-functions に所属しています。に所属しています。プレフィックスfn: は、慣習的にこの名前空間に関連付けられています。この名前空間はXPath の評価で使用されるデフォルトの関数名前空間で、関数にプレフィックスを与える必要はありません。プレフィックスを使用する場合は、そのプレフィックスがXML ドキュメント内で宣言されているXPath 関数の名前空間に関連付けられていることを確認してください。関数使用の例：current-date() (関数の名前空間はXML ドキュメント内で宣言されていない)、fn:current-date() (関数名前空間がXML ドキュメント内で宣言されていないか、XML ドキュメント内で宣言されており、プレフィックスfn: に関連付けられています)。XML ドキュメント内で関数名前空間が宣言されている場合でも、名前空間プレフィックスを省略することができます。この場合、そのXPath 条件式で使用される関数は、関数のデフォルト名前空間のものとなります。
- Altova XPath 拡張子は名前空間 http://www.altova.com/xslt-extensions 内にあります。

メモ: 名前空間に関する点を要約すると、XML スキーマ名前空間からコンストラクターを使用する場合、XML ドキュメント内でXML スキーマ名前空間を宣言し、XPath 条件式では正しい名前空間プレフィックスを使用する必要がある、ということになります。XPath 関数に対してプレフィックスを使用する必要はありません。

XPath 2.0 および XPath 3.1 のデータ型

XML スキーマを参照しているXML ドキュメントに対して、XPath 2.0 および 3.1 条件式を評価しており、そのXML ドキュメントがスキーマに対して妥当である場合、オペレーションにより暗黙的に目的のデータ型に変換されないデータ型に対して、コンストラクターまたはキャストを明示的に使用する必要があります。内蔵のXPath エンジンにより使用されるXPath データモデルでは、XML ドキュメントから**細分化された**ノード値には全て、xs:untypedAtomic データ型が割り当てられています。xs:untypedAtomic 型は暗黙の型変換で通常上手く機能します。例えば、xs:untypedAtomic("1") + 1 という条件式の結果は、xs:untypedAtomic 値が加算演算子により暗黙的にxs:double に変換されるため、2 となります。算術演算子は、オペランドを暗黙的にxs:double に変換します。比較演算子は、比較の前に、オペランドをxs:string に変換します。

目的のデータ型への明示的な変換を行う必要がある場合もあります。例えば、2つの要素 startDate とendDate があり、XML スキーマにて両者ともxs:date 型で定義されている場合、endDate - startDate というXPath 2.0 条件式ではエラーが発生します。この場合、xs:date(endDate) - xs:date(startDate) または (endDate cast as xs:date) - (startDate cast as xs:date) とすることで、xs:dayTimeDuration 型のシングルトンシーケンスが正しく評価されます。

メモ: XPath の評価で使用されるXPath エンジンは、Altova XSLT エンジンで使用されるものと同一のもので、XSLT スタイルシート内で目的のデータ型に暗示的に変換されないXPath 2.0 および 3.1

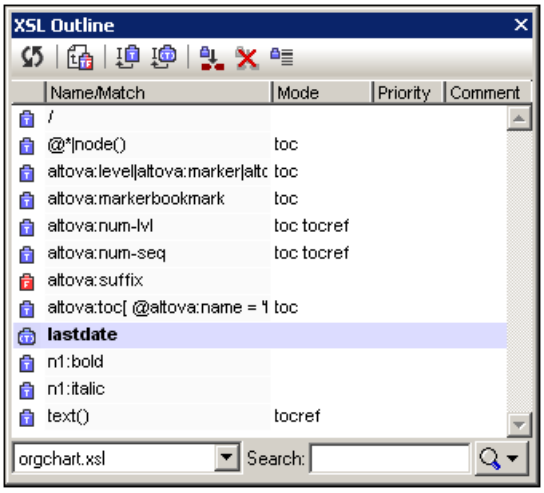
文字とエンティティ参照の文字列長

文字ならびにエンティティ参照が、string-length() 関数の入力文字列として使用される場合、参照を解決することはできず、解決されていないテキスト文字列の長さが返されます。しかしXSLT 環境において、これら参照は意味を持つことになり、解決された文字列の長さが返されます。

XPath 2.0 および 3.1関数サポート
付録を参照ください。

1.1.7 出力ウィンドウ :XSL アウトライン

XSLT スタイルシートや、 (オプションとして) インクルード、インポートされた全てのXSLT スタイルシートがXSL アウトラインウィンドウに表示されます (以下のスクリーンショットを参照)。XSL アウトラインウィンドウは、デフォルトでXMLSpy 下部の出力ウィンドウ内に配置されており、フロート状態にしたり、他のXMLSpy の端にドッキングさせることができます。

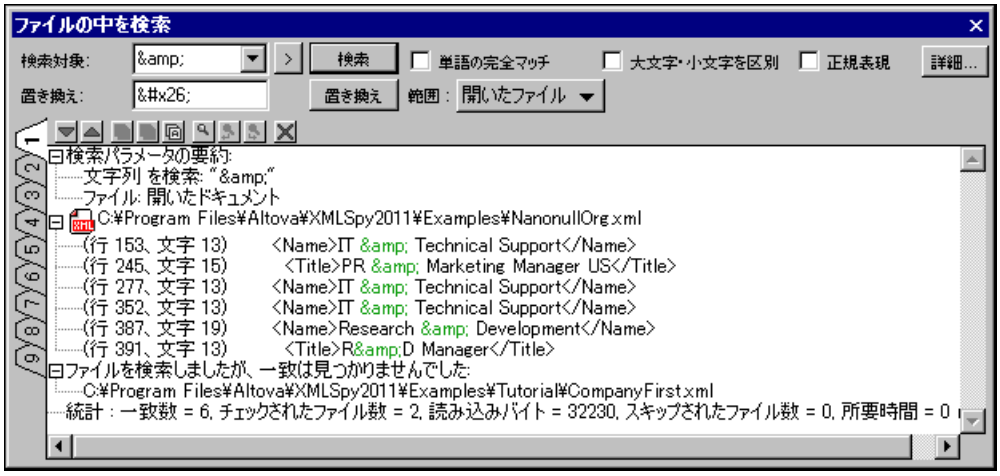


XSL アウトラインウィンドウでは、スタイルシート内のテンプレートや関数に関する情報が表示されます。ウィンドウでは、これら情報の並べ替えや検索を行ったり、ツールバーにあるコマンドを使用することで名前付きテンプレートに呼び出しを挿入したり、名前付きテンプレートを変換の開始点にセットすることができます。XSL アウトラインウィンドウの使用方法に関しては、XSLT とXQuery | XSLT | XSL アウトライン| XSL アウトラインウィンドウ のセクションを参照ください。

メモ: スタイルシートのファイルに関連した情報やコマンドは、Info ウィンドウの XSLT タブから使用することができます。これらコマンドの使用方法については、XSLT とXQuery | XSLT | XSL アウトライン| Info ウィンドウのセクションを参照ください。

1.1.8 出力ウィンドウ :ファイルの中を検索

ファイルの中を検索ウィンドウを使えば、一度に複数のドキュメントに対して検索ならびに置き換え操作を行ったり、発見されたインスタンスに対して素早くナビゲーションを行うメカニズムを使用することができます。検索ならびに置き換え操作の結果は、1から9までの番号付けされたタブに表示されます。結果タブに表示されている発見されたアイテムをクリックすると、ドキュメントがテキストビューで表示され、該当するアイテムに移動することができます。



検索条件

大まかに分けると、検索の条件には、(i) 何を検索するのかと、(ii) どこを検索するのか、の2つがあります。

検索する対象 検索を行う文字列は、検索対象テキストボックスに入力します。検索の対象が、単語として完全に一致する必要がある場合、単語の完全マッチチェックボックスにチェックを入れます。例えば、fit という検索文字列に対して単語の完全一致オプションを有効にすると、fit という単語に対してだけ検索が行われ、例えばfitness という単語は検索結果に含まれません。検索において大文字・小文字の区別をしたい場合は、大文字・小文字を区別チェックボックスにチェックを入れます。検索テキストボックスに入力された検索文字列が正規表現となる場合、正規表現チェックボックスにチェックを入れる必要があります。 ボタンをクリックすることで、正規表現の入力を簡単に行うことができるようになります。検索における正規表現の使用方法に関しては、検索のセクションで説明されます。**詳細**ボタンをクリックすることで、ファイルの中を検索が表示され、検索における高度な条件やアクションをセットすることができます。詳細については、編集 | ファイルの中を検索を参照ください。

検索範囲 検索は、(i) GUI 上に開かれている全てのファイル、(ii) 現在のプロジェクトにあるファイル、または (iii) 選択されたフォルダーのファイルに対して行うことができます。**詳細**ボタンをクリックすることで表示されるファイルの中を検索ダイアログで、更に詳細な条件を追加することができます。

置き換え

置き換えテキストボックスに入力された文字列により、検索の結果発見された文字列を置き換えることができます。置き換えテキストボックスが空の状態で**置き換え**ボタンをクリックすると、発見された文字列が空の文字列により置き換えられます。

結果

検索、または**置き換え**ボタンをクリックすると、その結果がファイルの中を検索出力ウィンドウに表示されます。表示される結果は4つのパートにより構成されます：

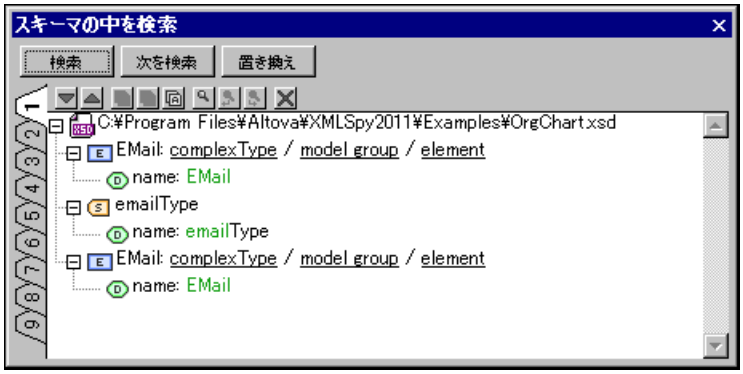
- 検索文字列や検索対象となった範囲を表示する、検索パラメータの要約。
- 検索または置き換えられた文字列のリスト 検索、または 置き換え」ボタンが押下により変化)。このリストに表示されるアイテムは、検索または置き換えがなされた箇所へのリンクとなり、クリックすることでドキュメントの該当箇所がテキストビューで表示されます。ドキュメントが開かれていない場合、そのドキュメントがテキストビューで開かれ、該当箇所がハイライトされます。

- 検索が行われたが、一致する結果が見つからなかったファイルのリスト。
- 一致した数や、チェックされたファイル数といった、検索アクションの統計情報。

メモ: ファイルの中を検索機能では、複数のファイルに対して、一度に**検索**や**置き換え**コマンドを実行し、その結果を出力ウィンドウに表示します。結果を逐次確認しながら進んでいく検索を行うには、検索コマンドを使用してください。

1.1.9 出力ウィンドウ :スキーマの中を検索

XML スキーマがスキーマビューでアクティブになっている場合、XMLSpy XMLSpy のスキーマビューでの検索ならびに置き換え機能を使用することができます。スキーマビューでの検索ならびに置き換え機能にアクセスするには、スキーマビューがアクティブになっている状態で (i) 編集メニューから、**検索**または**置換**コマンドを選択するか、(ii) スキーマウィンドウの**検索**または**置き換え**コマンドを選択します (以下のスクリーンショットを参照)。



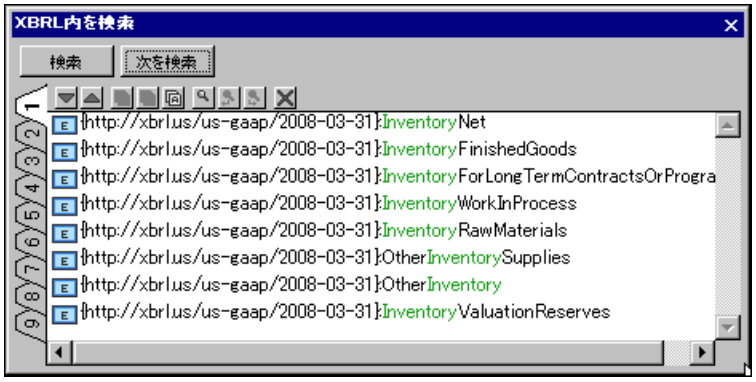
スキーマビューでの検索ならびに置き換えの結果は、スキーマの中を検索出力ウィンドウに表示されます。検索文字列は緑色で表示されます (上のスクリーンショットでは、email が検索文字列で、大文字・小文字の区別をしないように指定されたことが分かります)。スキーマファイル内の位置も示されていることに注目してください。

結果は (1から9と番号の振られた) 9つあるタブに表示されます。行われた検索の結果を1つのタブに保持しておき、新たな検索の結果を別のタブにて行うことで、検索結果を比較することができます。検索を行う前に新しいタブを選択することで、新たなタブにて検索が行われます。スキーマの中を検索ウィンドウ内に表示される結果をクリックすることで、メインウィンドウのスキーマビューにて、該当する箇所が表示、ハイライトされます。出力タブに表示されるメッセージはクリップボードにコピーすることもできます。詳細については、結果と情報 機能の、スキーマ内の検索 のセクションを参照ください。

1.1.10 出力ウィンドウ :XBRL 内を検索

XBRL 内を検索ウィンドウには、XBRL タクソノミドキュメントに対する検索の結果が表示されます。ウィンドウには9つのタブが含まれており、1つのタブにある結果を、別のタブに表示されている結果と比較することができます。

XBRL 内を検索する機能は、XBRL タクソノミがXBRL ビューで開かれているときに使用することができます。この検索機能の使用方法については、ユーザマニュアルの XBRL セクション にある XBRL 内を検索を参照ください。



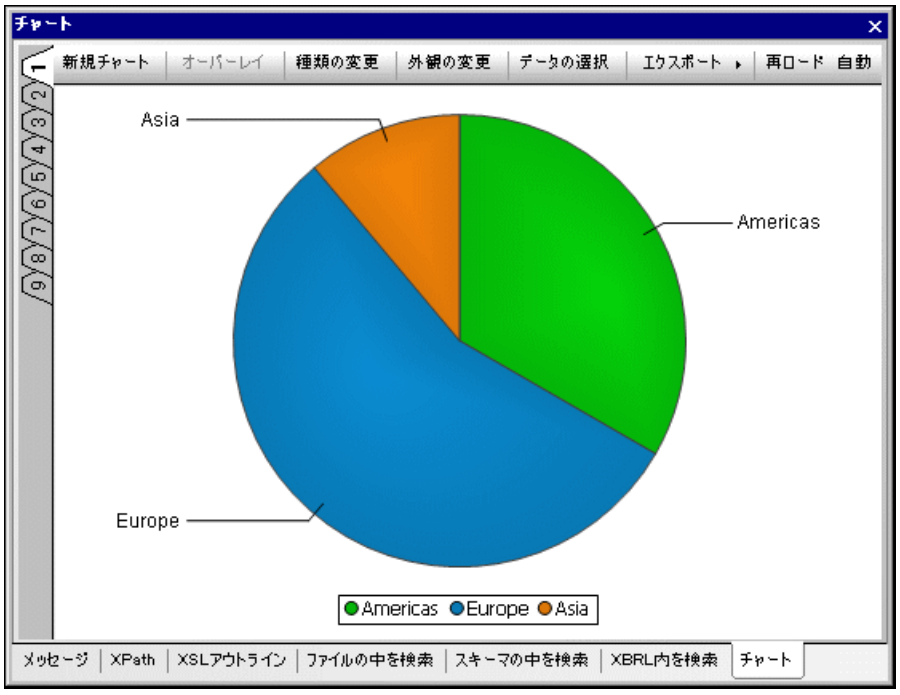
検索ボタンをクリックすることで、検索ダイアログが表示されます。検索が一度行われた後、**次を検索**ボタンをクリックすることで、XBRL ビューにて現在カーソルが位置している箇所を起点に、検索文字列に一致する次のインスタンスが出力ウインドウに表示されます。

以下のコマンドがXBRL 内で検索ツールバーで使用できます。

- **前**ならびに**後**アイコンにより、検索結果の選択箇所を移動します。
- **メッセージのコピー**コマンドにより、選択されたメッセージ、選択されたメッセージとその子供、全てのメッセージをクリップボードにコピーします。
- **検索**コマンドにより、XBRL 内を検索ウィンドウ内にある文字列の検索を行ないます。
- **クリア**コマンドにより、現在アクティブなタブ内のメッセージを全て削除します。

1.1.11 出力ウィンドウ :チャート

テキストビュー、またはグリッドビューでXML ドキュメントが開かれている場合、XML ドキュメント内で選択されているデータから、チャートウィンドウにグラフ (円グラフや棒グラフなど) を生成することができます。



チャートの作成

チャートを作成するための大まかなステップは以下のとおりです：

1. XML ドキュメント内にカーソルを配置し、コンテキストノードを選択します。
2. (上のスクリーンショットにあるように) チャートウィンドウ内の**新規チャート**ボタンをクリックするか、メインウィンドウ内で右クリックして、コンテキストメニューから**新規チャート**を選択します。.
3. 表示されるカラムの選択ダイアログにて、チャートデータテーブルに使用するデータを選択して、**OK** をクリックします。チャートウィンドウにてグラフが作成されます (上のスクリーンショットを参照)。

詳細については、ユーザマニュアルのXML セクションにある、チャート を参照ください。

チャートの修正と管理

グラフはチャートウィンドウにある (ウィンドウの左側にある) 9つのタブのどれかにて作成することができます。この機能を使うと、複数のチャートを別のタブに作成し、比較を行うことができます。あるタブに作成されたグラフは、そのタブで新たにグラフが作成されない限り置き換えられず、その他の方法で削除することもできません。チャートの生成のために使用されたXML ドキュメントが閉じられた後も、作成されたチャートはタブの中に保持されます。

ウィンドウの上部にあるボタンは以下のとおりです：

- *新規チャート* : カラムの選択ダイアログが表示され、チャートデータテーブルの設定を行うことができます。
- *オーバーレイ* : メインのチャート上にレイヤーを追加、または、削除することができます。新規のレイヤーを作成下の地、このレイヤーを新規のレイヤータブをクリックして、追加することができます。また、このオーバーレイチャートで使用されるデータを指定することもできます。
- *種類の変更* : グラフの種類を、例えば棒グラフや円グラフに変更します。
- *外観の変更* : グラフに表示されるフォントの大きさや、使用される色などを変更します。
- *データの選択* : チャートデータテーブルや、グラフにて表示されるデータの選択を行う、データの選択ダイアログが表示されます。このダイアログでは、データの系列や、X- 軸、Y- 軸などを修正することができます。X- 軸、Y- 軸のデータは、チャートデータテーブルからグラフィカルに選択することができます。**OK** をクリックすることで、チャートウィンドウに修正されたチャートが生成されます。
- *エクスポート* : チャートはイメージファイルとして、または XSLT や XQuery フラグメントとしてクリップボードにコピーすることができます。XSLT や XQuery フラグメントは、XSLT または XQuery ドキュメント内で使用することができ、Altova XSLT 2.0 エンジンや Altova XQuery エンジンにて、レンダリングが行われます。
- *再ロード*、*自動* : **自動**ボタンが有効になっていると、XML ドキュメントに変更が加えられると、その XML ドキュメントをベースにした、チャートウィンドウ内にあるグラフも描きかえられます。有効になっていない場合、**再ロード**ボタンが押されることで、グラフの更新が行われます。

詳細に関しては、ユーザーマニュアルのXML セクションのチャートセクションを参照してください。

1.1.12 メニューバー、ツールバー、ステータスバー

メニューバー

メニューバー (*図を参照*) には、様々なアプリケーションメニューが収められています。以下の慣習が適用されます：

- メニュー内のコマンドが、その時のビューや、カーソルの位置に対して適用**できない**場合、コマンドは無効になります。
- いくつかのメニューコマンドはサブメニューを持っており、更に別のオプションを選択することができます。サブメニューがあるメニューコマンドは、コマンド名の右側に表示される右方向矢印により示されます。

- いくつかのメニューコマンドはサブメニューを持っており、更に別のオプションを選択することができます。サブメニューがあるメニューコマンドは、コマンド名の右側に表示される右方向矢印により示されます。
- メニューコマンドにアクセスするには、メニュー名をクリックした後に、コマンドを選択します。メニューアイテムにサブメニューがある場合、そのメニューアイテムにマウスを移動することで、サブメニューが開かれます。その後、目的のサブメニューアイテムをクリックします。
- 適切なキーのコンビネーションにより、キーボードからメニューを開くこともできます。各メニューへのキーコンビネーションは **Alt+*KEY*** で、***KEY*** がメニュー名のキーになります。例えば、「**ファイル (F)**」メニューへのアクセスを行うには、**Alt+F** を押下します。
- (i) メニューに対するキーの組み合わせ (前の項目を参照) の後に、(i) 特定のコマンドに対するキーの組み合わせを入力することで、(メニュー内に収められているコマンドの) メニューコマンドへアクセスすることができます。例えば、新規ファイルを作成 (「**ファイル (F) | 新規作成 (N)**」) するには、**Alt+F** を押下した後、**Alt+N** を押下します。
- 特定のメニューコマンドは、**ショートカットキー**または、キーのコンビネーション (**Ctrl+*Key***) により直接選択することができます。ショートカットが割り当てられたコマンドは、コマンド名の右側にショートカットキーや、キーのコンビネーションが示されます。例えば、ショートカットキーのコンビネーションである **Ctrl+N** を使用することで、ファイルの新規作成を行うことができ、**F8** によりXML ファイルの検証を行うことができます。カスタマイズダイアログ (「**ツール | カスタマイズ**」) にあるキーボードタブにて、独自のショートカットを作成 することもできます

ツールバー

メニューバー (*図を参照*) には、メニューコマンドを選択するためのショートカットとなるアイコンが収められています。マウスポインターをアイコン上に移動させると、コマンドの名前が表示されます。アイコンをクリックすると、コマンドが実行されます。

ツールバーボタンはグループでまとめられています。「**ツール | カスタマイズ | ツールバー**」ダイアログでは、表示するツールバーのグループを選択することができます。これらの設定は現在のビューに対して適用されます。他のビューの設定を変更するには、そのビューに切り替えた後、「**ツール | カスタマイズ | ツールバー**」にて設定を変更します。GUI では、ツールバーグループのハンドル (またはタイトルバー) を別の位置にドラッグすることで、ツールバーグループの位置を変更することができます。ハンドルをダブルクリックすると、ツールバーがドッキングからフロート状態になり、タイトルバーをダブルクリックすることで、元の場所にドッキングされます。

ステータスバー

ステータスバーはアプリケーションウィンドウの下部に位置 (*図を参照*) しており、(i) ファイルのロードに関する情報や、(ii) マウスポインターがメニューコマンドや、コマンドショートカットにある時に、それらコマンドに関する情報を表示します。64ビット版のXMLSpyを使用している場合、XMLSpy の後に (x64) という表記が加わります。この表記は 32ビットバージョンにおいてはなされません。

1.2 アプリケーション環境

このセクションでは、アプリケーションの使用を始めるにあたり重要となる情報を記述します。このセクションを読み進めることで、XMLSpy の操作に慣れ親しみ、自信を持ってアプリケーションの使用を始めることができるようになります。設定やカスタマイズに関する重要な情報も含まれており、設定やカスタマイズのオプションにより設定可能な大まかな範囲や、それらの変更方法などついて知ることができます。

このセクションは以下のように構成されます：

- *設定、カスタマイズ* 重要な設定やカスタマイズを行う場所と、その変更方法について説明します。
- *チュートリアル、プロジェクト、サンプル*:アプリケーションパッケージに収められている、様々な非プログラムファイルの場所と、その注釈について記述します。
- *製品機能とドキュメンテーション、Altova 製品*: 製品の機能や、その他の形式で記述されたヘルプ、その他の Altova 製品に関する情報が記述されたAltova Web サイトへのリンクが掲載されます。

1.2.1 設定、カスタマイズ

In XMLSpy, には、複数の設定ならびにカスタマイズオプションが存在します。このセクションでは、これらのオプションの紹介を行ない、更にXMLSpy のメニューについても簡潔に説明します。このセクションは、以下のように構成されます：

- 設定
- カスタマイズ
- メニュー

設定

オプションダイアログのタブには、XMLSpy が持つ重要な設定が定義されています（以下のスクリーンショットはメニューコマンド「**ツール**」|「**オプション**」によりアクセスすることができます）利用可能なオプションがどのようなものか確認してください。

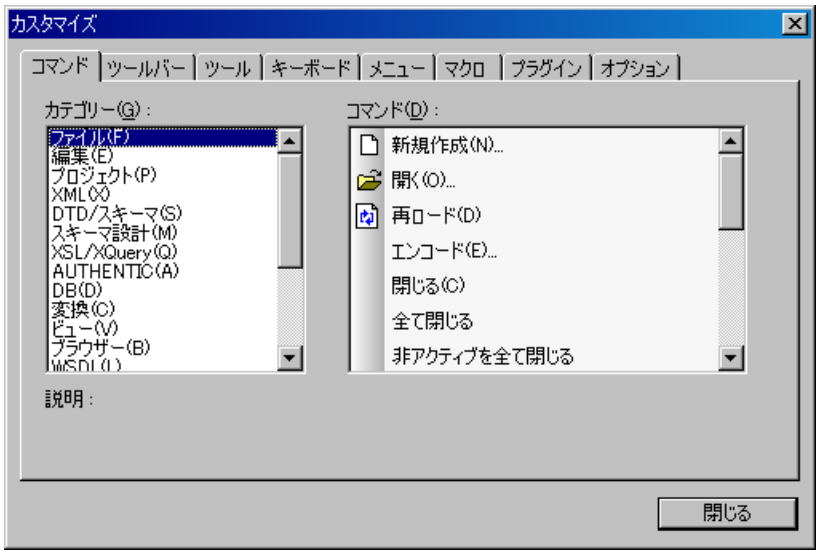
以下は最も重要な設定の要約です。詳細については、ユーザリファレンスにある*オプションダイアログ* の記述を参照ください。

- *ファイルタイプとデフォルトビュー*：ファイルタイプタブでは、XMLSpy が認識するファイルの種類を追加することができます。新たなファイル拡張子をリストの中に追加してください。各ファイルタイプに対して、そのファイルが準拠する仕様や、その種類（拡張子）のファイルを開く際に使用される XMLSpy のビュー（デフォルトビュー）を指定します。
- *ファイルの検証*：ファイルタブでは、ファイルを開く、保存する際に、ファイルの検証を自動的に行うか設定することができます。ファイルタイプタブでは、指定したファイルの種類の検証を無効にすることができます。

- *編集機能* :編集タブでは、入力ヘルパーの管理方法や、新規要素の生成、更に自動補完を有効にするなどの設定を行うことができます。ビュータブにおいても、その他の設定を個別に行うことができます。各ビューに対するフォントタブでは、これらビュー内部におけるノードの各種類のフォント設定を行うことができます。
- *XSLT とFO エンジン* : XSL タブでは、GUI から行われた変換で使われる外部 XSL エンジンを指定することができます。XMLSpy. 内部から FO 処理を行う際に使用される FO プロセッサーの場所も指定する必要があります。詳細については、XSLT 処理 のセクションを参照ください。
- *エンコーディング* : XML ならびに非 XML ファイルのエンコーディングはエンコードタブにより指定されます。

カスタマイズ

ユーザーはGUI の外観など、XMLSpyの様々な側面をカスタマイズすることができます。これらのカスタマイズオプションは、カスタマイズダイアログからアクセスすることができます (以下のスクリーンショットを参照) メニューコマンド **ツール | カスタマイズ** からアクセスすることができます)。



カスタマイズコマンドに関する詳細は **ユーザーレファレンス** のセクションを参照ください。

メニュー

メニューコマンドは、(i) ファイルの種類、(i) アクティブなビュー、(ii) 現在のカーソル位置または現在のドキュメントステータス という3つの要因により有効化、無効化されます。例えば:

- *ファイルの種類* :コマンド **DTD/ スキーマ | 他の DTD の取り込み**」は、アクティブなファイルが DTD でないと有効になりません。同様に、WSDL メニューのコマンドも、WSDL ファイルがアクティブでないと有効になりません。
- *アクティブなビュー* : **スキーマ設計** メニューにある殆どのコマンドは、アクティブなビューがスキーマビューでないと有効になりません。
- *現在のカーソル位置、ドキュメントステータス* :グリッドビューにて、子 ノードとして属性を加えるコマンド (**XML | 子を追加 | 属性**」) が有効になるかどうかは、グリッドビューにて選択されているアイテム 現在のカーソル位置) が要素かどうかに左右されます。XSLT ドキュメントがアクティブな状況でも、デバッグセッションが開始されない限り 現在のドキュメントステータス)、**デバッガーの停止** コマンドはアクティブになりません。

メモ:メニューのカスタマイズ (**ツール | カスタマイズ**) では、アイテムをGUI にドラッグすることで、コマンドの再構成を行うこ

とができます（メニューバー、ツールバーステータスバーを参照）。

1.2.2 チュートリアル、プロジェクト、サンプル

XMLSpy インストールパッケージには、チュートリアル、プロジェクト、そしてサンプルファイルが収められています。

チュートリアル、プロジェクト、サンプルファイルの場所

XMLSpy チュートリアル、プロジェクト、そしてサンプルファイルは以下のフォルダーにインストールされています：

```
C:\Documents and Settings\<username>\My Documents\
Altova\XMLSpy2017\Examples\
```

The `My Documents\Altova\XMLSpy2017` フォルダーは、PC 上で登録されたユーザーごとに、ユーザーの <username> フォルダー以下にインストールされます。従って、このインストールシステムでは、各ユーザーが、それぞれの場所に `Examples` フォルダーを持つことになります。

マスター XMLSpy フォルダーに関する注意

XMLSpy がインストールされると、以下の場所に、マスター `Altova\XMLSpy2017` フォルダーが次の場所に作成されます：

```
C:\Documents and Settings\All Users\Application Data\
```

このコンピューター上でユーザーが XMLSpy を初めて起動すると、XMLSpy は、マスターフォルダーのコピーを、ユーザーの <username>\ My Documents\ フォルダーに作成します。XMLSpy の他のユーザーが次回使用する場合、編集されたファイルがユーザーフォルダーにコピーされるため、チュートリアルやサンプルファイルを使用する際には、このマスターフォルダーを使用しないように注意する必要があります。

チュートリアル、プロジェクト、サンプルファイルの場所

チュートリアル、プロジェクト、サンプルファイルは、全て、`Examples` フォルダーに収められています。以下にその詳細を示します：

- XMLSpy *チュートリアル* ：Tutorial フォルダー。
- Authentic View *チュートリアル* ：Examples フォルダー。
- WSDL *チュートリアル* ： Examples フォルダー。
- *プロジェクトファイル* ： XMLSpy が開くサンプルプロジェクトは、Examples フォルダー内にある Examples.spp により定義されます。
- *サンプルファイル* ：Examples フォルダーならびに Examples のサブフォルダーに収められています。

1.2.3 XMLSpy 製品機能と Altova 製品

Altova Web サイト http://www.altova.com/jp/ には、以下にあるように XMLSpy に関連した様々な情報やリソースが含まれています。

XMLSpy 機能の一覧

Altova Web サイトには XMLSpy に含まれている最新機能の一覧や、XMLSpy のエディション（Enterprise、Professional、Standard）に応じてサポートされる機能の一覧などが掲載されています。ウェブサイトでは、以前のリリースから新たに追加された機能の一覧も参照することができます。

XMLSpy ヘルプ

このドキュメンテーションは、Altova により作成されたXMLSpy のヘルプです。ドキュメンテーションは、XMLSpy 内に収められており、**「ヘルプ」**メニューや **F1** によりアクセスすることができます。更に、Altova 製品のマニュアルは以下の形式でも入手することができます：

- オンライン HTML マニュアル Altova Web サイトのサポートページからアクセスすることができます。
- 印刷可能な PDF Altova Web サイトからダウンロードして、印刷することができます。
- 印刷物 Altova Web サイト上にあるリンクからお求め頂けます。

サポートオプション

ユーザマニュアル（本ドキュメンテーション）には記載されていない情報や、Altova 製品に関するお問い合わせは、Altova Web サイトの サポートセンター をご利用ください。以下のような機能がご利用になれます：

- よくある質問と回答ページへのリンク
- Altova 製品や、XML 全般に関するディスカッションフォーラム。
- オンラインサポートフォーム ：サポートパッケージをお持ちである場合、サポートリクエストを送信することができます。サポートリクエストは、サポートチームにより処理されます。

Altova 製品

Altova 製品のラインナップについては、Altova Web サイトを参照ください。

2 編集ビュー

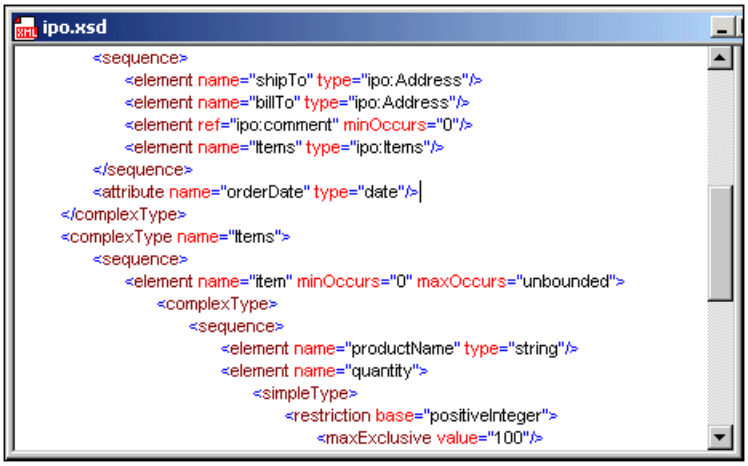
XMLSpy にはパワフルな編集ビューがあります。インテリジェント編集機能を備えたテキストビューに加えて、ドキュメントの編集を飛躍的に簡易化して表示するグラフィカルなビューも備わっています。XMLSpy にてアクティブになっているドキュメントの種類により、メインウィンドウには1つまたは複数の編集ビューが表示されます。例えば、Office Open XML ファイルや ZIP ファイルがアクティブになっていれば、メインウィンドウには1つだけの編集ビュー (アーカイブビュー) が表示されます。HTML ドキュメントがアクティブな場合、テキストビューとブラウザービューの2つの編集ビューが表示されます。XML ドキュメントがアクティブになれば、7つの編集ビュー (テキストビュー、グリッドビュー、スキーマビュー (XML スキーマドキュメントに対してのみ)、WSDL ビュー (WSDL ドキュメントに対してのみ)、XBRL ビュー、Authentic ビュー、そしてブラウザービュー) が表示されます。

このセクションでは、XMLSpy で使用できる様々なビューについて説明します：

- テキストビュー
- グリッドビュー
- スキーマビュー
- WSDL ビュー
- XBRL ビュー
- Authentic ビュー
- ブラウザービュー
- アーカイブビュー

2.1 テキストビュー

テキストビュー（下のスクリーンショットを参照）では、ドキュメントのテキスト（マークアップとコンテンツの両方）を直接入力することができます。XML ドキュメントでなくても、テキストファイル（例えば XQuery や HTML ドキュメント）なら何でテキストビューで編集することができます。数々の機能により素早く正確にドキュメントの入力作業を行うことができます。



このセクションでは、全ての種類のドキュメントで使用することのできる一般的なテキストビューの機能を紹介します。XML や XQuery、そして CSS といった特定の種類のドキュメントに対して割り当てられている機能については、それらのドキュメントタイプに関するセクションにて記述されます。例えば、テキストビューにおける XML 独自の機能については XML | テキストビューで XML を編集する セクションで記述されます。

一般的なテキストビューの機能は以下のように分類されています：

- テキストビューでのフォーマット ドキュメントのフォントプロパティ、インデンテーション、そしてワードラップの設定方法を説明します。
- ドキュメント表示 行番号、ブックマーク、ノードの展開/折りたたみ、そしてその他の表示に関わる機能についての情報を提供します。
- テキストビューでの編集 編集中に使用できる機能、特にインテリジェント編集機能について説明します。
- ドキュメント内の移動 テキストビュー内でドキュメントをナビゲートする様々な方法について説明しています。
- 検索および置換 テキストビューの検索および置換機能について説明しています。
- 入力ヘルパー コンテキストにより変化するデータ入力オプションを提供するウィンドウです。例えば、指定された位置で入力することのできる要素または属性が入力ヘルパーに表示され、ダブルクリックすることによりオプションを挿入することができます。

テキストビューに切り替える
ドキュメントのテキストビューを開くには、ドキュメントウィンドウの下部にある **「テキスト」** ボタンをクリックするか、**「ビュー | テキストビュー」** を選択します。

2.1.1 テキストビューでのフォーマット

テキストビューでは様々なフォーマットオプションが提供されます。

フォント

フォントファミリー、フォントサイズ、フォントスタイル、そしてテキスト背景色は以下のグループのドキュメントに対して個別に設定することができます：(i) 一般的なXMLドキュメント（HTMLを含む）；(ii) XQueryドキュメント；(iii) CSSドキュメント。

異なる意味を持つドキュメント内のテキストアイテムには別々の色を与えることができます。例えば、要素の名前、属性の名前、そして要素の内容を別々に色付けすることができます。テキストアイテムに対して異なる色付けをした場合、構文のカラー強調表示機能が有効になります。テキストフォントは「オプションダイアログのテキストフォント」タブにより設定することができ、その方法は「ユーザーリファレンス」「オプション」「テキストフォント」セクションに記述されています。

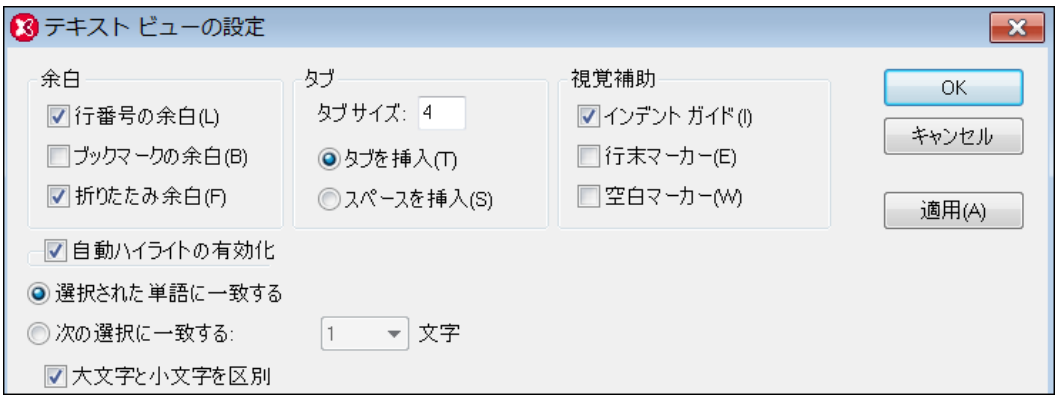
インデント

整形式のXMLドキュメントはインデント付きで表示することができます。つまりドキュメントの階層的な構造が新しい行やインデントにより表示されることになります。（下のスクリーンショットを参照）

```
<Office>
    <Name>Nanonull, Inc.</Name>
    <Desc>
    <Location>US</Location>
    <Address>
        <ipo:street>119 Oakstreet, Suite 4876</ipo:street>
        <ipo:city>Vereno</ipo:city>
        <ipo:state>DC</ipo:state>
        <ipo:zip>29213</ipo:zip>
    </Address>
    <Phone>+1 (321) 555 5155 0</Phone>
    <Fax>+1 (321) 555 5155 4</Fax>
    <EMail>office@nanonull.com</EMail>
    <Department>
        <Name>Administration</Name>
        <Person>
            <First>Vernon</First>
            <Last>Callaby</Last>
            <Title>Office Manager</Title>
            <PhoneExt>582</PhoneExt>
            <EMail>v.callaby@nanonull.com</EMail>
            <Shares>1500</Shares>
            <LeaveTotal>25</LeaveTotal>
            <LeaveUsed>4</LeaveUsed>
            <LeaveLeft>21</LeaveLeft>
        </Person>
```

この方法でドキュメントを表示するには、以下のような操作を行います：

1. オプションダイアログのビュータブにて、**「インデントを使用」**にチェックを入れます。この操作により階層的な構造を表示する為にインデントが使用されます。より深い階層構造を表示するのに親の要素よりも深いインデントが使用されます。**「インデントを使用」**オプションがチェックされていない場合、ドキュメントの全ての行がインデント0からスタートします。
2. テキストビューの設定ダイアログ（下のスクリーンショットを参照）にて、**「タブを挿入」**か**「スペースを挿入」**を選択します。この操作でインデントにタブまたはタブかスペースが使われるのか決定されます。スペースが選択された場合、構造的な階層のより深い箇所に、テキストビュー設定のタブサイズにより指定された数だけスペースが挿入されインデントを形成します。



3. **編集 | XML テキストを整形して出力** コマンドを選択するか、ツールバーの整形アイコンをクリックします。この操作によりドキュメントのテキストがインデントされた状態で表示され、階層構造をより簡単に閲覧することができます。インデンテーションはテキストビュー設定ダイアログのタブペインにより決定されます。必要ない余白や冗長な空白は削除されます。

注意： このようなインデントはドキュメントを保存するときやビューを切り替えるときにも自動的に行われます。ドキュメントが整形式でない場合、エラーメッセージが表示されます。エラーを修正し、再度テキストを整形して出力します。インデントの階層はインデントガイドにより垂直の点線により表示され（上のスクリーンショットを参照）、**「テキストビュー設定」**により有効/無効の指定を行うことができます。

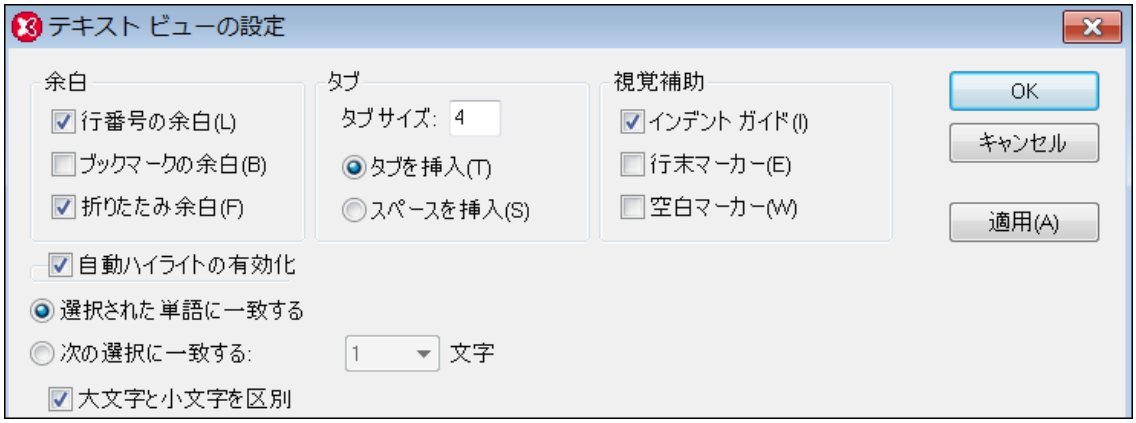
フォーマットにタブとスペースを使用する
インデントオプションが有効にはならないXML ドキュメントではないテキストに対してもタブとスペースによるフォーマットを適用することができます。**「Return」** キーを押すか **「Shift」+ 「Return」** キーを押すことで、カーソルは前の行の開始位置に対応する位置に移動します。

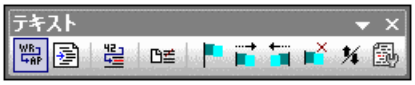
ワードラップ
表示 | ワードラップ コマンドを有効にすることで、メインウィンドウの幅よりも長い行は折り返して表示されます。対応するアイコンはテキストツールバーにもあります。

2.1.2 ドキュメントの表示

テキストビューでは大規模なテキストの閲覧と編集を簡単にする視覚機能を使用することができます。とても便利な機能として：(i) 行番号、(ii) ブックマーク、(iii) ソースの折りたたみ（ノードごとに展開したり折りたたんだりする機能）、(iv) インデントガイド、そして (v) 行末マーカーと空白マーカーがあります。これらのコマンドはテキストビュー設定ダイアログ（1番目のスクリーンショット）とテキストツールバー（2番目のスクリーンショットを参照）にて使用することができます。

テキストビュー設定ダイアログには **「表示 | テキストビューの設定」** コマンドまたは、テキストツールバーの **「テキストビューの設定」** ボタンからアクセスすることができます。テキストビュー設定ダイアログの設定内容はアクティブなドキュメントだけでなく、アプリケーション全体に対して適用されます。





その他の便利な機能として、「ズーム」と「サジェートと検索」機能があります。

行番号

行番号は行番号余白に表示され（下のスクリーンショットを参照）、テキストビュー設定ダイアログで有効/無効にすることができます（上のスクリーンショットを参照）。テキストのセクションが折りたたまれているときには、折りたたまれている行の番号も同様に隠されます。関連するコマンドに「行/文字へ移動」コマンドがあります。

ブックマーク

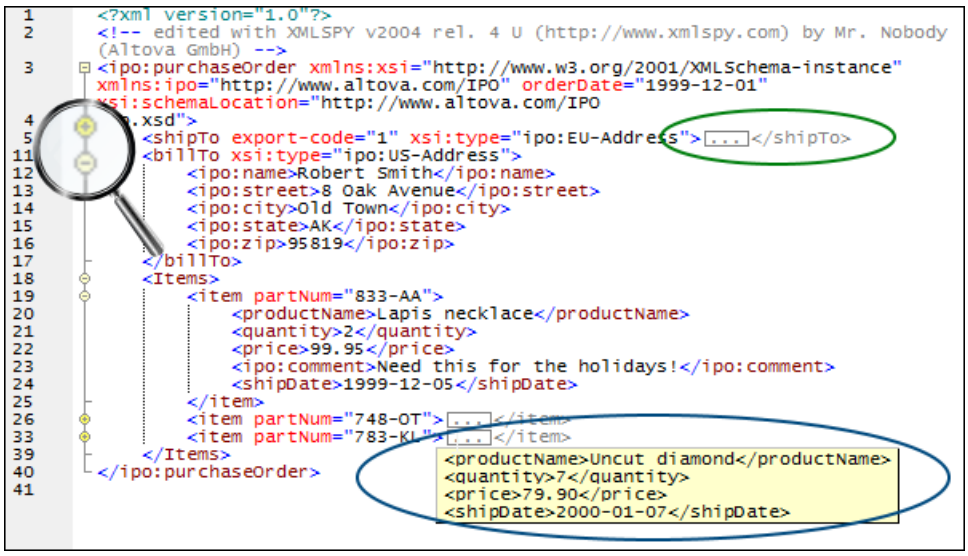
ドキュメントの行をブックマークして素早い参照やアクセスを行うことができます。ブックマークの余白が有効化されていれば、ブックマーク余白にブックマークが表示されます。そうでない場合、ブックマークされた行はシアン色でハイライトされます。

ブックマークの余白はテキストビュー設定ダイアログにて有効/無効化することができます（上のスクリーンショット参照）。

「編集」メニューとテキストツールバーのコマンドを使用することでブックマークの編集やブックマークへの移動を行うことができます。**「編集 | ブックマークの挿入/削除」**コマンドによりブックマークを挿入して、ドキュメント内の行をマークすることができます。ブックマークを削除するには、該当するブックマークがある行まで移動し、**「編集 | ブックマークの挿入/削除」**コマンドを選択することで行うことができます。ドキュメント内のブックマークを移動するには、**「編集 | 次のブックマークへ移動」**と**「編集 | 前のブックマークへ移動」**コマンドを使用します。これらのブックマークコマンドはテキストツールバーからも使用することができます（上のスクリーンショット参照）。

ソースの折りたたみ

テキストの折りたたみ（またはソースのおりたたみ）はXQuery中かっこ、XQueryコメント、XML要素、XMLコメントにより有効化され、これらのノードを展開または折りたたむ機能を意味します。このようなノードは、+/-サインによりソース折りたたみ余白内で表示されます（下のスクリーンショット参照）。余白はテキストビュー設定ダイアログにより切り替えのオンとオフを行うことができます（上のスクリーンショット参照）。下のスクリーンショット内では、3つのノードが折りたたまれています：`shipTo`要素2つの`item`要素。ノードが折りたたまれると、視覚的に省略記号により表示されます（下のスクリーンショットでは緑の枠でかこまれています）。内容がポップアップでの表示に大きすぎる場合、ポップアップのしたの部分に省略記号で表示されます。



テキストツールバーの **全て展開 折りたたみ** アイコン **全ての**ノードを展開、またはトップレベルドキュメント要素に折りたたむことができます。

ノードの切り替えアイコン (+ / − アイコン) をクリックする際に以下のオプションが利用可能になります：

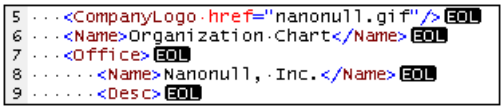
クリック[−]	ノードを折りたたみを行ないます。
クリック[+]	クリックすることでノードが展開され、そのノードの子孫は、折りたたみが行われる前の状態で表示されます。
Shift+クリック[−]	全ての子孫ノードを子ノードのレベルまで折りたたみます。クリックされたノードそのものは開かれ、折りたたまれた全ての子ノードが表示されます。この操作は、枝の末端にあるノードを除き、全てのノードに対して適用することができます。
Ctrl+クリック[+]	ノードまたはその子孫が折りたたまれている場合、**「Ctrl」+クリック**により、折りたたまれている全ての子孫ノードを展開することができます。つまり、この操作でクリックされたノードと、その子孫ノード全てが展開されます。

インデントガイド

インデントガイドとは垂直に表示される点線のことで、インデントのレベルを表示します（上のスクリーンショットを参照）。テキストビュー設定ダイアログにて有効/無効化を切り替えることができます。

行末マーカー、空白マーカー

改行マーカーと空白マーカーはテキストビュー設定ダイアログにて有効/無効化を切り開けることができます。下のスクリーンショットではこれらのマーカーがドキュメント内でどのように表示されるかを表しています（各ドットはスペースを表しています）。



ズームインとアウト

「Ctrl」キーを押しながらスクロール（マウスのホイール）を使用することでテキストビューの拡大と縮小を行うことができます。この機能を使うことで、テキストビューのテキストサイズを調整することができます。フォントサイズを変更したい場合は、オプションダイアログにて行うことができます。

2.1.3 テキストビューでの編集

以下のテキスト編集機能は一般的な全ての種類のドキュメントに対して使用できるテキストビュー内の機能です。これらの機能は（**編集**」メニュー以下にある）**切り取り**、**コピー**、**貼り付け**、**削除**、そして**すべて選択**の様な編集の為の一般的な機能とは異なるものです。

- 構文の色分け
- インテリジェント編集
- 自動補完
- 開始タグと終了タグのマッチ
- 同じ階層で要素を移動させる
- 要素全体を選択し親要素に移動する
- ドラッグアンドドロップとコンテキストメニュー
- 制限無しに元の状態に戻す
- スペルチェック

（XML や XQuery などの）ドキュメントの種類によっては、更に独自の機能が利用可能です。これらの機能については、それぞれ対応するドキュメントの種類に関するセクションで記述されています。

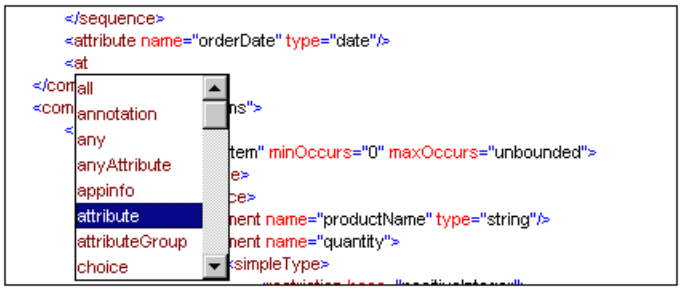
メモ： 大きなファイルに関しては、速いロードと編集のため自動補完と入力ヘルパーが無効化されます。ファイルのサイズの敷居はユーザーにより指定することができます。更に詳しい情報に関しては、「オプション| 編集」を参照してください。

構文の色分け

構文の色分けはテキストが持つ意味によって変化します。例えば、XML ドキュメントの場合、XML ノードが要素、属性、コンテンツ、CDATA セクション、コメント、または XML 処理命令であるかによって、ノード名に そして場合によってはノードのコンテンツも）異なる色が付けられます。ドキュメントは 3 つのグループに分けられています：(i) 一般的な XML（HTML を含む）；(i) XQuery ；そして (ii) CSS です。それぞれのグループの 構文の色を含む）テキストプロパティはオプションダイアログ（「**ツール| オプション**」）のテキストフォントタブにより設定できます。
。これは視覚的な編集のヘルプとしての役割を果たします。

インテリジェントな編集

スキーマをベースにした XML ドキュメントを扱っている場合、XMLSpy のテキストビューで様々なインテリジェント編集機能を使用することができます。これらの機能を使用することで、現在編集しているコンテンツモデルに従った正しい要素、属性、または属性の値を素早く入力することができます。例えば、ある要素の開始タグを入力しようとしているときに、スキーマがその位置え許している要素の一覧がポップアップで表示されます（下のスクリーンショットを参照）。要素名を選択し、「**Enter**」を押すことで開始タグにその要素名が入力されます。また、開始タグが作成された後、終了タグが自動的に追加されます（*下の自動補完below を参照してください*）。



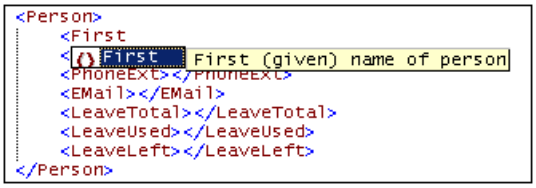
ポップアップウィンドウは以下の場合にも表示されます：

- カーソルが要素タグの間にあり、その要素に属性が定義されているときにスペースバーを押すと、使用可能な属性全てを含んだポップアップが表示されます。
- カーソルが列挙の値を持つ属性の二重引用符（"）の中にある場合、表示されるポップアップには列挙値が表示されます。
- 終了タグの開始を表す </を入力したとき、対応する要素の名前がポップアップに表示されます。
- 内容を持たない要素を空要素で記述しようとするか、2つのタグに囲まれた内容の無い要素を空要素に変換したい場合、要素の開始タグの最後にスラッシュを入力します（例：<element/>）。空要素が作成されます。）

自動補完

テキストビューを使った編集の結果、整形式ではないXMLやその他の（HTMLの様な）マークアップドキュメントを作成してしまうことがあります。例えば、終了タグが漏れていたり、スペリングに間違いがあったり、構造的なミスマッチがあったりした場合です。XMLSpyは自動的に要素の開始ならびに終了タグを完成し、要素の名前が入力されると同時に必須の属性も挿入されます。カーソルも自動的に開始と終了タグの中間に移動し、すぐに子要素やコンテンツを入力することができるようになっています：`<img src="" alt=""> </img>`

%XMLSPY%> は整形式の為にXMLのルールを使用し、自動補完の為に妥当性を使用します。ドキュメントの構造に関する情報はそのドキュメントがベースにしているスキーマから取得されます（HTMLやXSLTといったよく使用されるスキーマの情報は既にXMLSpyに組み込まれています）。自動補完はドキュメントの構造情報だけでなく、ドキュメント内に保存された値も参照します。例えば、XMLスキーマの列挙や注釈情報は自動補完により頻繁に参照されます。あるノードに対してスキーマ内で列挙が記述されていれば、そのノードが編集されるときにこれらの列挙情報が自動補完により表示されます。同様に、あるノードに対してスキーマ内に注釈が存在する場合、ノードがドキュメント内で入力されているときに、その注釈も表示されます（下のスクリーンショットを参照；*First (given) name of person* が`First`要素に対するスキーマの注釈になります）。



自動補完はオプションダイアログの編集タブ（**「ツール | オプション | 編集」**）にて有効/無効化することができます。

開始と終了タグの一致

カーソルがマークアップ要素の開始もしくは終了タグの中をポイントしているとき、**「Ctrl」+E**を押すことで選択を他のペアのメンバーに移動することができます。**「Ctrl」+E**を繰り返し押すことで、開始タグと終了タグを切り替えることができます。この機能を使って、簡単にXML要素の開始タグと終了タグを見つけることができます。これは、XML要素のタグを開始と終

了を検索するために役にたちます。更に、開始タグで、入力を始めると、開始と終了タグの名前が一致するか否かにより要素の名前は異なる色でハイライトされます

互いに相対する要素を移動する
カーソルが要素の中に位置しているとき、Alt＋上矢印 またはをAlt＋下矢印 押すことで、選択されている要素をその階層内で上か下に移動させることができます。

要素全体を選択し、親要素に移動する
カーソルが要素内に配置されている状態で 「Ctrl」 + 「Shift」 +E を押下することで、要素全体を選択することができます。**Ctrl+Alt+E** をクリックすると、親要素の開始タグがハイライトされます。両方のショートカットにより、ドキュメントの構成に対して相対的なカーソルの位置を素早く検索することができます。

ドラッグアンドドロップとコンテキストメニュー
ドラッグアンドドロップやテキストのブロックを他の場所に移動させたり、右クリックにより表示されるコンテキストメニューを使用して、切り取りやコピー、ペースト、削除、メールで送信、そして行 文字に移動などの) よく使われるコマンドにアクセスすることでもテキストの移動を行うことができます。

制限無しに元の状態に戻す
XMLSpy は全ての編集オペレーションに対して制限無しに元に戻す およびやり直しを提供します。

スペルチェック
テキストビューでは、内蔵の言語辞書を使用することでドキュメントのスペルチェックを行うことができます。ユーザー辞書を作成することで、言語処理に含まれていない単語を加えることができます。詳細については、スペリングならびにスペルチェッカーのオプションコマンドを参照ください。

2.1.4 ドキュメント内の移動

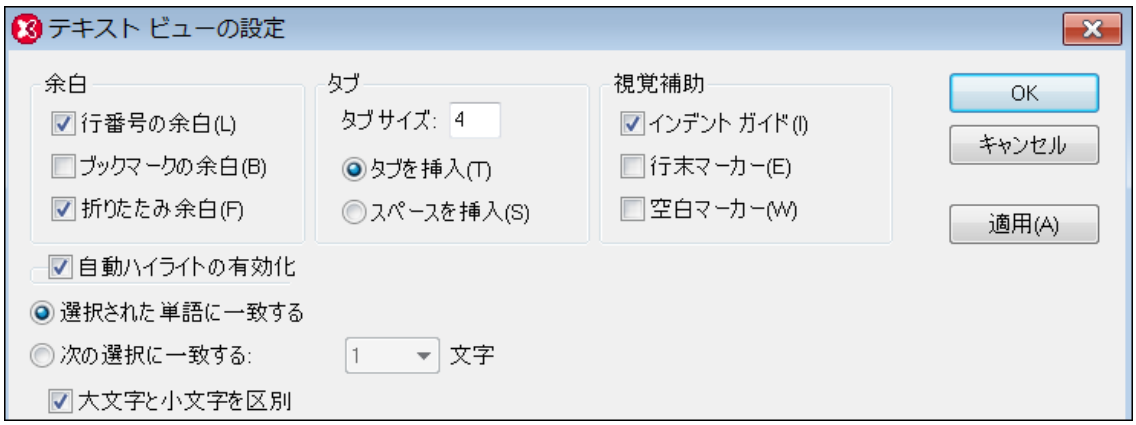
テキストビューでドキュメント内を移動するときには、以下の機能を使用することができます：

- テキストのハイライト により、選択したテキスト文字列または文字に一致する全てのマッチを検索することが可能になります。それぞれの一致は、スクロールバーに示されており、全ての一致に簡単に移動することができます。
- スクロールバー内のドキュメントの概要 により、ドキュメント内でのカーソルとテキストの相対的な位置を表示します。
- ライン文字に移動 により、指定したラインと文字に直接移動することができます。

テキストのハイライト
テキストビュー設定ダイアログ (**表示** | **テキストビュー設定**) 内でテキストのハイライトが有効化されていると、選択されたテキストのドキュメント内の全ての一致がハイライトされます。選択は、薄い青色でハイライトされ、一致する箇所は薄いオレンジ色でハイライトされます (*下のスクリーンショット参照*)。選択と、その一致する箇所は、スクロールバー上で四角の灰色のマーカーで表示されます。スクロールバー内で現在のカーソルの位置は、青いカーソルマーカーで示されています。

```
<expense-report xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ExpReport.xsd" currency="USD" detailed="false"
total-sum="556.9">
    <Person>
        <First>Fred</First>
        <Last>Landis</Last>
        <Title>Project Manager</Title>
        <Phone>123-456-7890</Phone>
        <Email>f.landis@nanonull.com</Email>
    </Person>
    <expense-item type="Lodging" expto="Sales">
        <Date>2003-01-01</Date>
        <expense>122.11</expense>
    </expense-item>
    <expense-item type="Lodging" expto="Development">
        <Date>2003-01-02</Date>
        <expense>122.12</expense>
        <description>Played penny arcade</description>
    </expense-item>
    <expense-item type="Lodging" expto="Marketing">
        <Date>2003-01-02</Date>
        <expense>299.45</expense>
        <description>Treated Clients</description>
    </expense-item>
    <expense-item type="Entertainment" expto="Development">
        <Date>2003-01-02</Date>
        <expense>13.22</expense>
        <Misc misctype="TeamBuilding"/>
        <description>Bought signed XMLSpy Handbook</description>
    </expense-item>
</expense-report>
```

テキストのハイライトをオンに切り替えるには、テキストビュー設定ダイアログ（**表示**|**テキストビューの設定**）内の*自動ハイライト*の有効化を選択します。*下のスクリーンショット*)。選択は、文字全体、または、文字の固定された数値で定義することができます。また、キャストを考慮するかを選択することもできます。



以下の点に留意してください：

- 文字の部分では、選択内で一致する必要がある文字の最小数を指定することができます。例えば、2つ以上の文字が一致するかを選択することができます。この場合、一文字の選択は、一致とはみなされず、2つ以上の文字から構成される選択のみがマッチされます。ですから、この場合、`t` を選択すると、一致する項目は表示されません。`ty` を選択する場合、`ty` に一致するマッチが表示されます。`typ` を選択すると、`typ` に一致するすべてのマッチが表示されます。
- 文字の検索の場合、文字を区別するために次の条件が考慮されます：角かっこのない要素名、要素タグの角かっこ、属性名、引用符なしの属性の値。

メモ： 要素開始タグと要素完了タグは、自動ハイライトの有効化に影響されない個別の機能です。オプションダイアログのテキストフォントタブ（**ツール**|**オプション**）内で、要素の開始タグと完了タグの一致に色を設定することができます。

スクロールバー内のドキュメントの概要

スクロールバーには以下の機能が搭載されています：

- 次の項目のそれぞれのサイズに対して相対的に表示されます (*下のスクリーンショット参照*): (i) ドキュメント全体 (スクロールバー); (i) ウィンドウ (縮小表示) 内の現在選択されているドキュメントセグメント; (ii) 現在選択されているテキスト (青いバー) および (v) 現在のカーソルの場所 (カーソルマーカー)。
- ドキュメント内を以下の方法で移動することができます：(i) スクロールバーの縮小表示を上下にドラッグする、または (ii) (下のスクリーンショット内で緑の枠で囲まれている) **「上にスクロール」**と**「下にスクロール」**矢印をクリックします。

以下の点に留意してください：

- ドキュメント全体の長さに対応したスクロールバーの長さ
- ウィンドウにドキュメントの一部のみがフィットする場合、ウィンドウに表示される部分は、スクロールバーの縮小表示に対応します (*上のスクリーンショット参照*)。ドキュメントを閲覧するために、ドキュメントの縮小表示を上下にドラッグします。
- 現在のテキストの選択は、スクロールバー上の青いバーで表示されています。スクロールバーのサイズに対応する青いバーのサイズは、ドキュメント全体のサイズに対するテキストの選択のサイズに相対します。テキストの選択が一行を超えない場合、青いバーは表示されません。
- カーソルの位置は、濃い青のカーソルマーカーにより表示されています。スクロールバー内のカーソルマーカーの相対的な位置は、ドキュメント内のカーソルの位置に相対します。

ライン/文字に移動

「表示」メニューおよびテキストツールバー内のこのコマンドにより、ドキュメントテキスト内の特定のラインまたは文字に移動することができます。

2.1.5 検索および置換

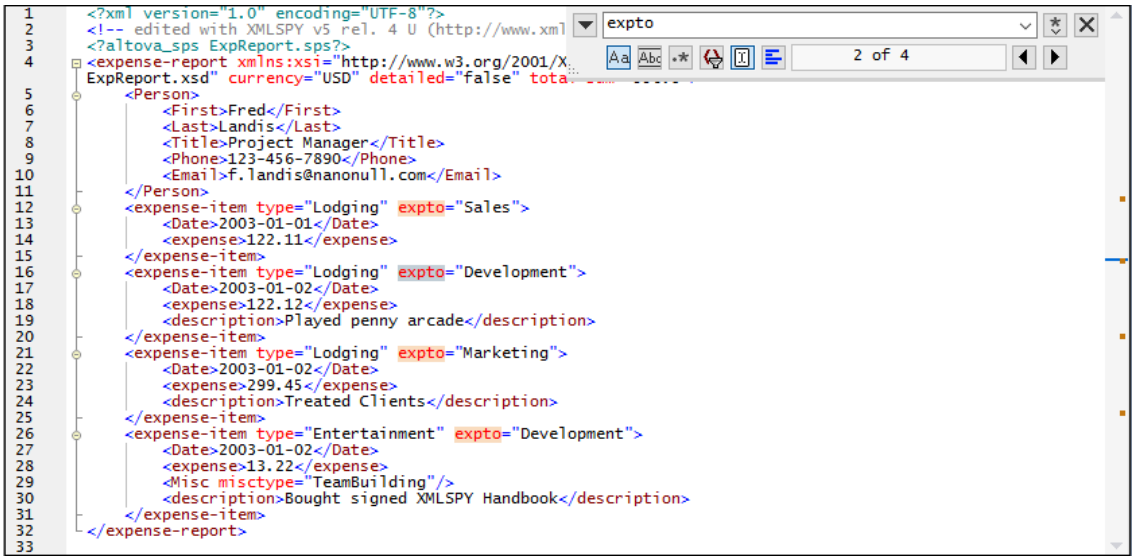
このセクション:

- ドキュメント内のテキストの検索
- 検索オプション
- テキストの置換え
- 正規表現の使用
- 正規表現のサンプル

ドキュメント内でのテキスト検索

Ctrl+F を押して、(または、メニューコマンド **編集 | 検索** を選択して)検索ダイアログを表示します (*下のスクリーンショット、右上*)。ドキュメント全体、または、選択されたテキストの範囲に対して、ダイアログに入力されたテキストが検索されます。

- 検索する文字列を入力、または*コンボボックス*を使用して文字列の最後10列の1つの文字列を選択します。
- 検索する文字列を入力、または、選択すると、全ての一致がハイライトされ、一致する箇所がベージュのマーカーで*スクロールバー*内で表示されます (*スクリーンショット参照*)。
- 現在選択されている一致は、他の一致と異なる色でハイライトされ、一致する箇所は濃紺のカーソルマーカーで表示されています。
- 一致の総数は、検索用語のフィールドの下に、現在選択されている一致のインデックス上の位置と共に表示されています (*下のスクリーンショット参照*)。例えば、`2 of 4` は、4つの一致の2番目が現在選択されていることを示しています。
- 1つの一致から次の一致に移動することができます。前後に移動するには、右下に有る **前へ** (Shift+F3) と **次へ** (F3) ボタンそれぞれを選択します。



- 検索および置換 モード間を切り替えるには、左上の下矢印をクリックします。
- ダイアログを閉じるには、右上の **閉じる** ボタンをクリックします。

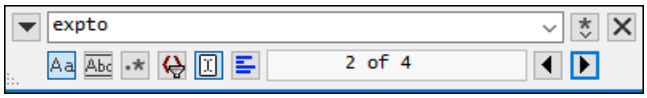
以下の点に留意してください:

- 検索ダイアログにはモードが存在しません。これは、テキストビューを使用し続けている間、開いておくことができることを意味します。

- ダイアログボックスを開く前にテキストが選択されている場合、選択されたテキストは、自動的に検索フィールドとして既知の検索用語フィールドに挿入されます。
- 範囲内で検索する場合、以下を行います：(i) 選択範囲をマークします。(i) *選択範囲内での検索*オプションをオンにして、選択をロックします。(ii) 検索用語を入力します。他の選択を検索するには、現在の選択を、*選択範囲内での検索*オプションをオフにして、ロックを解除してください。新しい選択を決定し、*選択範囲内での検索*オプションをオンにします。
- 検索ダイアログが閉じられた後、現在の検索を**「F3」**を押して繰り返すことができます。または、**「Shift+F3」**を押して後ろ向きに検索することができます。検索ダイアログが表示されます。

検索オプション

検索用語フィールドの下のボタンを使用して、検索の条件を指定することができます（*下のスクリーンショット参照*）。オプションがオンに切り替えられていると、ボタンの色が青に変更されます（*下のスクリーンショット内の最初の（キャスティング）オプションを参照してください*）。



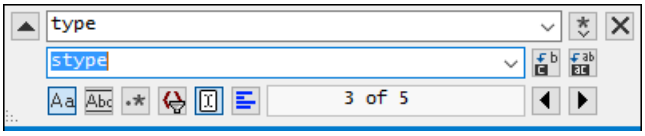
以下のオプションから選択することができます：

- *大文字と小文字を区別する*: オンに切り替える場合は大文字と小文字が区別されます（`Address` は、`address` とは異なります）。
- *単語単位*: テキスト内の単語単位が一致するものをのみを表示する。例えば、単語単位がオンに切り替えられており、入力文字列が`fit` の場合、`fit` のみが検索文字列に一致します。例えば、`fitness` 内の`fit` は、一致ではありません。
- *正規表現*: オンに切り替えられていると、検索用語は正規表現として読み込まれます。どのように正規表現が使用されるかに関する説明は、下の*正規表現*を参照してください。
- *フィルターの結果*：検索が実行される1つまたは1つ以上のドキュメントコンポーネントを選択します。
  - *アンカーの検索*：検索用語が入力されると、ドキュメント内の一致がハイライトされ、一致の内の1つが現在の選択としてマークされます。アンカーの検索は最初の選択がカーソルの位置に相対するかを決定します。アンカーの検索がオンになっていると、最初の現在選択されている一致は、カーソルの位置の次の一致を指します。アンカーの検索がオフになっている場合、最初の現在選択されている一致は、ドキュメントの始めから数えた一番目の一致です。
- *選択範囲内での検索*：オンに切り替えられると、現在の選択範囲がロックされ、検索が選択範囲内に制限されます。それ以外の場合、ドキュメント全体が検索されます。テキストの新しい範囲を選択する前に、現在選択されている検索範囲を検索範囲内オプションをオフにしてロックを解除してください。

テキストの置換え

テキスト文字列を置き換えるには、以下を行います：

1. **「Ctrl+H」**を押して（または、メニューコマンド**「編集 | 置換」**を選択して）置換えダイアログを開きます（*下のスクリーンショット*）。（または、ダイアログの左上の下矢印ボタンをクリックして検索/置換ダイアログの置換モードを切り替えることができます。）



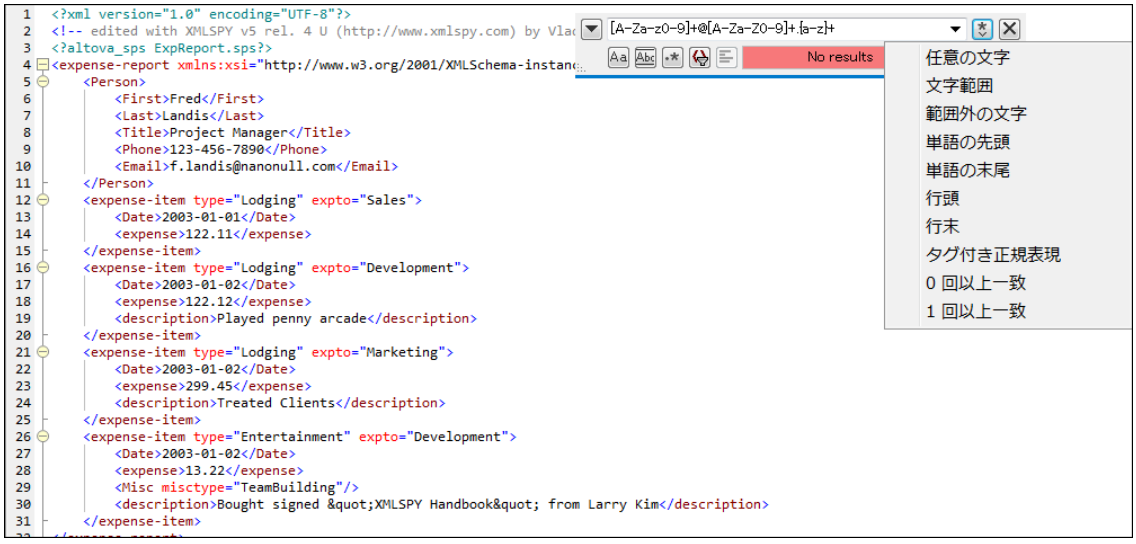
2. 検索フィールド内で置き換えられる文字列を入力し、フィールドの置換内に新規の文字列を入力します。置き換えられる一致するテキストの数と現在選択されているマッチは、フィールドの置換えの下に表示されます。また、一致の場所は縦スクロールバー上でページュのマーカーで表示されています。また、一致する箇所は水平スクロールバー内でページュのマーカーで表示されています。（詳細に関しては、上の *ドキュメント内でのテキストの検索* を参照してください）。例えば、上のスクリーンショットは、文字列 `type` に対して5つの一致が存在することを示しており、3番目の一致が現在選択されています。
3. **「次を置き換える」** と **「全てを置き換える」** ボタンは置換フィールドの右側にあります。 **「次を置き換える」** をクリックすると、以下の1つが起こります：(i) カーソルが一致、または、一致内に存在する場合、置換えが実行されます。 (i) カーソルが一致の外に存在する場合、次の一致に移動します。 **「次を置き換える」** をクリックして、この一致を置き換え、 **「全てを置き換える」** をクリックして全ての一致を置き換えます。

以下の点に留意してください：

- ドキュメント全体ではなく、選択範囲内でテキストを置き換えるには、以下を行います：(i) 選択をマークする。(i) *選択範囲内で検索* オプションをオンに切り替えて、選択をロックする。 (ii) 検索と置き換えテキスト文字列を入力します。 (ii) **「次を置き換える」** または **「全てを置き換える」** を必要に応じてクリックします。他の選択範囲内のテキストを置き換えるには、現在の選択範囲を*選択範囲内で検索*オプションをオフに切り替えてロック解除し、新しい選択を指定し、*選択範囲内で検索*オプションをオンにして新しい選択をロックします。
- 置換えのアクションを元に戻す場合は、 「Ctrl+Z」または、 **「編集 | 元に戻す」** を選択します。

正規表現の使用

正規表現（regex）を使用して、テキスト文字列を検索することができます。これを行うには、*正規表現*のオプションをオンに切り替えます（上の *検索オプション* を参照してください）。これにより、検索用語フィールド内のテキストが正規表現として評価されます。正規表現の構築へのヘルプは、検索用語フィールドの右にある **「正規表現 ビルダー」** ボタンをクリックしてください（*下のスクリーンショット参照*）。ビルダー内のアイテムをクリックして、正規表現メタ文字を対応する検索用語フィールドに入力します。下のスクリーンショットは、電子メールアドレスを検索するための簡単な正規表現を表示しています。メタ文字に関する簡単な説明は、下の*正規表現 メタ文字* セクションを参照してください。



正規表現メタ文字

正規表現 メタ文字のリストは以下の通りです。

.	任意の文字を一致する。これは単一の文字のプレースホルダーです。
\(	一致をタグした範囲の開始をマークする。
\)	一致をタグした範囲の終了をマークする。
\(abc**\)**	**\(** と **\)** メタ文字は、タグされた式の開始と終了をマークします。一致する箇所を、後で参照する(バックレファレンス)目的のためにタグ(「記録」)する場合、役に立つ可能性があります。9つまでの式をタグ付けすることができます(検索、または、置換フィールド内で後に参照することができます)。 例えば、**\(the\) \1** は文字列 the the に一致します。この式は、以下のように説明することができます:前に一致しているタグ付けされた箇所へバックレファレンスが後に続く スペース文字が後に続く文字列「the」に一致させる(そしてタグ付けされた箇所として記録する)。
\n	n が1 から9 までの整数の値を持つことができる**変数** の場合。式は、置き換える際にタグ付けされた1番目から9番目までの箇所を参照することができます。例えば、検索文字列が Fred**\([1-9]\)**XXX で、置換文字列が Sam**\1**YYY の場合、検索文字列内に、明示的に番号 1とインデックスされている1つのタグされた式が存在することを意味し、また、置換文字列内では、タグ付けされた式は、**\1** と参照されています。検索と置換式が Fred2XXX に対して適用されている場合、Sam2YYY が生成されます。
\<	単語の先頭に一致。
\>	単語の末尾に一致。
\x	このコンテキスト以外では特別な意味を持つ 文字 x が使用できるようになります。例えば、\[は、文字セットの始まりではなく、[として解釈されます。
[...]	文字のセットを示します。例えば、[abc] は、a、b または c のいずれかを意味します。範囲も使用することができます:例えば、[a-z] をすべての小文字のために使用できます。
[^...]	セット内の文字の補完です。例えば、[^A-Za-z] は、アルファベット文字を除く全ての文字を意味します。
^	行頭に一致。(セットの内部で使用される以外の場合。上を参照してください)。
$	行末に一致。例 :A+$ が1つ以上の行末の A を検索する場合。
*	0 回以上一致。例えば、Sa*m は、Sm、Sam、Saam、Saaam に一致するなど。
+	1 回以上一致。例えば、Sa+m は、Sam、Saam、Saaam に一致するなど。

特別文字の表記
以下の式に注意してください。

\r	キャリッジリターン (CR)。CR (**\r**) または LF (**\n**) を使用して新規のラインを検索または作成します。
\n	改行 (LF)。CR (**\r**) または LF (**\n**) を使用して新規のラインを検索または作成します。
\t	タブ文字
****	正規表現内に表示される文字をエスケープするために使用してください。例 :**\\\n**

正規表現のサンプル

このサンプルは、正規表現を使用してテキストを検索し置き換える方法について説明しています。多くの場合、テキストの検

索と置換えは複雑ではなく、正規表現を必要としません。しかしながら、通常の検索と置換え操作ではテキストを操作できないインスタンスが存在する場合があります。例えば、数千行からなるXML ファイル内の1つのオペレーション内の特定の要素を、囲まれたコンテンツに影響を与えずに、名前を書き換える必要がある場合が挙げられます。もう1つの例は、要素の複数の属性の順序を変更する場合です。この場合は、手動で行う際に必要とされる手間を省くため、正規表現の使用が役に立ちます。

サンプル 1：要素の名前変更

サンプルXML コードリストには、ブックのリストが含まれています。このサンプルの目的を、の各ブックの `<Category>` 要素を `<Genre>` と置き換えることとします。この目的を達成するために正規表現を使用します。

```
<?xml version="1.0" encoding="UTF-8"?>
<books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="books.xsd">
   <book id="1">
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <category>Fiction</category>
      <year>1876</year>
   </book>
   <book id="2">
      <author>Franz Kafka</author>
      <title>The Metamorphosis</title>
      <category>Fiction</category>
      <year>1912</year>
   </book>
   <book id="3">
      <author>Herman Melville</author>
      <title>Moby Dick</title>
      <category>Fiction</category>
      <year>1851</year>
   </book>
</books>
```

必要条件を満たすには、以下のステップを踏んでください：

1. 検索と置換ダイアログボックスを開くには、**Ctrl+H** を押します。
2. **正規表現の使用** .* をクリックします。
3. 検索フィールドに次のテキストを入力します：`<category>(.+)</category>`。この正規表現は、すべての `category` 要素に一致し、ハイライトされます。

(既知ではない)各要素の内部のテキストに一致するには、タグ付けされた条件式 `(.+)` が使用されています。タグ付けされた条件式 `(.+)` は、「1つまたは複数の文字 `.+` の発生をマッチし、このマッチを記録します。」を意味します。次のステップで表示されているように、後でタグ付けされた条件式を参照する必要があります。

4. 置換フィールド内に、次のテキストを入力します：`<genre>\1</genre>`。この正規表現置換えのテキストを定義します。検索フィールドから既にタグ付けされた条件式にバックレファレンスするために `\1` が使用されていることに注意してください。すなわち、このコンテキストでは、`\1` 「現在一致する <category> 要素の内部のテキスト」を意味します。
5. **全てを置き換える** をクリックして、結果を確認します。すべての `category` 要素が、このサンプルの目的である `genre` に名前を変更されました。

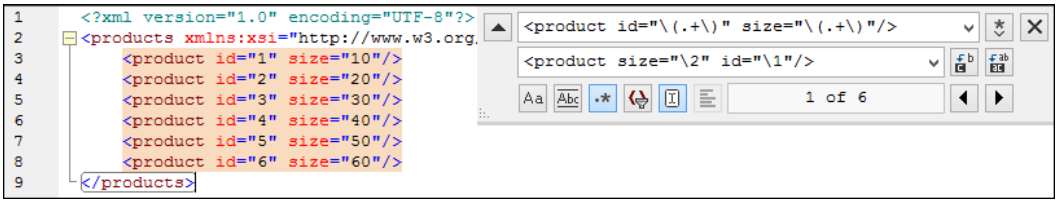
サンプル 2：属性の順序を変更する

下のサンプルXML コードリストには製品のリストが含まれています。各製品には以下の2つの属性が存在します：`id` と `size`。各 `product` 要素内の `id` と `size` 属性の順序の変更をこのサンプルの目的とします。（すなわち、`size` 属性が `id` の前に来るように変更します）。この条件を満たすために正規表現を使用することができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<products xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="products.xsd">
   <product id="1" size="10"/>
   <product id="2" size="20"/>
   <product id="3" size="30"/>
   <product id="4" size="40"/>
   <product id="5" size="50"/>
   <product id="6" size="60"/>
</products>
```

必要条件を満たすには、以下のステップを踏んでください：

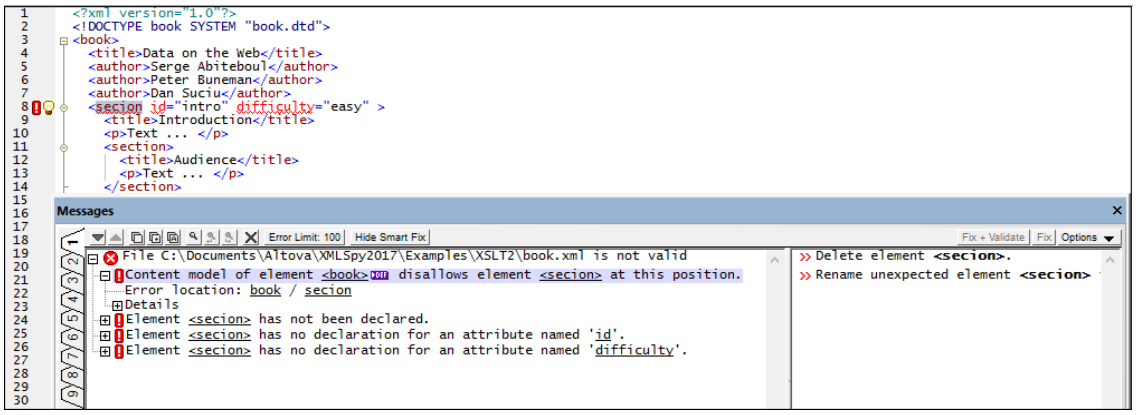
1. 検索と置換ダイアログボックスを開くには、**Ctrl+H** を押します。
2. **正規表現の使用** をクリックします。
3. 検索フィールドに次のテキストを入力します：`<product id="(.+)" size="(.+)"/>`。この正規表現は、XML ドキュメント内の製品要素に一致します。（既知ではない）各属性の値に一致するためには、タグ付けされた条件式 `(.+)` が2度使用されていることに注意してください。タグ付けされた条件式 `(.+)` は、各属性の（`.+` である文字の1度、または複数回の発生である）値に一致します。
4. 検索フィールドに次のテキストを入力します：`<product size="\2" id="\1"/>`。この正規表現には、一致した製品の要素のテキストの代替テキストが含まれています。`\1` と `\2` の2つの参照が使用されていることに注意してください。これらは、検証フィールドからタグ付けされた条件式に対応しています。すなわち、`\1` は、「属性 `id` の値」を意味し、`\2` は「属性 `size` の値」を意味します。



6. **全てを置き換える** をクリックして、結果を確認します。すべての `product` 要素は属性 `size` が属性の `id` 前に置かれるように更新されました。

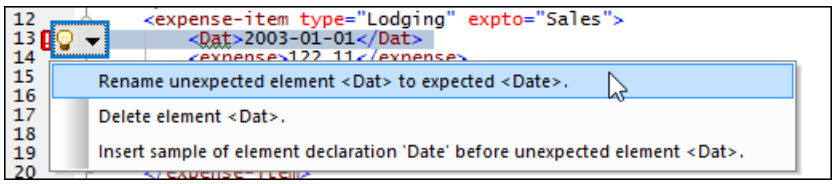
2.1.6 XML ドキュメントの検証

XML | 検証 (F8) コマンドは、ドキュメントを関連したDTD、XML スキーマ または、他のスキーマに対して検証します。ドキュメントが妥当な場合、検証に成功したことを示すメッセージがメッセージウィンドウに表示されます。それ以外の場合、左側のペイン内にエラーの原因が表示されます (*下のスクリーンショットを参照してください*)。左側のペインで原因が選択されている場合、使用できる場合は、右側のペインに表示されているスマートフィックスを使用します。スマートフィックスの提案は、関連付けられたスキーマ内の情報をベースにしています。スマートフィックスを適用するには以下を行います: (i) ダブルクリックする、(ii) 選択して、**フィックス** または **フィックス + 検証** オプションを選択します (*下のスクリーンショットを参照してください*)。



赤い四角はウィンドウの右側にあるスクロールバーの左横に表示されることに注意してください (*上のスクリーンショットを参照*)。スクロールバーは次の個別の2つの種類のマーカーを表示するためここで説明されています: (i) テキストの発生をハイライトする際に表示されるマーカー (茶色、スクロールバーの左側に表示されます)。次を参照: *ドキュメント内の移動*)。 (ii) テキストの発生の検索 (茶色。スクロールバーの左側に表示されます)。 次を参照: *検索と置換*)。

スマートフィックスがエラーのために使用できる場合、電球アイコンがエラーを生成するラインに表示されます (*下のスクリーンショットを参照*)。マウスをアイコンにポイントすると、使用することのできるスマートフィックスをリストするポップアップが表示されます (*スクリーンショットを参照*)。適用するフィックスを選択します。



メモ: 検証エラーのインジケーターとクイックフィックスはXMLSpy 内で検証されるドキュメントの型、例えばJSON ドキュメントなど、に対して使用することができます。

メモ: 上記の検証エラーインジケーターとスマートフィックスは、**XML | 検証 (F8)** コマンドが実行された場合のみ更新されますが、背景では更新されません。ですから、エラーを修正した後、**検証 (F8)** コマンドを再度実行し、エラーが修正されていることを確認してください。

XML ドキュメントの検証に関する詳細は、以下を参照してください: **検証** コマンド。

2.1.7 テキストビューでの入力ヘルパー

テキストビューで利用可能な入力ヘルパーの内容は編集中のドキュメントの種類により変化します。標準的なドキュメントの種類に対する入力ヘルパーは以下に示されています。入力ヘルパーの一般的な使用方法については以下に記述されてい

ます。特定のドキュメントの種類に対応したその他の機能については、それぞれのドキュメントの種類のセクションでそれぞれ記述されます。

- *XML*：要素（下のスクリーンショット）、属性、エンティティ

- *HTML*：要素、属性、エンティティ
- *CSS*：CSS アウトライン、CSS プロパティ、HTML 要素
- *DTD*：無し
- *XQuery*：XQuery キーワード、XQuery 変数、XQuery 関数
- *WSDL*：概要、詳細
- テキスト：エンティティ

XSD、XSLT、XHTML や RDF などのいくつかのドキュメントは、本質的には XML ドキュメントであり、要素、属性、そしてエンティティ入力ヘルパーが表示されます。

入力ヘルパーアイテムの表示と使用方法

入力ヘルパーに表示される様々なアイテムは色つきで表示されます。これらの色についてはそれぞれのドキュメントの種類で説明されます。大まかに言うと、入力ヘルパーについて以下のことを知っておくべきです：

- 入力ヘルパーはコンテキスト依存で、カーソルの位置に挿入することができるアイテムが表示されます。
- 選択された場所（またはその他の妥当な場所）に同じアイテムが既に挿入され、例えば XML の属性などで同様の挿入が行えない場合、アイテムはグレーで表示されます。
- アイテムが必須の場合、感嘆符（!）アイコンが隣に表示されます。
- カーソルがある位置に入力ヘルパーのアイテムを挿入するには、入力ヘルパーのアイテムをダブルクリックします。
- 入力ヘルパーから要素が挿入された場合、その開始と終了タグもドキュメントテキストに挿入されます。**「オプション」**ダイアログ（**「ツール | オプション | 編集」**）によりそう指定されている場合、必須の要素も一緒に挿入されます。
- 属性入力ヘルパーにより属性が挿入された場合、属性値の入力のためにイコールサイン（=）と二重引用符（""）も属性と一緒に挿入されます。カーソルは二重引用符の中に置かれ、そのまま属性値を入力できるようになります。

メモ： 大きなファイルでは、自動補完と入力ヘルパーを無効化し、より速いロードと編集を行うことができます。しきいファイルのサイズはユーザーにより指定することができます。詳細は[オプション | 編集]のセクションを参照してください。

2.1.8 テキストビューショートカット

テキストビューコマンドで頻繁に使用されるデフォルトのショートカットは、以下にリストされているとおりです。[カスタマイズダイアログのキーボードタブ]内でデフォルトのショートカットを変更することができます。

- *（検証と変換のためのショートカットも含む）機能キーのショートカット*

F1	ヘルプメニュー
F1 + Alt	最後に開かれたファイルを開く
F3	次を検索する
F4 + CTRL	アクティブなウィンドウを閉じる

F4 + Alt	XMLSpy を閉じる
F5	更新
F6 + CTRL	開かれているウィンドウを循環する
F7	整形式チェック
F8	検証
F10	XSL 変換
F10 + CTRL	XSL:FO 変換

ファイルとアプリケーションコマンド

Alt + F1	最後に開かれたファイルを開く
CTRL + O	ファイルを開く
CTRL + N	新規ファイル
CTRL + P	ファイルを印刷する
CTRL + S	ファイルを保存する
CTRL + F4	アクティブなウィンドウを閉じる
CTRL + F6	開かれているウィンドウを循環する
CTRL + TAB	開かれているドキュメントを切り替える
Alt + F4	XMLSpy を閉じる

文字キーパッドのショートカット (ノードの展開、折りたたみ)

Num +	展開
Num *	完全に展開する
Num –	折りたたむ
CTRL + Num –	選択されていない範囲を折りたたむ

その他のキー

上 /下矢印キー	カーソルまたは選択バーを移動する
Esc	編集を破棄する、または、ダイアログボックスを閉じる
Return	選択を確認する
Del	文字、または、選択範囲を削除する
Shift + Del	切り取る

編集コマンド

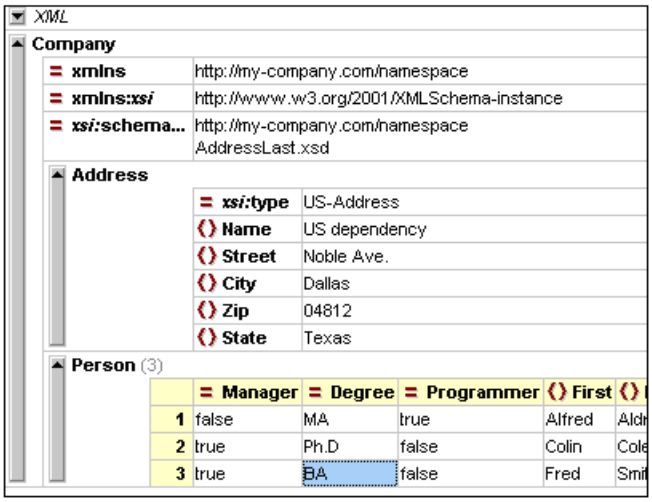
CTRL + A	全てを選択する
CTRL + F	検索する
CTRL + G	ライン/文字に移動する
CTRL + H	置き換える
CTRL + V	貼り付ける
CTRL + X	切り取る
CTRL + Y	やり直し
CTRL + Z	元に戻す

Text View commands

CTRL + E	Jump between Start/End Tags
CTRL + Shift + E	Select Element that Contains Cursor
CTRL + Alt + E	Go to Parent Element
CTRL + "+"	Zoom In
CTRL + "-"	Zoom Out
CTRL + 0	Reset Zoom
CTRL + mousewheel forwd	Zoom In
CTRL + mousewheel back	Zoom Out

2.2 グリッドビュー

グリッドビュー (下のスクリーンショットを参照) はXML ドキュメントやDTD の階層構造をネストされたコンテナーという形で表示します。より良いドキュメントの構造を知るためにこれらのコンテナーを折りたたむこともできます。グリッドビューのコンテンツと構造は簡単に操作することができます。



階層的なアイテム (例えばXML 宣言やドキュメントの種類、宣言、または子要素を含む要素など) はコンテナーの左側に位置している矢印のついたグレーのバーで表示されます。バーをクリックすることでアイテムを展開したり折りたたんだりすることができます。要素は ⟨⟩ アイコンで表され、属性は = アイコンで表されます。

表示とナビゲーション

- アイテムの内容はその種類により変化します。例えば、要素の場合だとコンテンツは属性、文字データ、コメント、そして子要素になります。属性は常に表示され、ファイルの内容に従って並べられます。属性に続いて、アイテムがソースファイルの内容と全く同じように表示されます。この順序はドラッグアンドドロップにより変更することができ、変更はソースファイルにも反映されます。
- 要素に文字データだけが含まれている場合、そのデータは要素と同じ行に表示され、その要素も自動的に階層構造を持っているとは見なされません。その他の文字データは属性やその他の子要素と一緒にインデントされて表示され、"Text" とラベル付けされます。
- 要素が折りたたまれている場合、その属性と値は同じ行にグレーで表示されます。この属性のプレビューは特に属性と内容が異なる同名の要素を大量に持つXML ドキュメントを編集しているときに役に立ちます (例えばデータベースのようなアプリケーションです)。
- 矢印キーにより選択中の行を移動することができます。
- テンキーの + そして -キーによりアイテムを展開したり折りたたんだりすることができます。

グリッドビューのカスタマイズ

- 列の大きさを変更するには、マウスポインターを境界線まで移動させ、ドラッグアンドドロップにより大きさを変更します。
- 列の幅を最適な幅に変更するには、列の右側にあるグリッドの線をダブルクリックします。
- 全てのコンテンツを表示するように列の幅を最適化するには、メニューアイテムの **表示 | 最適な幅** コマンドを選択するか、最適な幅アイコン ↔ をクリックします。
- セルの高さはコンテンツにより決定されます。セルの高さは **『ツール | オプション | ビュー | 拡張グリッドビュー』** の

「セルの高さ制限」にて設定することができます。

注意：グリッドビューでマークしたデータは、テキストビューに移動した後でもそのまま引き継がれます。

インテリジェントな編集
グリッドビューでスキーマをベースにしたXMLドキュメントを編集しているときにはインテリジェント編集が使用できます。詳細はをグリッドビューでの編集参照して下さい。グリッドビューで使用される入力ヘルパーについては以下で記述されます。

グリッドビューテーブル
グリッドビューでは再帰的な要素をテーブル（グリッドビューテーブル）で表示することができます。この機能は全てのXML ファイル（XML、XSD、XSLTなど）で使用可能です。

2.2.1 グリッドビューでの編集

DTD やXML スキーマの様なスキーマをベースにしたXML ドキュメントを編集中に、グリッドビューではスキーマから得られた情報をもとにしたインテリジェント編集を使用することができます。以下にこれらの機能の解説を行います。

要素屋属性名の挿入または追加
選択されたアイテムに関連する要素や属性を挿入するには、XML メニューのコマンドを使用するか、**属性と要素ツールバー** にあるアイコンを使用します。このツールバーが表示されていない場合、メニューオプションの**「ツール | カスタマイズ」**を選択して、**ツールバー**タブから**属性と要素**にチェックを入れてください。ツールバーにあるこれらアイコンコマンドとXML メニューコマンドについては、XML メニューの項を参照ください。

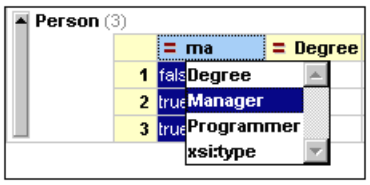
要素または属性を挿入または追加するには：

1. 挿入または追加する要素または属性に関連するアイテムを選択します。
2. 属性と要素ツールバー内の適切なアイコンをクリックするか、XML メニューから適切なコマンド（**挿入**、**追加**、または**子を追加**）を選択します。
3 目的の要素または属性をポップアップから選択し、クリックするか「Enter」を押下します。それ以外にも、目的の要素または属性の名前を入力することができます。

グリッドビューで選択されているものによって、入力ヘルパーの内容が常に変わり続けるのに気がついたかと思います。情報ウィンドウに表示される情報も、選択されている要素や属性に従って常に変わり続けます。

グリッドビューにおける改行とタブ
グリッドビューのセル内部にて改行を入力するには、「Ctrl」+「Enter」を押下してください。タブを入力するには、「Ctrl」+「タブ」を押下してください。

要素または属性の名前を編集する
要素や属性の名前を編集する時には、利用可能なオプションがポップアップで表示されます。これらのオプションは要素の位置やスキーマで定義されたコンテンツモデルに依存します。要素や属性のコンテンツが列挙やchoice などにより制限されている場合にも似たようなポップアップが表示されます。



要素名または属性名を編集するには：

1. 要素または属性名をダブルクリックします。利用可能なオプションがポップアップで表示されます。
2. ポップアップメニューから必要な要素または属性を選択します。

名前空間プレフィックスを追加する

「XML | 名前空間プレフィックス」コマンドによりグリッドビューの選択されたノードとその子孫（要素と属性）に名前空間プレフィックスを追加することができます。名前空間はこれとは別に宣言されなければなりません。

グリッドビューコンテキストメニュー

メニューとツールバーで使用できるコマンドに加えて、グリッドビューではメニューで最も頻繁に使用されるコマンドの集合が収められたコンテキストメニューが存在します。コンテキストメニューを使用するには動作させたいアイテムの上で右クリックして下さい。

スペルチェック

グリッドビューでは、内蔵の言語辞書を使用することでドキュメントのスペルチェックを行うことができます。ユーザー辞書を作成することで、言語処理に含まれていない単語を加えることができます。詳細については、スペリング および スペルチェッカーのオプション コマンドを参照ください。

XML 検証

グリッドビューではユーザーの入力を部分的に検証することができます。例えば、属性値における出現とパターンのエラーを検知することが可能です。しかしテキストだけを含めることができる要素に子要素を追加しても、エラーとは検知されません。XML ドキュメントが妥当であることを保証するには、テキストビューへ切り替えた後に検証処理を行なってください。

2.2.2 グリッドビューテーブル

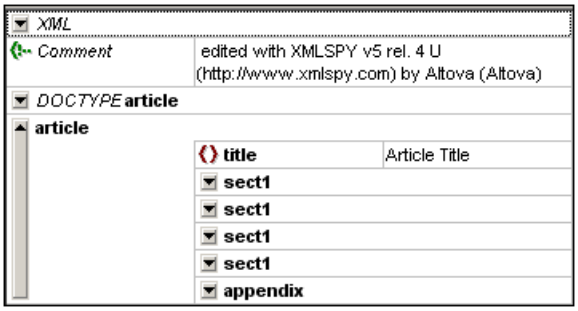
テーブルビューは再帰的な要素をテーブルの形で表示するグリッドビューに統合されたビューです。テーブルビューは通常のグリッドビューとは違い、テーブルとして表示されている要素の子となる型をそれぞれ列として表現します。それから全ての列や選択に対してのプロパティなどを設定することができます。

例えば、以下のようなXML ドキュメントを考えてみましょう：

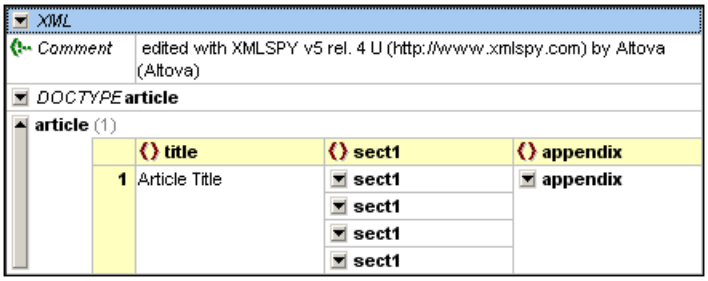
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (ht
<!DOCTYPE article SYSTEM "C:\Program Fi
<article>
  <title>Article Title</title>
  <sect1>
    <title>Section 1</title>
    <simpara/>
  </sect1>
  <sect1>
    <title>Section 2</title>
    <para/>
  </sect1>
  <sect1>
    <title>Section 3</title>
    <para/>
    <para/>
  </sect1>
  <sect1>
    <title>Section 4</title>
    <para/>
  </sect1>
  <appendix>
    <title/>
    <para/>
  </appendix>
</article>
```

ドキュメントの要素は`article` で、`article` は以下の子要素を含んでいます：`title` 要素 1つ、`sect1` 要素 4つ、`appendix` 要素を 1つ。`sect1` と`appendix` 要素はそれぞれ 1つの`title` 要素を含んでおり、任意の数の`para` か`simpara` 要素がそれに続きます。

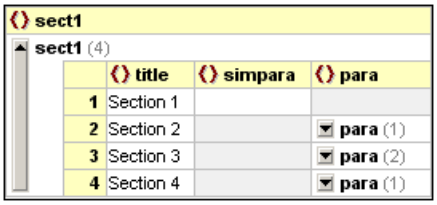
このドキュメントは通常のグリッドビューでは以下のように表示されます：



そして以下がこのドキュメント（より正確には`article` 要素）のテーブルビューになります。このビューを表示するには、通常のグリッドビューで`article` 要素をクリックして選択した後、テーブルとして表示アイコン をクリックするか、メニューから**「XML | テーブルテーブルとして表示（F12）」**を選択します。



（テーブルとして表示されている）`article` の子要素に注目して下さい。`title`、`sect1`、そして`appendix` が列として表示され、それぞれの出現が対応する列に表示されています。テーブルの構造は1つ下の子要素までにしか適用されないことにも注目して下さい（子要素の`sect1` 要素はテーブルとして表示されていません）。`sect1` 要素をテーブルとして表示するには、`sect1` 列にある任意の要素を選択し、 をクリックします。`sect1` 要素もテーブルとして表示されます（下のスクリーンショットを参照）：その下の子要素はそれぞれ列として`sect1` テーブルの中に表示されます。



それぞれの列では、4つある出現のなかで子要素（title、simpara、またはpara）が存在する場合、対応するセルの背景は白色で表示されます（例：sect1 最初のsimpara）。その出現に要素が存在しないのであれば、対応するセルの背景はグレーで表示されます（例：sect1 最初のpara）。従って、列にはsect1 の子要素全ての出現の集合が割り当てられ、ユニークな子要素の種類により列が作成されていると言うことができるでしょう。

注意：属性も子ノードと見なされ、属性に対しても列が作成されます。特定の要素を選択して、 をクリックするか、「F12」を押すことで通常のグリッドビューとの行き来を行うことができます。テーブルビューからテキストビューに移動した後、再度グリッドビューに移動すると、ドキュメントは通常のグリッドビューで表示されます。

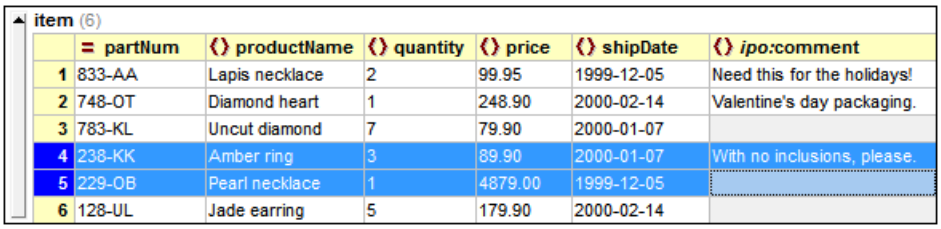
テーブルデータを操作する

以下のようにしてテーブルのデータを操作することができます：

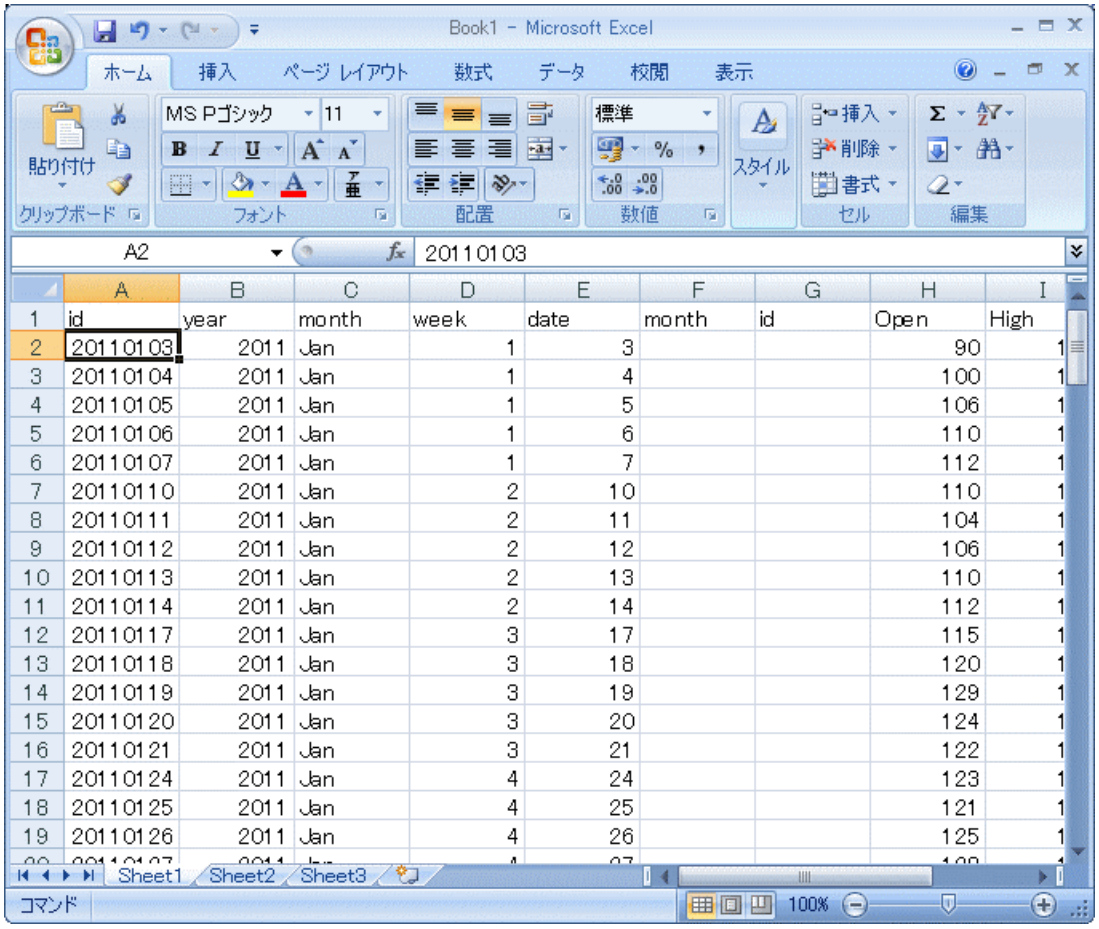
- 列のヘッダをドラッグアンドドロップして列を移動する。
- メニューコマンド「XML | テーブル | 昇順に並び替え」を使用してテキストノードの列のデータをソートする（降順でも可能）。
- メニューコマンド「XML | テーブル | 行を挿入」（または**行を追加**）で行の挿入（や追加）を行う。

テーブルビューと外部アプリケーションでデータをやり取りする

テーブルビューによるテーブル構造のデータを利用して、スプレッドシートアプリケーションとテーブルビューとの間でデータのやり取りを行うことができます。テーブルビューから他のアプリケーションにデータを移動したい場合は、テーブル内の必要なノードを選択したあと、構造化テキストとしてコピーオプションを使用して、データを直接（例えば）Excel のシートにコピーすることができます（ノードの選択はセル、列ヘッダ、下のスクリーンショットの様な）行ヘッダ、またはテーブル全体をクリックすることで行うことができます）。テーブル全体または列ヘッダを選択した場合は、列ヘッダも含めてコピーされます。



上のスクリーンショットではテーブルに11のbook 要素が表示されています。テーブル全てをExcel シートにコピーするには、book (11) 要素を選択し、構造化テキストとしてコピーし、Excel シートに貼りつけます。Excel にてA1 セルを選択し、貼り付けを行った結果を以下に示します。詳細に関しては、構造化されたテキストとしてコピーするを参照してください。



	A	B	C	D	E	F	G	H	I
1	id	year	month	week	date	month	id	Open	High
2	20110103	2011	Jan	1	3			90	1
3	20110104	2011	Jan	1	4			100	1
4	20110105	2011	Jan	1	5			106	1
5	20110106	2011	Jan	1	6			110	1
6	20110107	2011	Jan	1	7			112	1
7	20110110	2011	Jan	2	10			110	1
8	20110111	2011	Jan	2	11			104	1
9	20110112	2011	Jan	2	12			106	1
10	20110113	2011	Jan	2	13			110	1
11	20110114	2011	Jan	2	14			112	1
12	20110117	2011	Jan	3	17			115	1
13	20110118	2011	Jan	3	18			120	1
14	20110119	2011	Jan	3	19			129	1
15	20110120	2011	Jan	3	20			124	1
16	20110121	2011	Jan	3	21			122	1
17	20110124	2011	Jan	4	24			123	1
18	20110125	2011	Jan	4	25			121	1
19	20110126	2011	Jan	4	26			125	1

データのやり取りは逆の方向にも行うことができます。スプレッドシートアプリケーションからデータをコピーして、直接テーブルビューにデータを貼り付けることができます。

1. 外部アプリケーションでコピーする範囲を選択し、コピーします (Windows システムでは **「Ctrl」+C** によりクリップボードにコピーされます)。
2. XML ドキュメントのテーブルビューにてセルを1つ選択します。
3. **「Ctrl」+V** によりデータを貼りつけます。

テーブルビューで選択されている箇所にオリジナルのデータ構造に対応したデータが XMLSpy のテーブルに張り付けられます。以下のポイントに注意して下さい：

- テーブルビューのセルに既にデータが存在する場合、それらのデータは新しいデータにより上書きされます。
- 既にある行よりも貼り付けに多くの行が必要になる場合、新しい行が作成されます。
- 既にある列よりも貼り付けに多くの列が必要になる場合、新しい列は**作成されません。**
- セルのデータは対応するセルのコンテンツとなります。

更に複雑なデータのやり取りを行う方法として、XMLSpy にはテキストファイル、Word ドキュメント、またはデータベースから XML データをインポート (またはエクスポート) する機能があります。

2.2.3 グリッドビューでの入力ヘルパー

要素入力ヘルパー

グリッドビューでは、要素入力ヘルパーには3つのタブがあります： 「追加」、 「挿入」、そして 「子を追加」です。 **「追加」**タブには現在の要素の後に兄弟として追加することのできる要素が表示されます。 **「追加」**タブには現在の要素の前に挿入することのできる要素が表示されます。そして **「子を追加」**タブには現在の要素の子供として挿入することのできる要素が表

示されます。

要素を挿入するには、対応するタブを選択して必要な要素をダブルクリックして下さい。必須な要素は感嘆符（!）のマークが付いていることに注目して下さい。許された要素の兄弟でありながら、カーソルがある位置に挿入することができない要素は利用できません。

注意：「オプション」ダイアログ（**「ツール | オプション | 編集」**）から、要素が挿入されたときに必須の子要素も一緒に挿入されるよう設定することができます。

属性入力ヘルパー

属性入力ヘルパーには編集中の要素で使用することのできる属性の一覧が表示されます。必須の属性には属性名の隣に感嘆符（!）が表示されます。要素に対して既に属性が与えられている場合、その属性はグレーで表示されます。

- 追加ならびに挿入タブで属性を使用するには、グリッドビューで既存の属性か、属性の親要素の子である要素を選択し、必要な属性を入力ヘルパーからダブルクリックします。
- 子を追加タブで属性を使用するには属性の親要素を選択し、必要な属性をグリッドビューの入力ヘルパーからダブルクリックします。

注意：既存の属性で複数の出現が許されていないものはグレーで表示されます。

エンティティ入力ヘルパー

エンティティ入力ヘルパーには内部または外部のDTD の宣言されたエンティティを全て表示します。グリッドビューにてテキストノードまたは属性ノードが選択されている場合、そのようなノードは子を持つことができないので、「子を追加」タブは空で表示されます。

グリッドビューにてカーソルを使ったエンティティの挿入を行うには、テキストフィールドの必要な場所にカーソルを移動するか、必要なフィールドを選択した後、適切なタブを選択し、エンティティをダブルクリックします。以下の点に注意して下さい：

- カーソルが（属性フィールドを含む）テキストフィールド内に位置しているとき、カーソルの挿入ポイントにエンティティは挿入されます。
- （#PCDATA やsimpleType のような）テキストしか表示できない子が選択されていながらカーソルはテキストフィールド内にあるとき、既存のテキストコンテンツがエンティティにより**置き換えられます**。
- テキストでないフィールドが選択されている場合、選択された入力ヘルパーのタブに対応した位置にテキストとしてエン

ティティが作成されます。

注意：内部エンティティを追加した場合、その上方をエンティティ入力ヘルパーに表示するためには、ドキュメントを一度保存して再度開く必要があります。

2.2.4 グリッドビューショートカット

グリッドビューコマンドで頻繁に使用されるデフォルトのショートカットは、以下にリストされているとおりです。カスタマイズダイアログのキーボードタブ内でデフォルトのショートカットを変更することができます。

- *(検証と変換のためのショートカットも含む)機能キーのショートカット*

F1	ヘルプメニュー
F1 + Alt	最後に開かれたファイルを開く
F3	次を検索する
F4 + CTRL	アクティブなウィンドウを閉じる
F4 + Alt	XMLSpy を閉じる
F5	更新
F6 + CTRL	開かれているウィンドウを循環する
F7	整形式チェック
F8	検証
F10	XSL 変換
F10 + CTRL	XSL:FO 変換

- *ファイルとアプリケーションコマンド*

Alt + F1	最後に開かれたファイルを開く
CTRL + O	ファイルを開く
CTRL + N	新規ファイル
CTRL + P	ファイルを印刷する
CTRL + S	ファイルを保存する
CTRL + F4	アクティブなウィンドウを閉じる
CTRL + F6	開かれているウィンドウを循環する
CTRL + TAB	開かれているドキュメントを切り替える
Alt + F4	XMLSpy を閉じる

- *文字キーパッドのショートカット(ノードの展開、折りたたみ)*

Num +	展開
Num *	完全に展開する
Num –	折りたたむ
CTRL + Num –	選択されていない範囲を折りたたむ

- *その他のキー*

上/下矢印キー	カーソルまたは選択バーを移動する
Esc	編集を破棄する、または、ダイアログボックスを閉じる
Return	選択を確認する

Del	文字、または、選択範囲を削除する
Shift + Del	切り取る

編集コマンド

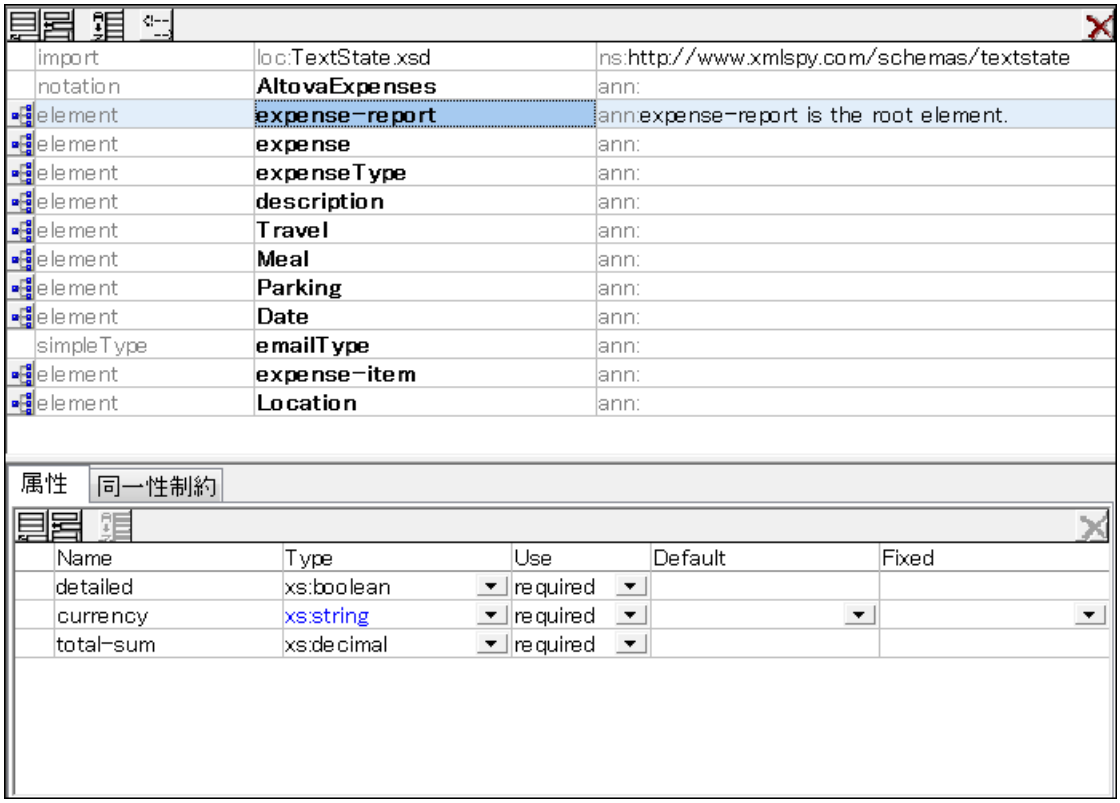
CTRL + A	全てを選択する
CTRL + F	検索する
CTRL + G	ライン/文字に移動する
CTRL + H	置き換える
CTRL + V	貼り付ける
CTRL + X	切り取る
CTRL + Y	やり直し
CTRL + Z	元に戻す

Grid View commands

CTRL + D	Append CDATA
CTRL + E	Append Element
CTRL + I	Append Attribute
CTRL + M	Append Comment
CTRL + T	Append Text
CTRL + Shift + D	Insert CDATA
CTRL + Shift + E	Insert Element
CTRL + Shift + I	Insert Attribute
CTRL + Shift + M	Insert Comment
CTRL + Shift + T	Insert Text
CTRL + Alt + D	Add Child CDATA
CTRL + Alt + E	Add Child Element
CTRL + Alt + I	Add Child Attribute
CTRL + Alt + M	Add Child Comment
CTRL + Alt + T	Add Child Text

2.3 スキーマ ビュー

スキーマビューではXML スキーマの参照と編集を行うことができます (以下のスクリーンショットを参照)。グラフィカルなインターフェイスと一般的なGUI の機能によりスキーマの構築を素早く正確に行うことができます。スキーマビューには (i) スキーマコンポーネント間の構造的なリレーションシップをデザインするための上部ペインと (ii) 上部ペインで選択されたコンポーネントに関連する定義を行うための下部ペインが表示されます。スキーマビューには、妥当なスキーマを作成するためのコンポーネント、詳細、ファセット入力ヘルパーも搭載されます。



上部ペイン: スキーマのデザイン

スキーマビュー上部のペインでは 2 つのモードを切り替えることができます:

- スキーマ概要: スキーマの全グローバル コンポーネント (グローバル要素や複合型など) がシンプルなテーブル リストで表示されます (スクリーンショットを参照)。グローバル コンポーネントのアイコンをクリックすることで、そのグローバル コンポーネントがコンテンツ モデル ビューで表示されます。すべてのグローバル コンポーネントがコンテンツ モデルを持てるわけではない (例えば単純型) ということに注意してください。
- コンテンツ モデル ビュー: 選択されたグローバル コンポーネントのコンテンツ モデルが表示されます (スクリーンショットを参照)。ペイン左上にあるグローバルの表示アイコンをクリックすることで、スキーマ概要が表示されます。

コンテンツ モデル ビューへの切り替え: コンテンツ モデルを含むグローバル コンポーネントに対して利用でき、グローバル コンポーネントのコンテンツ モデルがコンテンツ モデル ビューにて表示されます。

グローバルの表示: コンテンツ モデル ビューで選択することができ、スキーマ概要への切り替えを行います。

下部ペイン：属性、アサーション、同一性制約

スキーマビューの下部には、(上部ペインの)デザインで選択されているコンポーネントの属性、アサーション、同一性制約の定義が表示されます (スクリーンショットを参照)。このペインは便宜上 AAIDC ペインと呼びます。

- XSD 1.0 モードでは、(i) 属性と (ii) 同一性制約という 2 つのタブが下部ペインに表示されます。
- XSD 1.1 モードでは、(i) 属性、(ii) アサーション、(iii) 同一性制約の 3 つのタブが下部ペインに表示されます。

AAIDC ペインはスキーマ概要で常に表示されるほか、コンテンツ モデル ビューにて表示されることもあります。コンテンツ モデル ビューでは、3 種類の定義 (属性、アサーション、同一性制約) すべてを AAIDC ペインではなくデザイン内部に表示することもできます。スキーマ デザイン ツールバーの (i) **属性を図に表示**、(ii) **アサーションを図に表示**、(iii) **制約を図に表示** アイコンを切り替えることで、これらの表示を切り替えることができます。スキーマ表示構成ダイアログ (「スキーマ設計 | 表示設定」) にてこれらの設定を指定することもできます。AAIDC ペインのすべての定義型が図内に表示され、コンテンツ モデル ビューにて下部ペインが表示されなくなります。

スキーマ設定

スキーマ設定ダイアログ (「スキーマ設計 | スキーマ設定」) へはスキーマ ビューからアクセスすることができ、アクティブなスキーマに対するグローバルな設定を定義することができます。これらの設定は `xs:schema` 要素の属性に対応します。

このセクションの構成

このセクションは、以下に示されるサブセクションにより構成されます：

- XSD モード：XSD 1.0 と 1.1: 2 つの編集モードから選択を行えます。
- スキーマ概要: グローバル コンポーネントのプロパティを編集します。
- コンテンツ モデル ビュー: 個々のグローバル コンポーネントのコンテンツ モデルを編集します。
- 属性、アサーション、同一性制約: コンポーネントに含まれるこれらプロパティを定義します。
- 入力ヘルパー: コンポーネントに含まれる様々なプロパティを素早く定義します。
- スマート制約: 基底型から派生した型を視覚的に作成ならびに編集します。
- xml の使用：固定された属性: `base`、`id`、`lang`、`space` 属性をスキーマ コンポーネントへ視覚的に追加します。
- 戻る/進む：位置の移動: 以前に表示された内容を表示するためのスキーマ ビュー機能について説明します。

SchemaAgent へ接続

XMLSpy からは SchemaAgent への接続を行い、他のスキーマから得られたコンポーネントを GUI に表示して、これらコンポーネントを編集されているスキーマにて使用することができます。XMLSpy にて SchemaAgent を使用する方法については、「SchemaAgent を使用する」のセクションを参照ください。

スキーマの中を検索

スキーマの中を検索機能により、スキーマ内部のインテリジェントな検索 (スキーマに関連した条件による絞り込み) を行うことができます。例えば、検索の結果を特定種類のコンポーネントへ絞り込むことで、より効率的な検索を行うことができます。スキーマの中を検索機能の詳細については、「DTD と XML スキーマ」のセクションを参照ください。

2.3.1 XSD モード: XSD 1.0 と 1.1

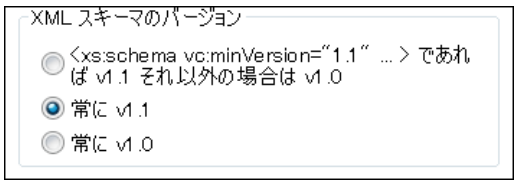
スキーマビューでは、XML スキーマ (XSD) が編集されるか、そして、XML Schema 1.0 仕様 (XSD 1.0) または XML Schema 1.1 仕様 (XSD 1.1) に従い検証されるかを選択することができます。**XSD モード**は、次の 2 つの設定をベースにした編集で使用されます : アプリケーション内、他の XSD ドキュメント内。

XSD モードの選択

XSD モードは、アクティブなドキュメントのためのスキーマビューの編集と検証機能 (XSD 1.0 または 1.1) を決定します。以下のどちらかを行うことができます : (i) 選択されたモードですべての XSD ドキュメント スキーマビュー内のか編集されることのできるアプリケーション全体の設定にする、または (ii) XSD ドキュメント内の XSD バージョン番号を保存して、この情報に従い、アプリケーションは、XSD モードを自動的に選択します。

アプリケーション全体モード

アプリケーション全体設定は、オプションダイアログ のファイルタブで行います (**[ツール | オプション]**、*下のスクリーンショット参照*)。*バージョン 1.0* または *バージョン 1.1* ラジオボタンを選択すると、選択されたモードはアプリケーション全体モードになります。スキーマビュー内で開かれたすべての XML スキーマ ドキュメントはこのモードで編集することができます。(*v1.1 を選択する場合は、if <xs:schema vc:minVersion="1.1" の場合 です。その他の場合は、v1.0 設定を選択します。* モードは、ドキュメント内の情報により、アプリケーション全体ではありません。この点に関しての情報に関しては下の他のセクションとドキュメント固有のモードを参照してください。)



オプションダイアログの XML スキーマ バージョン設定内で選択することにより、いつでも 2 つの アプリケーション全体モード間 (*バージョン 1.0* および *バージョン 1.1*) を切り替えることができます (*上のスクリーンショット*)。

メモ: 現在の設定が、アプリケーション全体設定である場合、設定およびモードの切り替えは、スキーマデザイン ツールバー (*次のセクション参照*) 内の **XSD 1.0** ツールバーボタンまたは **XSD 1.1** ツールバーボタンを使用して行うことができます。モードの切り替えは、一時的であり、ドキュメントが再ロードされると、アプリケーションの全体モードに切り返されます。ビューの変更またはスキーマビューが更新されると **[ファイル | 再ロード]** を介して 再ロードが実行されます。

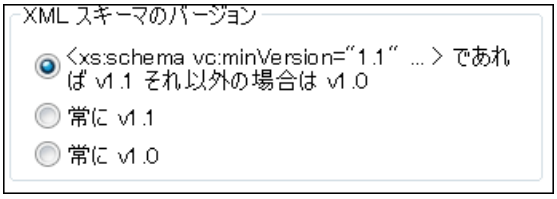
ドキュメント固有のモード

XSD ドキュメント内に XSD モード情報を保存することを選択できます。ドキュメントがロードされると、自動的にスキーマビューの XSD モードへの切り替えを有効化します。スキーマデザイン ツールバー内の **XSD 1.0** または **XSD 1.1** ボタンをクリックすることにより、XSD モード情報を XSD ドキュメントに追加することができます。(*下のスクリーンショット*)。これを行うにあたり、選択されたモードは、トップレベル `xs:schema` 要素の `vc:minVersion` 属性内に保存されます (`vc:minVersion` の値は、手動でテキストビューに追加することもできます)。

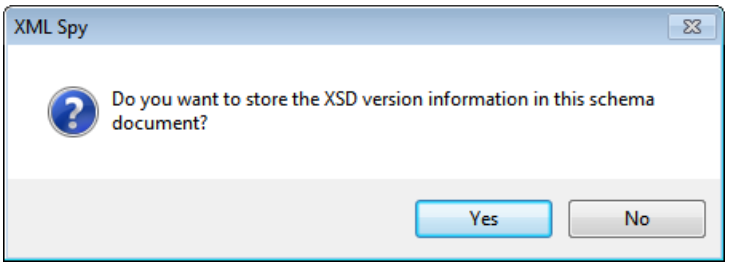
メモ: vc:minVersion 属性 が存在する場合は、名前空間 http://www.w3.org/2007/XMLSchema-versioning 内に存在する必要があります。この場合は、XML スキーマ ドキュメントは、この名前空間にvc: 名前空間 プレフィックスをバインドする名前空間宣言が必要になります。**XSD 1.1** ツールバーボタンを使用すると、名前空間は自動的に追加されます (*上のスクリーンショット*)。ドキュメント内の他のノード名がこの名前空間に存在しない場合、**XSD 1.0** ツールバーボタンをクリックすると、名前空間宣言が削除されます。

ドキュメント固有の モードをアクティブ化し、ドキュメントのXSD モードを指定するには、次を行います :

1. *ドキュメント固有のモードのアクティブ化 :オプションダイアログのファイル タブ (* **「ツール | オプション」** *)で、XML スキーマ バージョン オプション<xs:schema vc:minVersion="1.1"の場合は、v1.1 に、その他の場合は v1.0 に設定します (下のスクリーンショット参照)* 。これは、スキーマビュー内のXSD モードがxs:schema 要素のvc:minVersion 属性に対して設定されるようにXMLSpy に示します。



2. *ドキュメントのXSD バージョンの指定 :スキーマデザイン ツールバー内の (上のスクリーンショット)* **XSD 1.0** または**XSD 1.1** ボタンをクリックします。確認 ダイアログがポップアップします (*下のスクリーンショット*)。



3. **「はい」** をクリックすると、次の結果を引き起こします : (i) xs:schema 要素の vc:minVersion 属性 に対応する値を入力します。 (i) **XSD 1.1** が選択されている場合、vc: 名前空間 プレフィックスへのバインドと共に XMLSchema-versioning 名前空間が宣言されます。**XSD 1.0** が選択されている場合、XMLSchema-versioning 名前空間内に他のノードが存在しないと、名前空間宣言は削除されます。XML スキーマ ドキュメントは、XSD バージョン番号を含むようになります。 ファイルを保存する際、XSD モード情報も保存されます。ファイルを再度開くまたは再ロードすると、スキーマビューは、xs:schema 要素のvc:minVersion 属性に含まれているとおり自動的にドキュメントのXSD モードに切り替えられます。

メモ: ドキュメント固有の モード オプションが選択され、そしてXSD ドキュメントがvc:minVersion 属性 または 1.0 または1.1 以外のvc:minVersion 属性の値を持つ場合、 スキーマビューのデフォルトはXSD 1.0 モードです。

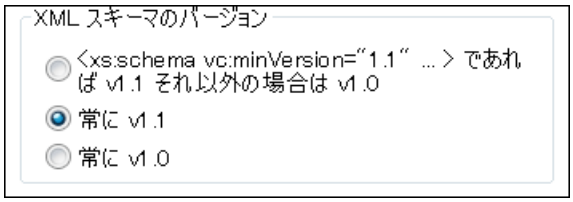
メモ: vc:minVersion 属性 と xsd:version 属性を混同しないように注意してください。前者は、XSD バージョン番号を有し、後者は、バージョン番号を有します。

新しいドキュメントのXSD モード

新しいXSD ドキュメントを作成する場合、XSD 1.0 またはXSD 1.1 ドキュメントとして作成するかを問われます。XSD 1.1 が選択されると、新しいドキュメントが属性 `/xs:schema/@vc:minVersion="1.1"` と共に作成され、`vc:` 名前空間 プレフィックスへのバインドを持つ `XMLSchema-versioning` 名前空間が宣言されます。XSD 1.0 が選択されると、`vc:minVersion` 属性または`XMLSchema-versioning` 名前空間宣言が追加されません。しかしながら、スキーマビュー内でどのXSD モードが有効化されるかは、オプションダイアログのファイルタブで選択されている XML スキーマ バージョンによります (**「ツール| オプション」**)。 これら2つの設定どのように相互に作用するかについての詳細は、次のセクションを参照してください。

有効化されたXSD モード

XSD モードが有効化される場合、スキーマビュー内 の以下に影響されます (i) 存在 /不在—そして、存在する場合、/ XSD ドキュメントの`xs:schema/@vc:minVersion` 属性の値、および (ii) オプションダイアログのファイル タブで選択された XML スキーマ バージョン オプション(**「ツール| オプション」**、下のスクリーンショット)。



以下の シチュエーションが考えられます。下のテーブルの *XML スキーマ バージョン*は、上に表示されるXML スキーマ バージョン ペイン内の選択を参照します。下のテーブル内の`vc:minVersion` 値 はXML スキーマ ドキュメント内の `xs:schema/@vc:minVersion` 属性の値を参照します。

XML スキーマ バージョン	`vc:minVersion` 属性	XSD モード
常にv1.0	存在しません、または すべての値と存在します	1.0
常にv1.1	存在しません、または すべての値と存在します	1.1
@vc:minVersion の値	属性は、`1.1` の値を持ちます	1.1
@vc:minVersion の値	属性は 存在しません、または 属性は `1.1` 以外の値と存在します	1.0

メモ: 最初の 2つの行で説明されている シチュエーションの場合、 XSD 1.1 スキーマをXSD 1.0 モード内で開くこと またその逆は可能です。矛盾の扱い方については下で説明されています。

XSD モード機能

インターフェイスおよびスキーマビューの編集機能は、有効化されるXSD モード (XSD 1.0 またはXSD 1.1) により変更されます。

XSD 1.0 モードが有効化される場合 :

- 新しいXML Schema 1.1 コンポーネント および プロパティへの編集サポートを使用することができません。しかしながら、XSD 1.1 コンポーネント または プロパティがXSD ドキュメント内で既存の場合、これらは表示され、削除されることもできます。

- 検証は、XSD 1.0 仕様に対して行われます。ですから、 XSD 1.1 コンポーネント または プロパティが (既に) スキーマ内に存在する場合、検証 エラーが報告されます。

XSD 1.1 モードが有効化されていると、 XML Schema 1.1 仕様に関するXML Schema 1.1. 検証のすべての機能のための編集サポートが提供されます。

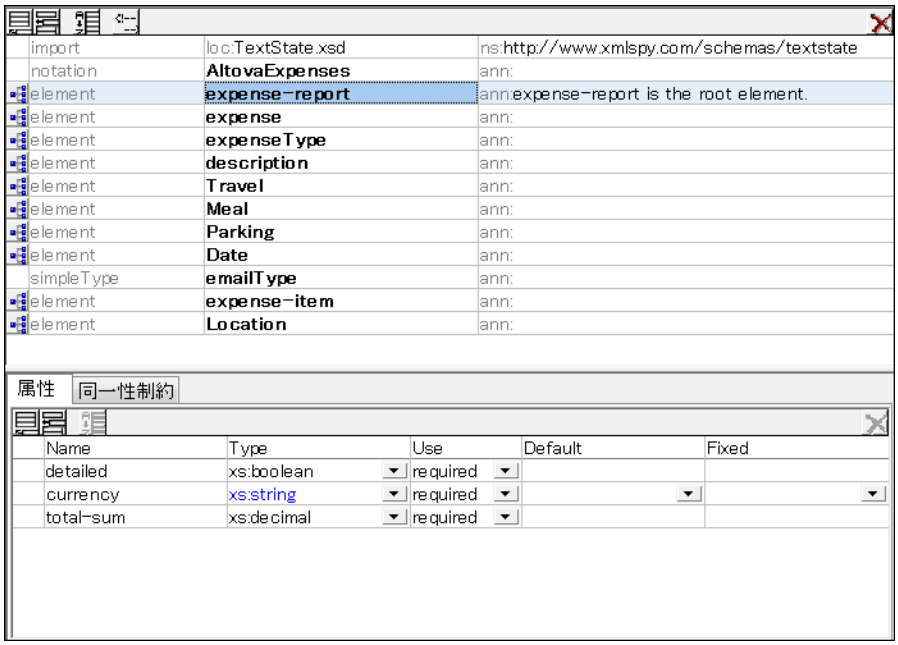
XSD 1.0 モード内での XSD 1.1 機能の扱いかた

XSD 1.0 によりサポートされないXSD 1.1 機能 (例えば、アサーション) がドキュメント内に存在する場合、その 機能が表示され、 そして以下に説明されるXSD 1.0 モードにより取り扱われます。

- ***アサーション:*** 選択された単純型でアサーションが1つでも存在する場合、ファセット入力ヘルパーで*アサーション*タブが存在します。 アサーションの削除以外の編集を行うことはできません。
- ***アサート:*** 存在する場合、複合型ダイアグラム内でアサーションが表示されます。AAIDC ペインでアサーション タブを使用することはできません。また、 アサーションはコンテキストメニューを介し追加されることはできません。 削除以外のプロパティの編集を行うことはできません。
- ***属性:*** 存在する場合、新しい プロパティ`inheri`テーブル が表示されます。空の値を選択する以外の編集を行うことはできません (これはより効果的なプロパティの削除方法です)。
- ***複合型:*** 存在する場合 新しい プロパティ`defaultAttributesApply` が表示されます。in 詳細入力ヘルパー。空の値を選択する以外の編集を行うことはできません (これはより効果的なプロパティの削除方法です)。
- ***ドキュメント:*** 新しいXSD 1.1-固有の コンポーネント および プロパティは、スキーマビュー内 ドキュメントに含まれていません。
- ***ファセット:*** 未知のファセットは、検証 エラー を引き起こし、 そして赤で表示されます。
- ***スキーマ内の検索:*** 新しいXSD 1.1-固有の コンポーネント および プロパティは、無視されます。
- ***ID 制約 (IDC):*** プロパティ`isRef` が表示されます。参照の場合、はオフに切り替えることができます。 名前の IDC がオフに切り替えられると、
- ***複数の代替 グループ:*** コンボボックスは、単一の代替 グループを選択します (XSD 1.0 内では単一の代替 グループのみが許可されています)。
- ***オープン コンテンツ:*** 存在する場合はダイアグラムに表示されます。 コンテキストメニューを介して追加されません。 削除以外の編集を行うことはできません。 デフォルトのオープン コンテンツが、複合型内部で表示されません。
- ***オーバーライド:*** 存在する場合、グローバル グリッドを表示します。これは、 メニューを介して追加されません。削除以外の (ロケーション) の編集を行うことはできません。 コンポーネント (`xs:override` の子) は、無視されます、そして コンポーネント 入力ヘルパーに含まれません。
- ***スキーマ設定:*** 存在する場合、新しい プロパティ`defaultAttributes` および `xpathDefaultNamespace` は、ダイアログ内に表示されます。空の値を選択する以外編集は不可能です (これはより効果的なプロパティの削除方法です)。
- ***単純型:*** 未知の型は検証 エラーを引き起こし、赤字で表示されます。
- ***型 代替:*** 存在する場合ダイアグラムに表示されます。 コンテキストメニューを介して追加されません。削除する以外 (プロパティの) の編集を行うことはできません。
- ***ワイルドカード:*** 新しい プロパティが存在する場合は表示されます。空の値を選択する以外編集は不可能です (これはより効果的なプロパティの削除方法です)。

2.3.2 スキーマ概要

スキーマ概要ではスキーマ内に含まれるすべてのグローバル コンポーネント (`import` 要素、グローバル要素、複合型など) が表示されます (以下のスクリーンショットを参照)。



グローバル コンポーネントの挿入や追加、削除のほか、プロパティを修正することができます。プロパティを修正するには、スキーマ概要リストにてグローバル コンポーネントを選択してください。
選択されたグローバル コンポーネントのプロパティは、詳細入力ヘルパー、ファセット入力ヘルパー、または属性 アサーション/同一性制約 (AAIDC) ペインで編集することができます。

コンテンツ モデルを含んでいるグローバル コンポーネントには、グローバル コンポーネント リストの左に **「コンテンツ モデル ビューへの切り替え」** アイコンが表示されます。このアイコンをクリックすることで、そのコンポーネントがコンテンツ モデル ビューで表示され、コンポーネントのコンテンツ モデルを編集することができます。

コンテンツ モデル ビューへの切り替え: コンテンツ モデルが含まれるグローバル コンポーネントにて表示され、コンポーネントのコンテンツ モデルをコンテンツ モデル ビューにて表示します。

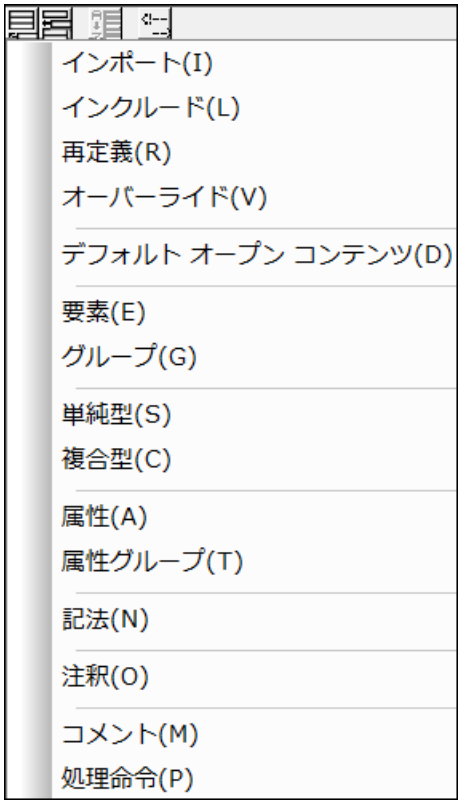
グローバルの表示: コンテンツ モデル ビューにて表示され、クリックすることでスキーマ概要が表示されます。

このセクションでは、最初にスキーマ概要のGUI メカニズムが説明され、各種グローバル コンポーネントの特徴について記述されます。

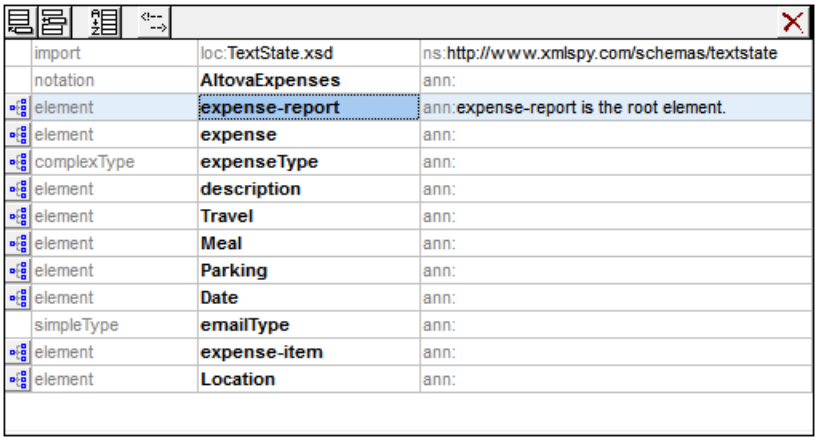
GUI メカニズム

グローバル コンポーネントは、トップレベルの `xs:schema` 要素の子として追加されます。グローバル コンポーネントはペイン上部にある **「追加」** アイコンまたは **「挿入」** アイコンをクリックし、追加するグローバル コンポーネントをメニューから選択することにより追加することができます (以下のスクリーンショットを参照)。

以下のスクリーンショットでは、XSD 1.1 モードで追加することのできるグローバル コンポーネントを左側に、XSD 1.0 モードで追加することのできるコンポーネントが右側に示されます (オーバーライドとデフォルト オープン コンテンツは XSD 1.1 の機能です)。



追加することのできるグローバル コンポーネントの数に制限はありません。スキーマ内に含まれているすべてのグローバル コンポーネントがスキーマ概要のテーブルに表示されます (以下のスクリーンショットを参照)。



	import	loc:TextState.xsd	ns:http://www.xmlspy.com/schemas/textstate
	notation	**AltovaExpenses**	ann:
	element	**expense-report**	ann:expense-report is the root element.
	element	**expense**	ann:
	complexType	**expenseType**	ann:
	element	**description**	ann:
	element	**Travel**	ann:
	element	**Meal**	ann:
	element	**Parking**	ann:
	element	**Date**	ann:
	simpleType	**emailType**	ann:
	element	**expense-item**	ann:
	element	**Location**	ann:

スキーマ概要での編集

スキーマ概要の編集機能に関する以下の点に留意してください:

- スキーマ概要に表示されたリストはドラッグ アンド ドロップにより並び替えることができます。
- キーボードの矢印キーと **[タブ]** キーを使用することで、ナビゲーションを行うことができます。

- グローバル コンポーネントや属性、アサーション、同一性制約を切り取り/コピーして、別の位置やスキーマへ貼り付けることができます。
- コンポーネントを右クリックすることで表示されるコンテキスト メニューにより、そのコンポーネントの注釈データを切り取り、コピー、貼り付け、削除することができます。
- グローバル コメントやグローバルな注釈内で改行を入力するには、**「Ctrl」 + 「Enter」**を押下してください。タブを入力するには、**「Ctrl」 + 「タブ」**を押下してください。

スキーマ概要と関連するアイコン

グローバル コンポーネントの追加: グローバル コンポーネントのリストの最後に新たなグローバル コンポーネントを追加します。定義によりコンポーネントがドキュメントの最初に出現する必要のある場合、リストの最初に追加されます。

グローバル コンポーネントの挿入: 選択されているコンポーネントの直前にグローバル コンポーネントを追加します。定義によりコンポーネントがドキュメントの最初に出現する必要のある場合、リストの最初に追加されます。

並び替え: コンポーネントの並び替えダイアログが表示され、実際に並び替えを行う前に、並び替えを行うための条件（種類の前に名前で並び替えを行う、またはその逆）を指定することができます。以下の記述を参照ください。

コメント: グローバル コメントの表示を複数行で行うのか、または 1行で行うのかを指定することができます。以下にある記述を参照ください。

コンテンツ モデル ビューへの切り替え: コンテンツ モデルが含まれるグローバル コンポーネントにて表示され、グローバル コンポーネントのコンテンツ モデルがコンテンツ モデル ビューにて表示されます。

グローバルの表示: コンテンツ モデル ビューにて表示され、クリックすることでスキーマ概要が表示されます。

スキーマ概要とコンテンツ モデル ビューの切り替え

（複合型、要素宣言、モデル グループなどの）グローバル コンポーネントには、コンポーネント名の左側に**「コンテンツ モデル ビューへの切り替え」**アイコンが表示されます（上記にあるアイコンのリストを参照ください）。これにより、コンポーネントの構造とコンテンツが記述されるコンテンツ モデルを、これらグローバル コンポーネントに含めることができることが示されます。

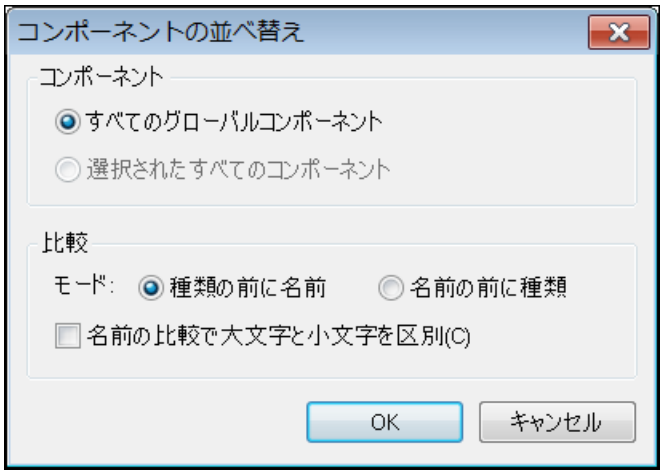
このアイコンをクリックすることで、スキーマ概要から、そのグローバル コンポーネントのコンテンツ モデル ビューへの切り替えが行われます。その他のグローバル コンポーネント（注釈や単純型、属性グループなど）にコンテンツ モデルは含まれておらず、**「コンテンツ モデル ビューへの切り替え」**アイコンも表示されません。コンテンツ モデル ビューでは、**「グローバルの表示」**アイコンをクリックすることでスキーマ概要が表示されます。

グローバル コンポーネントの並べ替え

スキーマ概要ツールバーの**並び替え**アイコンをクリックすることで、グローバル コンポーネントの並べ替えを行うことができます

(上記のアイコンリストを参照ください)。コンポーネントの並べ替えダイアログが表示され、並べ替え可能なすべてのグローバル コンポーネントを並べ替えるのか、または選択されたコンポーネントのセットだけを並べ替えるのか指定することができます。「Shift」キーを押しながらクリックすることでコンポーネントの範囲を選択することができ、「Ctrl」キーを押しながらクリックすることで、選択対象となるコンポーネントを追加することができます。

メモ: ドキュメントの最初に出現しなければならないグローバル コンポーネント (`include` や `import` など) は並び替えの機能により影響を受けません。これらのコンポーネントは、並べ替えを行うことのできるグローバル コンポーネントの範囲に含まれません。

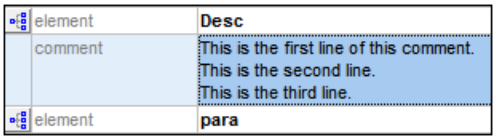


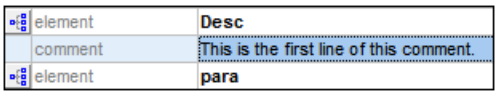
並べ替えを行う範囲を選択した後には、アルファベット順で並べ替えを行うのか、またはコンポーネントの種類により並べ替えを行った後に名前による並べ替えを行うかを選択することができます。

並べ替えの順序はスキーマのテキストにより行われます。

グローバル コメント:行の表示モード

グローバル コンポーネントは複数行のテキスト フィールド、または 1行のテキスト フィールドに表示することができます (以下のスクリーンショットを参照)。





これら 2種類の表示モードを切り替えるには、スキーマ概要ペインの上部にある「コメント」アイコンをクリックして、目的のオプションを選択してください。コメントのテキスト内部に改行を入力する (そして複数行のコメントを作成する) には、「Ctrl」+「Enter」キーを押下してください。コメントを 1行で表示するモードの場合、カーソルを複数行コメントに移動させることで、複数行のコメントがポップアップで表示されます。

グローバル コンポーネント

グローバル コンポーネントとは、トップレベルの xs:schema 要素の子として追加されるコンポーネントのことです (それに対してローカル コンポーネントは、その他のコンポーネント内部で作成されます)。複合型や要素、属性といったグローバル コンポーネントへは、スキーマ内にあるその他のコンポーネントから参照することができます。

スキーマ概要でグローバル コンポーネントを作成する

グローバル コンポーネントは通常スキーマ概要で作成ならびに編集されます。スキーマ概要では、追加または挿入アイコンによりコンポーネントを追加することができます。グローバル コンポーネントのコンテンツ モデル (コンテンツ モデルが存在する場合) は、そのグローバル コンポーネントのコンテンツ モデル ビューにて作成ならびに編集することができます (コンポーネント名の左側にある**コンテンツ モデル ビューへの切り替え**アイコンをクリックすることで、コンテンツ モデル ビューが表示されます)。

スキーマ概要にて作成されたグローバル コンポーネントによってはコンポーネント入力ヘルパーへも追加されます。コンポーネントにコンテンツ モデルが含まれる場合、コンポーネント入力ヘルパーでその名前をダブルクリックすることでコンポーネントがコンテンツ モデル ビューで開かれ、編集を開始することができます。

メモ: コンテンツ モデル ビューで編集作業をしながらもグローバル コンポーネント (要素、属性、単純型、複合型、モデル グループ) を作成することができます。ウィンドウを右クリックして、**新規グローバル | <作成するグローバル コンポーネントの種類>** を選択してください。

メモ: コンテンツ モデル ビューにて編集作業を行う中、ローカル要素をグローバル要素へ、または要素に属性や子要素が含まれる場合にはグローバル複合型へ変換することもできます。ローカル要素が選択されている状態で、ウィンドウを右クリックすることで表示されるコンテキスト メニューから **「グローバル化 | 要素」** または **「グローバル化 | 複合型」** を選択してください。

グローバル コンポーネント	スキーマにおける位置	コンテンツ モデル
include	先頭	なし
import	先頭	なし
redefine	先頭	なし
override 1.1	先頭	なし
defaultOpenContent 1.1	Includes、Imports、Redefined、Overrides の後で、それ以外すべてより前	あり
element	defOpenCont 以降ならどこでも	あり
group	defOpenCont 以降ならどこでも	あり
simpleType	defOpenCont 以降ならどこでも	なし
complexType	defOpenCont 以降ならどこでも	あり
attribute	defOpenCont 以降ならどこでも	なし
attributeGroup	defOpenCont 以降ならどこでも	なし
notation	defOpenCont 以降ならどこでも	なし

annotation	defOpenCont 以降ならどこでも	なし
コメント	どこでも	なし
処理命令	どこでも	なし

スキーマビューでこれらのコンポーネントを編集する際に留意すべき点を以下に記します：

インクルード、インポート、再定義、オーバーライド

これら 4 つのグローバル コンポーネントにより、他のスキーマ ドキュメントを現在編集中のスキーマ ドキュメントにて再利用することができます。

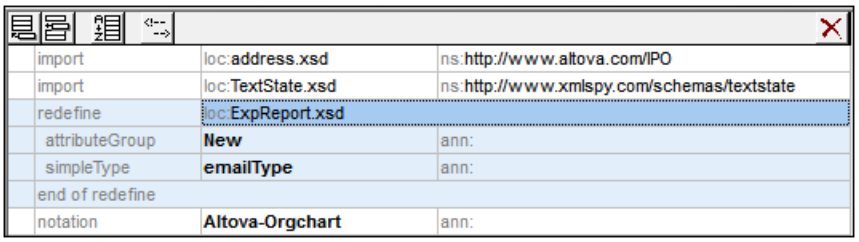
- インクルードにより、現在編集中のドキュメントと同じターゲット名前空間を持つドキュメントを再利用します。
- インポートにより、現在編集中のドキュメントとは異なるターゲット名前空間を持つドキュメントを再利用します。
- 再定義とオーバーライドはインクルードの一種で、現在編集中のターゲット名前空間と同じターゲット名前空間を持つドキュメントを再利用するという点では同じですが、インクルードされたスキーマの一部を修正するという点で異なっています。再定義は XML Schema 1.0 の機能で、1.1 での利用は推奨されておらず、再定義を置き換える XML Schema 1.1 の機能としてオーバーライドが追加されました。

これら 4 つのすべてには再利用されるスキーマを指し示す schemalocation 属性が含まれています。スキーマビューでこれらコンポーネントの loc フィールドをダブルクリックすると、再利用されるファイルの参照と、現在のドキュメントに対して相対となるパスをセットすることができます。更に import コンポーネントにはインポートされたスキーマのターゲット名前空間を示すための namespace 属性が含まれます。

現在編集中のスキーマ ドキュメントにてスキーマが（インクルード、インポート、再利用、またはオーバーライドにより）再利用されると、そのグローバル コンポーネント、名前空間、同一性制約が編集中ドキュメントのコンポーネント入力ヘルパーに表示されます。

再定義

redefine コンポーネントでは、複合型、単純型、モデル グループ、そして属性グループを修正することができます。再定義されるコンポーネントは、redefine コンポーネントの loc フィールドにて指定されます（以下のスクリーンショットでは、ExpReport.xsd スキーマ内にあるコンポーネントが再定義されます）。redefine コンポーネントが追加されると、グローバル コンポーネント リストの redefine と end of redefine 行の間に再定義するコンポーネントを追加する必要があります（以下のスクリーンショットでは、New と emailType コンポーネントが再定義されます）。



以下の操作によりコンポーネントを再定義することができます：

1. end of redefine 行を選択します。
2. スキーマ概要左上にある **挿入** アイコンをクリックします。
3. 定義するコンポーネントの種類（複合型、単純型、モデル グループ、属性グループ）を選択します。redefine コ

ンポーネント内にコンポーネントが追加されます。
4. 再定義するコンポーネントの名前を入力します。再利用されるスキーマから得られたコンポーネントに含まれるすべてのプロパティが表示されます。
5. 詳細ならびにファセット入力ヘルパーからプロパティを選択ならびに修正することができるほか、(コンポーネントにコンテンツモデルが含まれる場合)コンテンツモデルビューにてコンテンツモデルを修正することができます。

メモ: グローバルコンポーネントのリストやコンポーネント入力ヘルパー内から、ドラッグアンドドロップやコピーと貼り付けを行うことで、再定義するコンポーネントを挿入することもできます。

再定義されたコンポーネントは、スキーマ内にある他のコンポーネントから参照することもできます。

オーバーライド

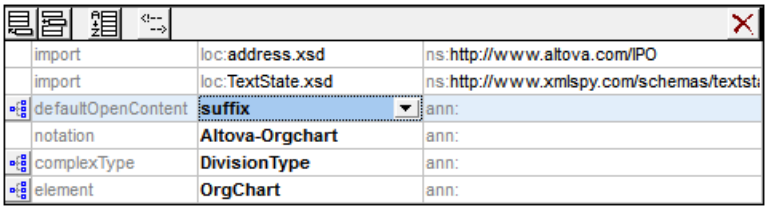
`override` 要素では、以下のコンポーネントを定義することができます:複合型、単純型、グローバル要素、グローバル属性、モデルグループ、属性グループ、記法。`override` 要素内でこれらコンポーネントのうちの1つが定義されている場合、オーバーライドされるすべてのスキーマにある同じ型と名前を持ったすべてのコンポーネントが、このコンポーネントによりオーバーライドされます。オーバーライドされたスキーマは、オーバーライドの `loc` フィールドにて指定されます。

オーバーライドは再定義と違い、再利用されるコンポーネントをベースにすることなく、最初から定義されたコンポーネントとなります。スキーマビューでは再定義を行うためのコンポーネントを追加するのと同様に、オーバーライドを行うためのコンポーネントを追加することになります。`end of override` 行の上にオーバーライドするコンポーネントを挿入し、そのプロパティを定義してください(上記の再定義セクションを参照ください)。オーバーライドと再定義の主な違いは、オーバーライドにコンポーネントが追加されても、そのコンポーネントは再利用されたスキーマから得られたいかなるコンポーネントもベースにしていないということです。

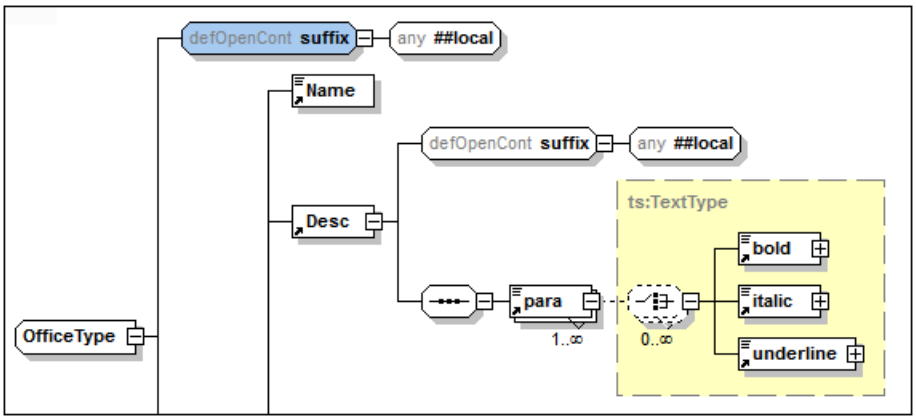
デフォルト オープン コンテンツ

`defaultOpenContent` 要素はXSD 1.1 に新たに追加された仕様で、混合または要素だけのコンテンツを含めることのできる複合型に追加することのできる1つ以上の要素を指定します。XSD 1.1 にて新たに追加された `openContent` 要素に似ており、`openContent` 要素が単一の複合型に対して適用されるのに対して、`defaultOpenContent` 要素はスキーマ内にあるすべての複合型に対して適用されます。

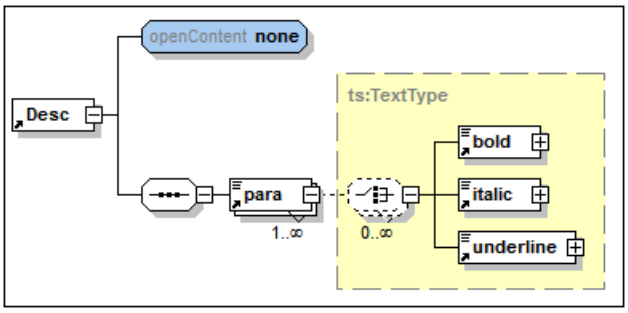
`defaultOpenContent` 要素は `include`、`import`、`redefine` の後、そしてコンポーネントの定義前に1回出現します(以下のスクリーンショットを参照)。要素には `interleave` または `suffix` という値を含めることのできる mode 属性が含まれており、デフォルトの値は `interleave` となります。



`defaultOpenContent` 要素には、コンテンツモデルビューで編集することのできるコンテンツモデルが含まれています。宣言されると`defaultOpenContent` 要素がスキーマ内にあるすべての複合型に対して適用されます。以下のスクリーンショットでは、`defaultOpenContent` が複合型の`OfficeType` ならびに`Desc` 要素に対して自動的に適用されていることが理解できます。



openContent 子要素を追加することで、その複合型に対して適用されたdefaultOpenContent 要素をオーバーライドすることができます。以下のスクリーンショットでは、defaultOpenContent 要素が適用されたDesc 要素にopenContent 要素が追加されており、defaultOpenContent 要素がオーバーライドされます。



グローバル要素 (element)

スキーマ概要ではグローバル要素を作成することができます。グローバル要素にコンテンツ モデルが含まれる場合、グローバル要素のコンテンツ モデル ビューにて定義することができます。要素が選択されている状態で、属性、アサーション、そして同一性制約をAAIDC ペインの対応するタブにて定義することができます。ファセットやその他のプロパティは、要素のファセットならびに詳細入力ヘルパーにて定義することができます。グローバル要素は複合型から参照することができます。

モデル グループ (group)

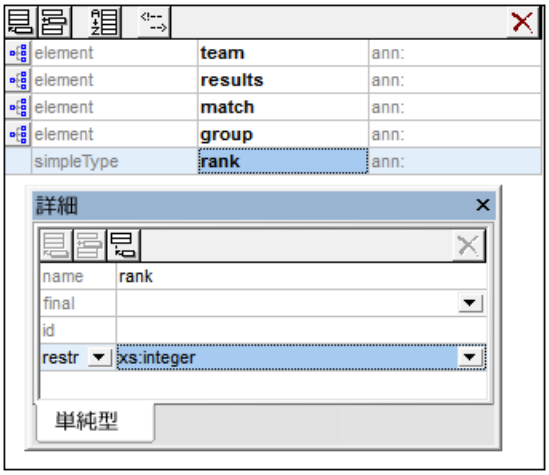
スキーマ概要では名前付きのモデル グループを作成し、複合型にて参照することができます。名前付きのモデル グループ (xs:group 要素) により、再利用可能なコンテンツ モデルを予め定義することができ、sequence グループ、choice グループ、または all グループのうちどれかを含めることができます。

スキーマ概要では、グループ コンポーネントを追加して、名前をつけた後にコンテンツ モデル ビューにてコンテンツ モデルを定義することで、名前付きモデル グループを作成することができます。名前付きモデル グループを複合型のコンテンツ モデルへ追加することができるようになります。

名前付き単純型 (simpleType)

スキーマ概要では、名前付きの単純型を作成し、要素や属性の宣言から参照することができます (以下のスクリーンショットを参照)。

詳細入力ヘルパーでは、単純型のコンテンツ (restriction、list、union) と対応する種類 (それぞれ基底型、アイテム型、メンバー型) を指定することができます。例えば以下のスクリーンショットでは、単純型の制約の基底型に xs:integer が指定されています。詳細については「詳細入力ヘルパー」のセクションを参照ください。ファセットにより単純型の制約を行うには、ファセット入力ヘルパーにあるオプションを使用してください。



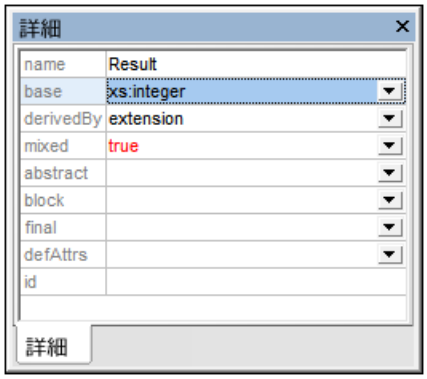
メモ: 匿名型は、スキーマ概要またはコンテンツ モデル ビューにて、単純型コンテンツの属性または要素で宣言することができます。詳細入力ヘルパーにある derivedBy プロパティの値を restriction、list、または union に指定することで、その要素または属性宣言内部に匿名の単純型が作成されます。制約ファセットをファセット入力ヘルパーで、その他のプロパティを詳細入力ヘルパーで定義することができます。

名前付き複合型 (complexType)

スキーマ概要では名前付きの複合型を作成し、要素宣言から参照することができます。名前付きの複合型が選択されている状態で、AAIDC ペインの対応するタブにて属性とアサーションを定義することができます。

複合型には 4 種類のコンテンツを含めることができます (以下のリストを参照)。詳細入力ヘルパーにて各種コンテンツを指定し、必要に応じてコンテンツ モデル ビューにてコンテンツ モデルを定義することができます。

- **単純コンテンツ**: 単純コンテンツの基底型をセットします (以下のスクリーンショットを参照)。mixed 属性には (デフォルト値の) false をセットする必要があり、以下にあるスクリーンショットでセットされた true は赤色で示されています。



- **要素**: コンテンツ モデルにて子要素を作成します。基底型はありません。
- **混合コンテンツ**: `mixed` 属性を`true` にセットする必要があります。文字データは、子要素ノードとともに、要素内のどこにでも配置することができます。文字データはデータ型を持っていないため、基底型をセットすることはできません (上のスクリーンショットを参照)。コンテンツ モデルでは子要素を作成することができます。
- **空のコンテンツ**: 要素には文字データも子要素も含まれません。基底型は含まれず、`mixed` の値を`false` にする必要があります。空のコンテンツ要素のデータは、通常属性に収められます。

メモ 属性とアサーションは (AAIDC ペインにて) 4種類のコンテンツすべてに対してセットすることができます。

メモ: ある要素に対するコンテンツ モデルをコンテンツ モデル ビューで作成することにより、その要素内部で匿名の複合型が作成されます。

グローバル属性と属性グループ (attribute、attributeGroup)

グローバル属性と属性グループはスキーマ概要にて追加されます。

- **グローバル要素**のプロパティは属性の詳細入力ヘルパーにて定義されます。
- **グローバル属性グループ**を作成すると、(i) グローバル コンポーネントのリストにてグローバル属性グループを選択し、(ii) AAIDC ペインの属性タブにて属性を追加し、(iii) 各属性のプロパティを選択された属性の詳細入力ヘルパーにて定義することで、属性をグループへ追加することができます。

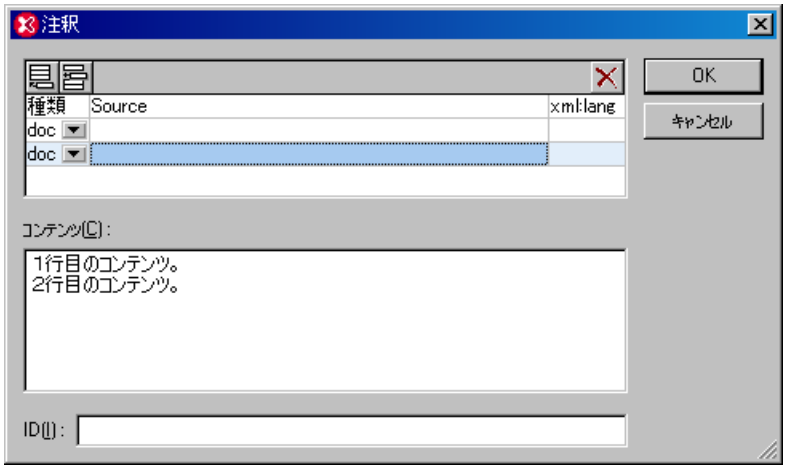
グローバル属性と属性グループが作成されると、要素や複合型の宣言にて参照することができるようになります。

記法 (notation)

記法は常にグローバルなコンポーネントで、ローカルの記法というのは存在しません。記法のプロパティは記法の詳細入力ヘルパーにて指定されます。記法の名前はグローバル コンポーネントのリストから直接指定することができます。スキーマ内にあるすべての記法はコンポーネント入力ヘルパーに表示され、簡単に参照することができます。

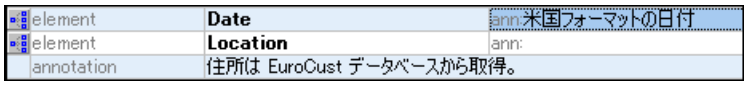
グローバル注釈

グローバル注釈はグローバルなコンポーネントで、いくつかのグローバル コンポーネントにて利用することのできるオプションの注釈とは異なります。注釈グローバル コンポーネントを右クリックして、コンテキスト メニューから **すべての注釈データ** を選択すると注釈ダイアログが表示され、グローバル注釈を編集することができます (以下のスクリーンショットを参照)。



各注釈には id 属性と複数の ドキュメント や appinfo 子要素を含めることができます。ダイアログの左上にある **「追加」** または **「挿入」** ボタンをクリックし、対応するコンボ ボックスから doc または app を選択することで、ドキュメント または appinfo 要素を追加することができます。ダイアログ上部のペインにて doc または app を選択し、そのコンテンツを **「コンテンツ」** ペインにて入力します。コンテンツにて **「Enter」** キーを押下することで、コンテンツ内に改行を含める (そして複数行のコンテンツを作成する) ことができます。上のスクリーンショットでは ドキュメント 要素が選択され、2行からなるコンテンツが確認できます。各 ドキュメント または appinfo 要素に対してオプションの source や xml:lang 属性を入力することもできます。

スキーマ概要では、グローバル注釈にある最初の ドキュメント または appinfo 要素だけが表示され、編集することができます。その複数行のコンテンツが含まれているコンポーネントの場合、マウスオーバーすることにより、すべての行が複数行のポップアップ ボックスにて表示されます。その他 ドキュメント や appinfo 要素のコンテンツを表示するには、グローバル注釈の注釈ダイアログへアクセスしてください。



メモ: いくつかのグローバル コンポーネントで利用することのできるオプションの注釈も注釈ダイアログにて上記と同様の方法により編集することができます。

コメントと処理命令

コメントと処理命令は、スキーマ概要にあるグローバル コンポーネント内の任意の場所へ挿入することができ、コンテンツ モデル ビューに追加することはできません。コメントや処理命令が単純型または複合型内部に含まれる場合、まとめて含まれるオブジェクトの最後に移動されます。このような処理を避けるには、コメントの代わりに注釈の使用が推奨されます。

2.3.3 コンテンツ モデル ビュー

コンテンツ モデルはコンポーネントの構造とコンテンツから構成され、以下のコンポーネントにコンテンツ モデルは含まれます :

- 複合型
- 要素
- モデル グループ
- デフォルト オープン コンテンツ

スキーマ概要に示されたこれらのコンポーネントの左には、**「コンテンツ モデル ビューへの切り替え」** アイコンが表示されます。

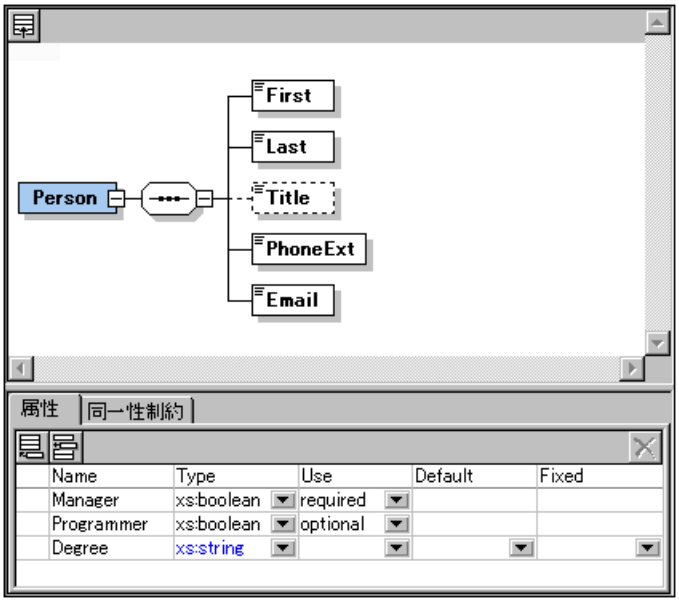
コンテンツ モデル ビューへの切り替え: コンテンツ モデルが含まれるグローバル コンポーネントにて表示され、コンポーネントのコンテンツ モデルをコンテンツ モデル ビューにて表示します。

グローバルの表示: コンテンツ モデル ビューにて表示され、クリックすることでスキーマ概要が表示されます。

コンテンツ モデル ビューへの切り替えアイコンをクリックすることで、そのコンポーネントの**コンテンツ モデル ビュー**が表示されます（以下のスクリーンショットを参照）。その他にも、スキーマ概要にて (i) コンポーネントを選択し、メニューから**「スキーマ設計 | 図の表示」**オプションを選択するか、(ii) コンポーネント入力ヘルパーにあるコンポーネント名をダブルクリックすることでもコンテンツ モデル ビューが表示されます。同時に開くことのできるコンテンツ モデル ビューは、スキーマごとに 1 つしかないという点に留意してください。コンテンツ モデルが表示されている場合、**「Ctrl」**キーを押下しながら現在表示されているコンテンツ モデル内にあるコンポーネントをダブルクリックすることで、そのコンポーネントのコンテンツ モデルへジャンプすることができます。

コンテンツ モデル ビュー全般の説明

コンテンツ モデル ビューではコンテンツ モデルがツリーとして表示されます（以下のスクリーンショットを参照）。メニューの**「スキーマ設計 | 表示設定」**により表示されるスキーマ表示構成ダイアログでは、ツリーの外観を設定することができます。



以下の点に留意してください：

- コンテンツ モデル ツリーにはコンポジッターとコンポーネントの 2 種類あります。更に、属性、アサーション、同一性制約、オープン コンテンツをコンポーネントに接続されたボックス内に表示することもできます。
- ツリーの各階層は、コンポジッターにより接続されます。コンテンツ モデルに含めることのできる階層の数に制限はありません。
- オブジェクトを右クリックすることで表示されるコンテキスト メニューから、そのオブジェクトに対して相対的に他のオブジェクトに対してを追加することができます。
- コンテンツ モデル内にあるコンポーネントは、ローカル コンポーネントとして定義またはグローバル コンポーネントを参照することができます。
- ドラッグ アンド ドロップ機能によりオブジェクトを移動することができます。
- キーボード ショートカットにより、オブジェクトをコピー（**「Ctrl」+C**）または貼り付ける（**「Ctrl」+V**）ことができます。

- オブジェクトのプロパティは詳細入力ヘルパーならびにAAIDC ペインにて編集することができます。
- コンポーネントの属性、アサーション、同一性制約はコンテンツ モデル ビュー下部にあるAAIDC ペインに表示されます。属性と同一性制約はAAIDC ペインの代わりにコンテンツ モデルの図に表示することもできます。この表示オプションはスキーマ表示構成ダイアログにてセットすることができます。表示オプションはスキーマ デザイン ツールバーにある **「図に表示」**ボタンからもセットすることができます (以下のスクリーンショットを参照)。

- コンポーネントを右クリックして、コンテキスト メニューから **「コンポーネントの並び替え」**コマンドを選択することで、兄弟コンポーネントを並び替えることができます。ローカル名とコンポーネントの種類から並べ替えの優先順位を指定することができます。

これらの機能の詳細については、このセクションにあるサブセクションと、チュートリアルを参照ください。

「グローバルの表示」アイコンをクリックするか、メニュー オプションの **「スキーマ設計 | すべてのグローバルを表示」**を選択することで、スキーマ概要が表示されます。

コンテンツ モデル オブジェクト

コンテンツ モデル ビューでは、図の中に表示されたオブジェクトを以下の 3 種類の大まかなグループに分けることができます :

- コンポジッター: (i) シーケンス、(ii) 選択、(iii) すべて
- コンポーネント: (i) 要素、(ii) 複合型、(iii) モデル グループ、(iv) ワイルドカード
- その他: (i) 属性、(ii) 属性グループ、(iii) アサーション、(iv) 制約、(v) オープン コンテンツ

これらのオブジェクトの外観をそれぞれ以下に示します。

コンポジッター

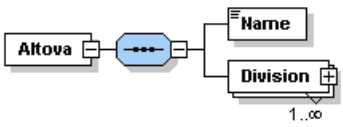
コンポジッターにより、子要素が出現する順序が定義されます。コンポジッターには `sequence`、`choice`、`all` の 3 種類があります。

以下の操作によりコンポジッターを挿入することができます :

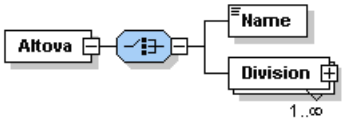
1. 子要素を追加する要素を右クリックします。
2. コンテキスト メニューから **「子の追加 | シーケンス(選択、すべて)」**を選択します。

それぞれのコンポジッターは以下のような外観を持ちます。

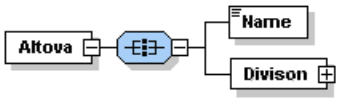
- **シーケンス**



- **選択**



- **すべて**



コンポジッターの種類を変更するには、そのコンポジッターを右クリックして、コンテキスト メニューから **「モデルの変更 | シーケンス(選択、すべて)」** を選択してください。コンポジッターを追加した後には、子要素やモデル グループを追加することになります。

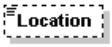
コンポーネント

コンテンツ モデルで使用されるコンポーネントのリストを以下に示します。コンポーネントの種類とその構造に関するプロパティの詳細が、各コンポーネントの外観により示されます。

- 必須の要素 1つ

  詳細: 長方形によりそれが要素であることが示され、実線の境界線によりその要素が必須であることが示されます。番号の範囲が示されていないことから、その要素の出現回数が 1回である (`minOcc=1`、`maxOcc=1`) ことが理解できます。上の例では、`Country` が要素の名前となります。青色によりその要素が現在選択されていることが示されます (クリックすることでコンポーネントを選択することができます)。コンポーネントが選択されていない場合、要素は白色で表示されます。

- 省略可能な要素 1つ

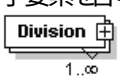
  

  詳細: 長方形によりそれが要素であることが示され、点線の境界線によりその要素が省略可能であることが示されます。番号の範囲が示されていないことから、その要素の出現回数が 1回である (`minOcc=1`、`maxOcc=1`) ことが理解できます。上の例では、`Location` が要素の名前となります。
  メモ コンテキスト メニューのにある **「省略可能」** をクリックすることで、その要素がオプションとなります。

- 複数の必須要素

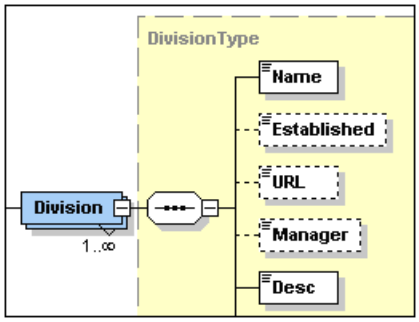
  詳細: 長方形によりそれが要素であることが示され、実線の境界線によりその要素が必須であることが示されます。番号の 1..5は`micOcc=1` ならびに`maxOcc=5` を表しています。上の例では`Alias` が要素の名前となります。

- 子要素を含む複数の必須要素

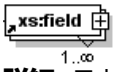
  

  詳細: 長方形によりそれが要素であることが示され、実線の境界線によりその要素が必須であることが示されます。番号の1..∞は`micOcc=1` ならびに`maxOcc=unbounded` を表しています。プラス記号により混合コンテンツ (少なくとも 1つの子要素または属性が含まれる) を示しています。`Division` が要素の名前となります。

メモ: コンテキスト メニューの **「無制限」** コマンドにより、`maxOcc` を `unbounded` にセットすることができます。要素の + 記号をクリックすることで、子要素が表示されます。



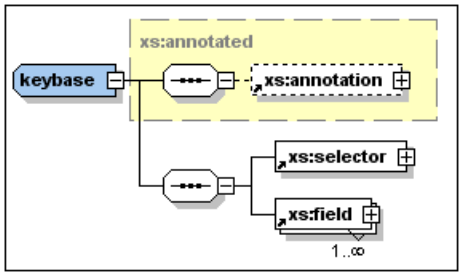
- グローバル要素を参照している要素



詳細: 長方形の左下にある矢印により、要素がグローバル要素を参照していることが示されます。長方形によりそれが要素で、実線によりその要素が必須であることが示されます。番号の 1..∞ は `micOcc=1` ならびに `maxOcc=unbounded` を表しています。プラス記号により混合コンテンツ (少なくとも 1 つの子要素または属性が含まれる) を示しています。`xs:field` が要素の名前となります。
メモ: グローバル要素は単純型や複合型の定義から参照することができ、スキーマ内にある複数の場所で再利用することができます。グローバル要素への参照は、(i) グローバル要素と同じ名前をローカルの要素名として入力するか、(ii) ローカル要素を右クリックしてコンテキスト メニューから **「参照」** を選択することで行うことができます。**「Ctrl」** キーを押下しながら要素をダブルクリックするか、要素を右クリックしてコンテキスト メニューから **「定義へ移動」** を選択することで、その要素が参照しているグローバル要素の定義を確認することができます。存在しない要素への参照が作成された場合、参照される定義がないということを表すため、要素名が赤色で示されます。

- 複合型

詳細: 一部角の取れた長方形とプラス記号により、それが複合型であることが示されます。この例で示される複合型は `keybase` という名前で、このシンボル (角の取れた長方形とプラス記号) により、グローバルの複合型であることが示されます。グローバルの複合型はスキーマ概要で宣言され、そのコンテンツ モデルは通常コンテンツ モデル ビューで定義されます。グローバルな複合型は、(i) 要素のデータ型として使用することができるほか、(ii) 詳細入力ヘルパーにて要素や複合型への割り当てを行うことで別の複合型の基底型として使用することができます。



上に示される `keybase` 複合型は、`xs:annotated` を基底型とするようにスキーマ概要で宣言されています。基底型は灰色の破線を境界線とする長方形と、黄色の背景色により示されます。コンテンツ モデル ビューでは子要素の `xs:selector` と `xs:field` が作成されています (`xs:selector` と `xs:field` 長方形の左下には小さな矢印が示されており、これらの要素が同名のグローバル要素を参照していることが理解できます)。

要素以下に子要素や属性をコンテンツ モデル ビューにて作成することで、ローカルの複合型が定義されます。ローカルの複合型を表す特別なシンボルは存在しません。

メモ: コンテンツ モデルの基底型は、破線のグレー色で表示される境界線と 黄色の背景色により表示されます。名前をダブルクリックすることで、基底型のコンテンツ モデルへ移動することができます。

- モデル グループ

詳細: プラス記号とともにすべての角が取れた長方形はモデル グループを表します。モデル グループにより要素宣言を定義して再利用することができます。
メモ モデル グループを (スキーマ概要にて) 宣言するには名前を与える必要があります。その後、(コンテンツ モデル ビューにて) 要素宣言を含んだ子コンポジッターをモデル グループへ割り当てることになります。モデル グループは、子として挿入、または他のコンポーネントのコンテンツ モデル内に挿入または追加することにより使用されます。

- ワイルドカード

詳細: `any` という名前のついたすべての角が取れた長方形はワイルドカードを表します。
メモ ワイルドカードはスキーマ内部や他の名前空間にて指定されていない要素を表すためのプレースホルダとして使用されます。`##other` = 要素はスキーマで定義されているターゲット名前空間を除くすべての名前空間に属することができ、`##any` = 要素はあらゆる名前空間に属することができます。`##targetNamespace` = 要素はスキーマ内で定義されたターゲット名前空間へ属する必要があります。`##local` = 要素はどの名前空間へも属することができず、`anyURI` = 要素は指定された名前空間に属することになります。

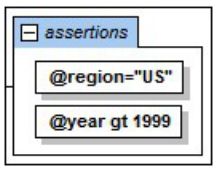
その他のオブジェクト

その他のオブジェクトは属性、属性グループ、アサーション、同一性制約、その他のコンテンツになります。

- 属性、属性グループ

詳細: イタリック体の *'attributes'* という文字と、展開可能な長方形により表示されます。各属性を表す長方形は、(i) 属性が省略可能の場合は破線の境界線、(i) 属性が必須の場合は実線の境界線で表示されます。属性グループと属性ワイルドカードも *'attributes'* の長方形内に表示されます。
メモ 属性は図の内部と詳細入力ヘルパーの両方で編集することができます。属性は、コンテンツ モデルの図内部またはコンテンツ モデル ビュー下部にある AAIDC ペイン内部に表示されます。これら 2 つの表示方法は属性の表示アイコン をクリックすることで切り替えることができます。要素内にある属性の順序を変更するには、属性をドラッグして、目的の場所で矢印が表示されている状態でドロップしてください。

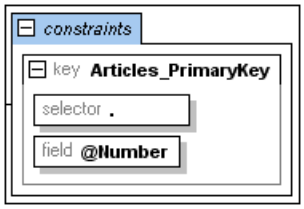
- アサーション



詳細: イタリック体の *'assertions'* という文字と、展開可能な長方形により表示されます。各アサーションは、ボックス内の長方形として示されます。
メモ アサーションは図の内部と詳細入力ヘルパーの両方で編集することができます。アサーションは、コンテンツ モデルの図内部またはコンテンツ モデル ビュー下部にある AAIDC ペイン内部に表示されます。これら 2 つの表示方

法はアサーションの表示アイコン をクリックすることで切り替えることができます。要素内にあるアサーションの順序を変更するには、アサーションをドラッグして、目的の場所で矢印が表示されている状態でドロップしてください。

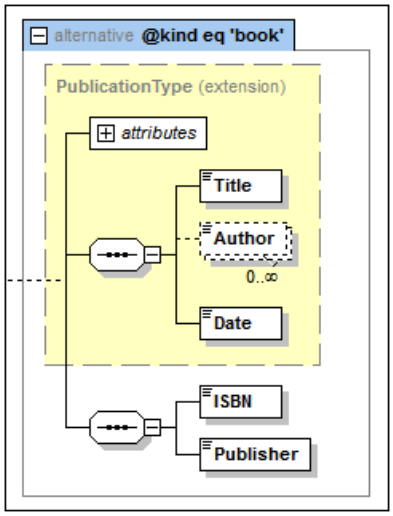
- 同一性制約

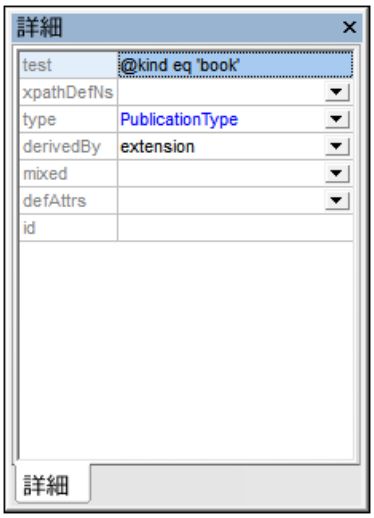


詳細: イタリック体の'*constraints*' という文字と、展開可能な長方形により表示されます。
メモ: コンポーネントのコンテンツ モデルに表示された同一性制約により、`key` と`keyref`、そして`unique` 要素により定義された制約が表示されます。`ID` データ型により定義された同一性制約はコンテンツ モデルの図ではなく詳細入力ヘルパーに表示されます。同一性制約はコンテンツ モデル ビュー、またはスキーマ概要の同一性制約タブにて表示ならびに編集することができます。コンテンツ モデル ビューでは、制約の表示アイコン により、制約ボックスの表示を切り替えることができます。

- 条件型の割り当て

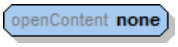
詳細: `alternative` という名前のついた長方形には、評価されるXPath 条件式が含まれます（上のスクリーンショットを参照）。`alternative` 要素の種類は詳細入力ヘルパーにより指定されます。複合型の場合、展開された`alternative` 要素のボックスに表示され、編集を行うことができます（以下のスクリーンショットを参照）。単純型は図に表示されませんが、詳細入力ヘルパーの単純型タブにて定義することができます。





メモ: `alternative` 要素はXSD 1.1 にて新たに導入されました。XPath 条件式の結果が`true` となる場合、`alternative` 要素により指定された型が選択された型となります。`alternative` 兄弟の中で、最初に`true` と評価された`alternative` 要素が選択されるため、`alternative` 要素の順序には意味があります。`alternative` 要素のボックスをドラッグすることで、順序は変更することができます。詳細については、**条件型アサーション**を参照してください。

- デフォルト オープン コンテンツ、オープン コンテンツ



詳細: defaultOpenContent と openContent 要素は、defOpenContent と openContent というラベルとともにコンテンツ モデル ビューに示されます。ワイルドカード要素コンテンツは any ボックスにより示されます (上のスクリーンショットを参照)。
メモ defaultOpenContent ならびに openContent 要素は XSD 1.1 にて新たに追加されたものです。デフォルト オープン コンテンツはグローバル コンポーネントで、スキーマ概要にて作成されます。特定コンポーネントのコンテンツ モデルを表しているコンテンツ モデル ビューでは、デフォルト オープン コンテンツをそのコンポーネントで使用されるオープン コンテンツにより置き換え、スキーマのデフォルト オープン コンテンツをオーバーライドすることができます。コンポーネントの子としてオープン コンテンツを追加してください。デフォルト オープン コンテンツのボックスがオープン コンテンツのボックスに置き換えられます。コンテンツ モデル ビューでは、オープン コンテンツの mode 属性と、そのワイルドカード要素の名前空間を図のなかや詳細入力ヘルパーの両方で編集することができます。任意の複合型のコンテンツ モデル ビューから、スキーマ全体に対して適用されるデフォルト オープン コンテンツを修正することもできます。

メモ:

- スキーマ表示構成ダイアログにて指定された定義済みの詳細情報の表示は、定義済みの詳細情報ツール バー アイコン により切り替えることができます。
- 属性、アサーション、同一性制約の表示アイコンをクリックすることにより、これらの情報をコンテンツ モデルの図の中で表示するか、または (コンテンツ モデル ビュー下部にある) AADIC ペイン内部に表示するかを切り替えることができます。
- コンテンツ モデル ビューでは、**「Ctrl」** キーを押下しながら図の中に表示されているグローバル コンポーネントの名前をダブルクリックすることで、そのグローバル コンポーネントのコンテンツ モデルへジャンプすることができます。基底型の名前をダブルクリックすることでも、基底型のコンテンツ モデルへ移動することができます。

コンテンツ モデル ビューでの編集

コンテンツ モデル ビューで編集を行う方法は、以下のカテゴリーにより説明されます:

- コンテンツ モデル ビューの構成
- 属性、アサーション、同一性制約
- コンテンツ モデル ビュー アイコン
- コンテキスト メニューからの操作
- キーボード ショートカットとドラッグ アンド ドロップ
- コンポーネント プロパティ
- 注釈
- コメントと処理命令
- コンテンツ モデルのドキュメント化

コンテンツ モデル ビューの構成

スキーマ表示構成ダイアログ **「スキーマ設計 | 表示設定」** では、スキーマ全体に対するコンテンツ モデル ビューの構成を指定することができます。構成のオプションに関する詳細については、ユーザー リファレンスにある **「ビューの構成」** セクションを参照ください。ここで行われる設定はスキーマ全体、そしてスキーマ ドキュメンテーションの出力と印刷の出力にも適用されることに注意してください。

属性、アサーション、同一性制約

コンポーネントの属性、アサーション、そして同一性制約は、コンテンツ モデル ビューの下部に表示されるAAIDC ペインに、またはコンテンツ モデル ビューそのもの、つまり図の中に直接表示されます。後者の表示オプションは、スキーマ表示構成ダイアログやスキーマ設計ツールバーにある図の中に表示ツールバー ボタンからセットすることができます（以下のスクリーンショットを参照）。

属性、アサーション、同一性制約の挿入ならびに編集方法については、**属性、アサーション、同一性制約**のセクションを参照ください。

コンテンツ モデル ビュー アイコン

- **グローバルの表示**: コンテンツ モデル ビューで表示され、スキーマ概要への切り替えに使用されます。
- **定義済みの詳細の追加**: コンテンツ モデル ビューが表示されている際にスキーマ設計ツールバーで有効になります。コンポーネント内にある定義済みの詳細情報の表示を切り替えます。
- **属性を図に表示**: コンテンツ モデル ビューが表示されている際にスキーマ設計ツールバーで有効になります。属性の表示を図または属性タブで行うか指定します。
- **アサーションを図に表示**: コンテンツ モデル ビューが表示されている際にスキーマ設計ツールバーで有効になります。アサーションの表示を図またはアサーション タブで行うか指定します。
- **制約を図に表示**: コンテンツ モデル ビューが表示されている際にスキーマ設計ツールバーで有効になります。同一性制約の表示を図または同一性制約タブで行うか指定します。
- **同一性制約の可視化**: コンテンツ モデル ビューが表示されている際にスキーマ設計ツールバーで有効になります。同一性制約の情報表示を切り替えます。

コンテキスト メニューからの操作

コンテンツ モデル ビューにおける編集のいくつかは、コンテンツ モデル ビューで右クリックすることで表示されるコンテキスト メニューから行います（以下のスクリーンショットを参照）。コンテンツ モデル ビューで指定された場所で許されている処理のコマンドだけが有効になります。操作は右クリックされたオブジェクトに対して相対的に行われます。例えば、「子の追加」コマンドを選択すると、右クリックされたオブジェクトに対する子が追加されます。

コンテキスト メニューから利用可能な操作のリストを以下に示します :

- **子コンポーネントとコンポジッターの追加**: **「子の追加」**コマンドにはサブメニューが含まれています。サブメニューから目的のコンポジッターまたはコンポーネントを選択してください。

- **コンポーネントとコンポジッターの挿入 追加**: 右クリックされたオブジェクトと同じ階層にコンポジッターまたはコンポーネントを挿入します。 **挿入」**コマンドにより右クリックされたオブジェクトの前に、 **追加**」コマンドによりその後へオブジェクトが新たに追加されます。

- **コンポジッターの変更**: コンポジッターを右クリックして、 **「モデルの変更 | <新たな種類のコンポジッター>」**を選択してください。

- **グローバル コンポーネントの作成**: (i) **新規グローバル」**コマンドはコンテンツ モデル ビューのあらゆる場所からアクセスすることができます。サブメニューには新たに作成するグローバル コンポーネントを選択するためのメニューが収められています。(i) グローバル コンポーネントとして作成することのできるオブジェクトを右クリックすると、コンテキスト メニューで **「グローバル化」**コマンドが有効になります。このコマンドを選択することで、コンテンツ モデル ビューにあるオブジェクトがグローバル コンポーネントとして作成され、オブジェクトそのものは、新たに作成されたグローバル コンポーネントへの参照となります。

- **出現定義の変更**: **省略可能**」ならびに **無制限」**コマンドを組み合わせることで、コンポーネントの出現設定を (i) 省略可能」= 0 または 1、(i) 省略可能」+ 無制限」= 0から無限、(ii) 無制限」= 1から無限、(v) なし = 1のように指定できます (省略可能」によりコンポーネントの`minOccurs` 属性が、 無制限」により`maxOccurs` 属性が影響を受けます)。

- **ローカルとグローバル定義の切り替え**: ローカル要素と同じ名前のグローバル要素が存在する場合、 **参照」**コマンドによりグローバル要素への参照と、ローカルの定義を切り替えることができます。

- **別のコンテンツ モデルへジャンプ**: コンポーネントを右クリックして **「定義へ移動」**を選択することで、そのコンポーネントのコンテンツ モデルへジャンプします (このコマンドは、コンテンツ モデルを含むコンポーネントに対してのみ有効化されます)。 **「Ctrl」**キーを押下しながらコンポーネントをダブルクリックしてもジャンプすることができます。

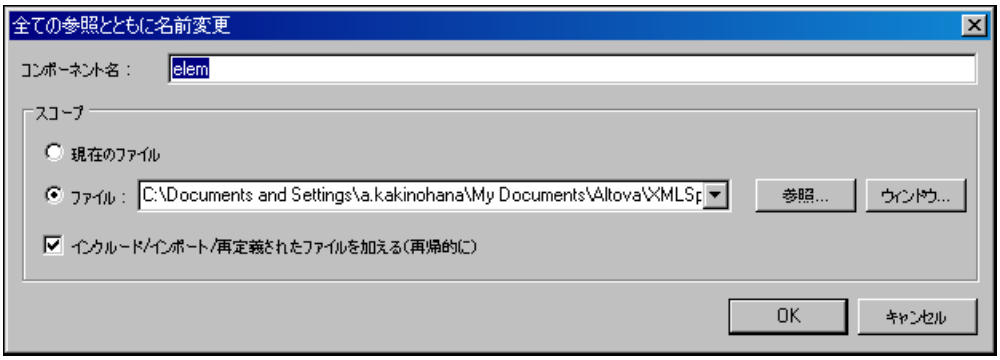
- **定義済みの詳細情報を編集**: 定義済みの詳細が (スキーマ設計ツールバーの**定義済みの詳細を追加**アイコンにて)図の中に表示されるようセットされている場合、編集コマンド以下のサブメニューに定義済みの詳細が表示されます。定義済みの詳細を選択することで、その値を図の中で編集することができます。

- **コンポーネントとコンポジッター注釈の作成**: **「注釈の編集」**コマンドにより、コンポジッター/コンポーネントの下部に注釈を作成するための空間が作成されます（以下のスクリーンショットを参照）。この空間では注釈の入力と編集を行うことができます。注釈が既に存在する場合、コメントをクリックすることで注釈テキストがハイライトされ編集することができるようになります。既存の注釈テキストをダブルクリックすることでも編集を行うことができます。

XML スキーマドキュメントでは、コンポジッターまたはコンポーネントの`annotation/`ドキュメント要素内部に注釈が作成されます。以下にあるドキュメンテーションに関する記述も参照ください。

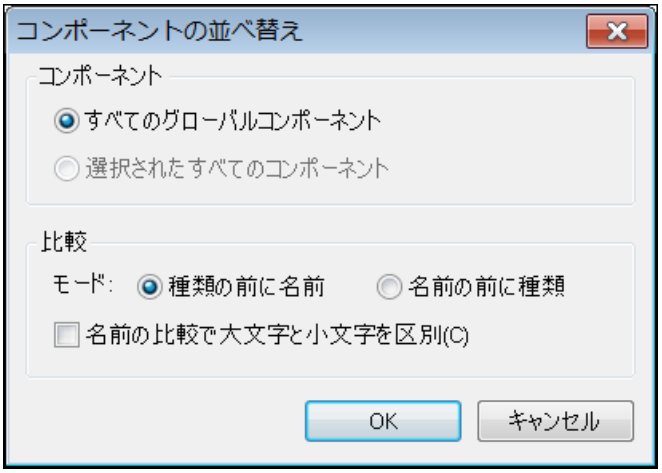
- **インスタンス ノードの XPath をコピー**: **「インスタンス ノードの XPath をコピー」**コマンドは、グローバル要素ならびにグローバルな複合型にて定義された要素と属性に対して有効になります。このコマンドにより、選択されたノードを指定するための XPath 条件式がクリップボードへコピーされます。path 条件式は、コンテンツモデルビューに現在表示されているグローバル コンポーネントから開始されます。

- **コンポーネントの検索と名前変更**: **「すべての参照を検索」**ならびに**「すべての参照とともに名前変更」**コマンドはグローバル要素に対して有効になります。これらのコマンドは、アクティブなドキュメント、またはアクティブなドキュメントに関連するすべてのドキュメントにて選択されたコンポーネントの全出現を検索し、選択されたコンポーネントの全出現を名前変更します。



上のスクリーンショットでは、右クリックされたコンポーネントと、選択されたスコープ内におけるそのコンポーネントへの参照名が EMail に置き換えられます。詳細については「グローバルの検索と名前変更」を参照ください。

- **宣言と参照の並べ替え**: **「並べ替え」**コマンドにより、選択されているすべてのコンポーネントまたは、選択されたコンポーネントの兄弟を並べ替えることができます。コンポーネントの並べ替えダイアログにて設定を行い、**「OK」**をクリックしてください。



「Shift」または「Ctrl」キーを押下しながらクリックすることで、複数のコンポーネントを選択することができます。コンポーネント名とその種類を並び替えのキーとして使用することができます。

キーボード ショートカットとドラッグ アンド ドロップ

キーボードの「Ctrl」+C ならびに「Ctrl」+V ショートカットを使用することで、コンテンツ モデル ビュー内にある要素のコピーと貼り付けをそれぞれ行うことができます。構造的な理由からコピーや貼り付けが行えない場合、その旨のメッセージが表示されます。

その他にも、ドラッグ アンド ドロップにより (i) オブジェクトを図の別の場所へ移動、(ii) 属性などのコンポーネントを入力ヘルパーから図の中へコピーすることができます。

コンポーネント プロパティ

コンポーネントのボックス内に定義済みの詳細が表示されるようにコンテンツ モデル ビューが設定されている場合、図の中でこれら詳細情報を直接編集することができます。定義済みの詳細表示は、**「定義済みの詳細を追加」**ツールバー アイコンをクリックすることで切り替えることができます (上記のアイコン リストを参照ください)。

表示設定がなされている場合、コンポーネントのプロパティを詳細入力ヘルパーで変更することで、その変更がプレースホルダ フィールドに反映されます。

注釈

XML スキーマの注釈は annotation 要素内部に含まれます。注釈には 2 つの種類があり、それぞれ annotation の子要素に含まれます :

- ドキュメント 子要素 : スキーマの編集者にとって有用な情報が含まれます。
- appinfo 子要素 : 処理を行うアプリケーションが使用することのできるスクリプトや情報を挿入することができます。

両方の子要素を含んだ annotation 要素の例を以下に示します :

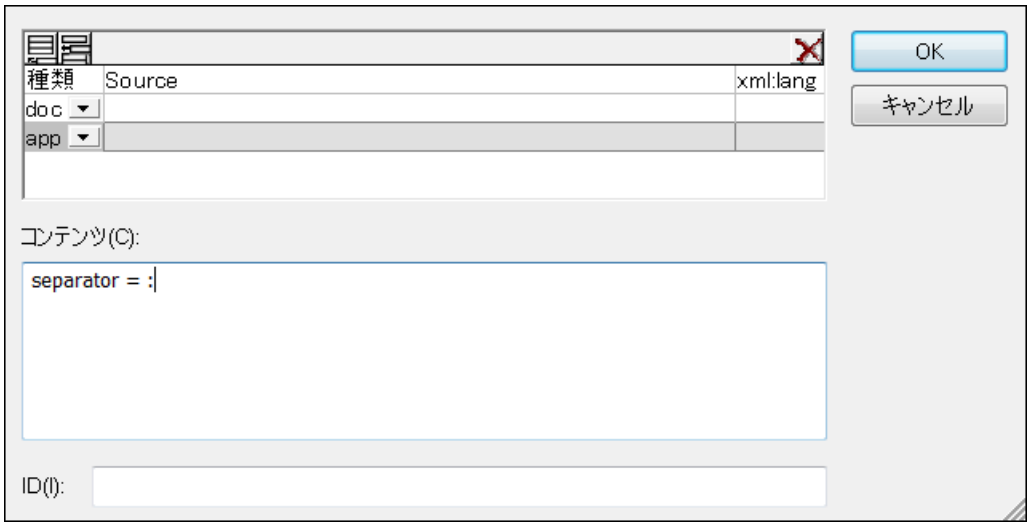
```
<xs:element name="session_date" type="xs:dateTime" nillable="true">
   <xs:annotation>
      <xs:ドキュメント>Date and time when interview was held</xs:ドキュメント>
      <xs:appinfo source="http://www.altova.com/datehandlers/
interviews">separator = :</xs:appinfo>
   </xs:annotation>
</xs:element>
```

コンテンツ モデル ビューでは、以下の操作により個々のコンポジッターやコンポーネントに対して annotation を作成することができます :

1. コンポジッターまたはコンポーネントを右クリックします。

2. コンテキスト メニューから、**「すべての注釈データ」**を選択します。注釈ダイアログ ボックスが表示されます (以下の

スクリーンショットを参照)。その要素に対して注釈 (ドキュメント または appinfo) が既に存在する場合、ダイアログの対応する行に、それらの情報が表示されます。



3. appinfo 要素を作成するには、ダイアログの左上にある追加 または挿入 アイコンをクリックしてください。
4. 新たに作成された行の種類フィールドから app オプションを選択してください。
5. ダイアログのコンテンツペインに、アプリケーションにより処理されるスクリプトや情報を記述します。
6. オプションとして、処理アプリケーションが参照することのできるソースURI を、ソース フィールドに入力することもできます。

コメントと処理命令

XML スキーマ ドキュメントがXMLSpy 内部にロードされるまたは変更されると、単純型ならびに複合型内にあるコメントと処理命令が、それらを含んでいるオブジェクトの最後部にまとめて移動されます。このような移動を望まない場合、コメントの代わりに注釈を使用する必要があります。

コンテンツ モデルのドキュメント化

スキーマに関する詳細なドキュメンテーションをHTML ならびにMS Word フォーマットで生成することができます。各グローバル コンポーネントに関する詳細なドキュメンテーションが生成され、目次ページに表示されたグローバル コンポーネントのリストから、個々のコンポーネントへ素早くアクセスすることができます。更に、関連する要素 (子要素や複合型など) もハイパーリンクにより参照され、要素から要素へのナビゲーションを簡単に行うことができます。スキーマのドキュメンテーションを生成するには、メニュー コマンドから **「スキーマ設計 | ドキュメンテーションの生成」** を選択してください。

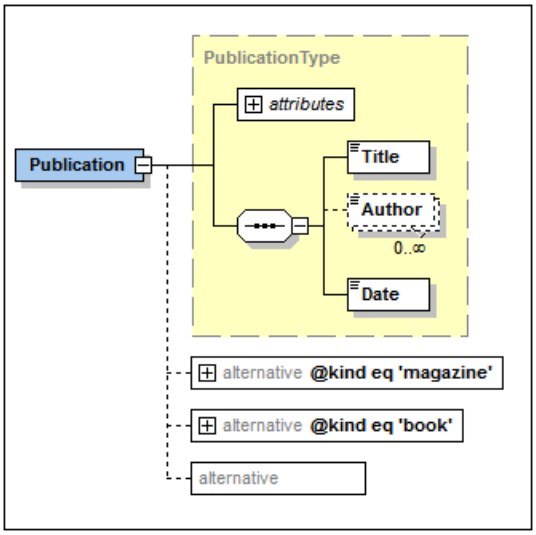
条件型割り当て

条件型割り当てはXSD 1.1 に新たに追加された機能で、要素に含まれる属性値や属性が存在するかによりXML ドキュメント内にある要素の型が決定されます。以下の様なXML ドキュメントを例に考えてみましょう:

```
<publication kind="magazine">
      ...
</publication>
```

スキーマでは、インスタンス要素の kind 属性値により、複数の publictaion 要素の型を指定することができます。スキーマでは、XSD 1.1 にて新たに追加された alternative 要素が使用されます。各 alternative 要素では、それぞれ異なる型が指定されます。

以下のスクリーンショットでは、test 属性を含む alternative 子要素（@kind eq 'magazine' と @kind eq 'book'）とともに Publication 要素が宣言されています。3番目の alternative 要素には test 属性が含まれておらず、（詳細入力ヘルパーで割り当てられており、図の中には表示されていない）xs:error の単純型が割り当てられており、必要なときに XML 検証エラーが返されます。



上の宣言を以下に示します：

```
<xs:complexType name="PublicationType">
	<xs:sequence>
		<xs:element name="Title" type="xs:string"/>
		<xs:element name="Author" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
		<xs:element name="Date" type="xs:gYear"/>
	</xs:sequence>
	<xs:attribute name="kind" type="xs:string"/>
</xs:complexType>

<xs:complexType name="MagazineType">
	<xs:complexContent>
		<xs:restriction base="PublicationType">
		<xs:sequence>
				<xs:element name="Title" type="xs:string"/>
				<xs:element name="Date" type="xs:gYear"/>
			</xs:sequence>
		</xs:restriction>
	</xs:complexContent>
</xs:complexType>

<xs:element name="Publication" type="PublicationType">
	<xs:alternative test="@kind eq 'magazine'" type="MagazineType"/>
	<xs:alternative test="@kind eq 'book'">
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="PublicationType">
					<xs:sequence>
```

```
                                        <xs:element name="ISBN"
type="xs:string"/>
                                        <xs:element name="Publisher"
type="xs:string"/>
                                    </xs:sequence>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:alternative>
                    <xs:alternative type="xs:error"/>
</xs:element>
```

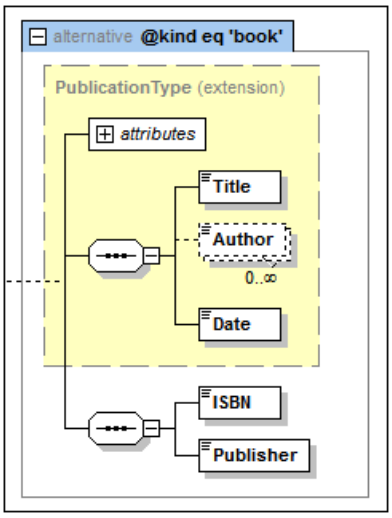
以下の点に注目してください:

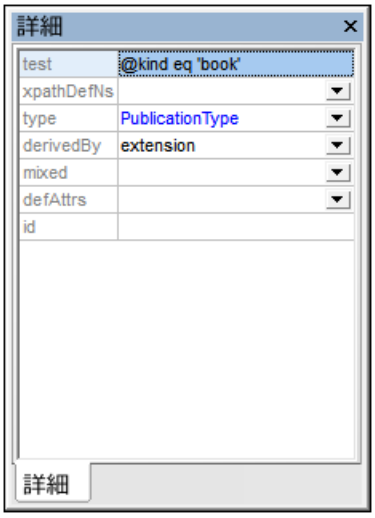
- 複数あるalternative 要素の中で、最初にtrue と評価されたalternative 要素が選択されるため、alternative 要素の順序は重要な意味を持ちます。コンテンツ モデル ビューでは、alternative 要素のボックスをドラッグすることで、その順序を変更することができます。
- Publication 要素には型 (PublicationType) が含まれていることに注目してください。どのalternative 要素も使用されない場合、この型がデフォルトの型として使用されます。上の例では、最初の 2 つのalternative がfalse を返した場合、xs:error 型のalternative 要素が使用されます。
- どのalternative 要素条件も満たされず、デフォルトの型も定義されていない場合、要素はanyType 型に割り当てられます。これにより XML コンテンツの整形式を保つことができる場合もあります。
- alternative 要素と単純型のxs:erroor はXSD 1.1 に新たに追加された要素です。

コンテンツ モデル ビューでの編集

要素のコンテキスト メニューから子としてalternative 型を要素宣言に追加することもできます (以下のスクリーンショットにあるコンテンツ モデルを参照)。

alternative 要素の型は詳細入力ヘルパーで指定されます。複合型の場合、それがalternative 要素の展開された長方形にて表示され、編集することができます (以下のスクリーンショットを参照)。単純型は図に表示されませんが、詳細入力ヘルパーの単純型タブにて定義することができます。





メモ: 詳細入力ヘルパーでは、XPath 条件式の名前空間をxpathDefaultNamespace 属性にて指定することができます。XPath デフォルト名前空間に関する詳細については、以下の記述を参照ください。

xpathDefaultNamespace の使用

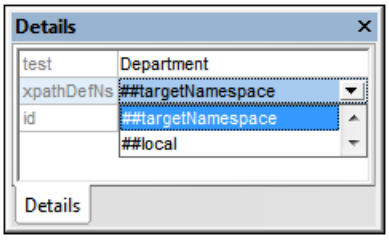
XML スキーマ ドキュメントで宣言されたデフォルト名前空間は、そのXML スキーマ ドキュメントのデフォルト名前空間で、スキーマ ドキュメント内の接頭辞のない要素名に対して適用されますが、スキーマ ドキュメント内の接頭辞のないXPath 条件式に対しては適用されません。

XSD 1.1 にて新たに追加された`xpathDefaultNamespace` 属性は、XPath 条件式にて接頭辞のない要素の名前空間を指定するために使用されます。

XPath デフォルト名前空間は、それらが宣言されたXML スキーマ要素をスコープとします。`xpathDefaultNamespace` 属性は、以下のXML Schema 1.1 要素内に記述することができます :

- `xs:schema`
- `xs:assert` と`xs:assertion`
- `xs:alternative`
- `xs:selector` と`xs:field` (同一性制約にて)

XSD 1.1 モードでは、`xs:schema` 上の`xpathDefaultNamespace` がスキーマ設定ダイアログ (**スキーマ設計 | スキーマ設定**) にてセットされます。上に示されるその他の要素の場合、`xpathDefaultNamespace` 属性が詳細入力ヘルパーにてセットされます (以下のスクリーンショットを参照)。



`xs:schema` 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、`xpathDefaultNamespace` 属性が許されている属性でオーバーライドすることができます (上のリストを参照ください)。

実際の名前空間を含める代わりに、`xpathDefaultNamespace` 属性には、以下のうちのどれかを含めることができます :

- `##targetNamespace`: スキーマのターゲット名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- `##defaultNamespace`: スキーマの名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- `##local`: XPath デフォルト名前空間は指定されません。

ドキュメントに対してXPath デフォルト名前空間が宣言されなかった場合、XPath 条件式における接頭辞なしの要素は名前空間を持たない要素として扱われます。

メモ: XPath デフォルト名前空間宣言は属性に対して適用されません。

オープン コンテンツ モデル

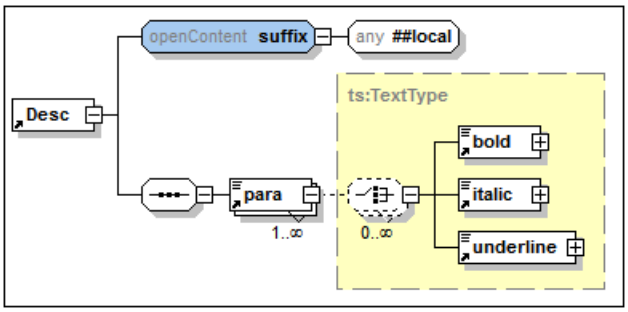
オープン コンテンツはXSD 1.1 に新たに追加された機能で、複合型にて宣言され、(i) コンテンツ モデルにて定義された要素の間に、または (ii) コンテンツ モデルで定義された要素の後に記述されるあらゆる種類の要素 (複合型のコンテンツ モデルでは定義されていない要素) を許可するために使用されます。

`openContent` 要素は複合型の子要素で、複合型のコンテンツ モデル前に一度出現します (以下のスクリーンショットを

参照)。

モード

openContent 要素には必須の mode 属性が含まれており、値には interleave、suffix、または none のいずれかが使用されます (以下のスクリーンショットを参照)。デフォルトでは interleave が使用されます。



これら値が持つ意味を以下に記します:

- mode="interleave" または mode="suffix" となっている場合、最低ならびに最大出現数が指定されていないワイルドカード要素コンテンツ (xs:any) が必要になります。これにより、任意の数の未定義要素 (ワイルドカード) が許されることになります。
- モードが interleave の場合、コンテンツモデルにて定義された個々の要素の前または後に任意の数の未定義要素が出現できるようになります。
- モードが suffix の場合、コンテンツモデルにて定義されている最後の要素の後に任意の数の未定義要素が出現できるようになります。
- モードが none の場合、未定義の要素 (xs:any 子要素) の出現は許されず、コンテンツモデルもオープンとはなりません。none 値は、スキーマ全体をスコープにする defaultOpenContent 要素をオーバーライドするために使用されます。

コンテンツモデルビューでは、複合型の子要素として (コンテキストメニューの **「子の追加」** から) オープンコンテンツが追加されます。図の中にある openContent ボックスをダブルクリックして値 (interleave、suffix、または none) を選択するか、詳細入力ヘルパーからモードの値を選択してください。

ワイルドカード (xs:any) プロパティ

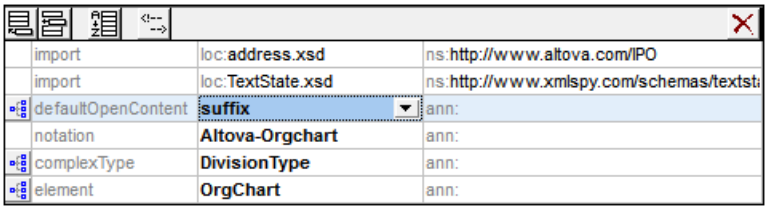
ワイルドカードプロパティは、ワイルドカードの詳細入力ヘルパーにて指定されます。図の中でワイルドカードを選択し、詳細入力ヘルパーにてプロパティ値を入力してください。

デフォルト オープン コンテンツ

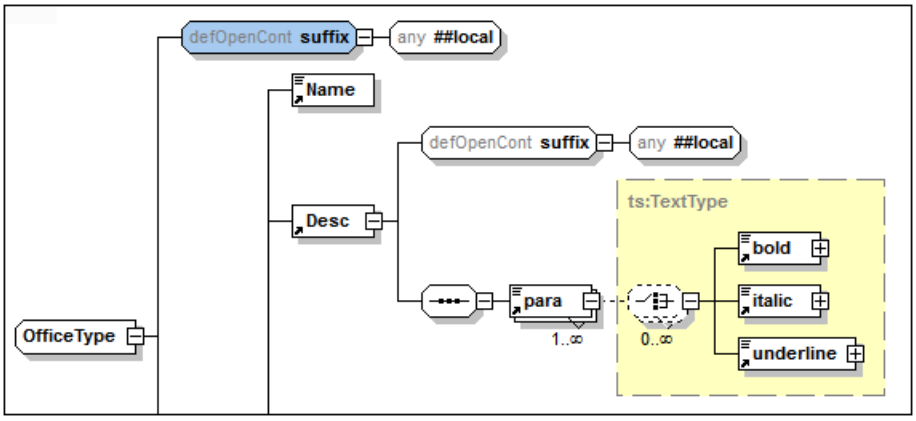
defaultOpenContent 要素は XSD 1.1 にて新たに追加された機能で、混合または子要素だけが許されている複合型に追加することのできる未定義の要素を指定するために使用されます。同じく XSD 1.1 で新たに追加された openContent 要素に似ていますが、openContent 要素が単一の複合型に対して適用されるのに対して、defaultOpenContent 要素はスキーマ内にあるすべての複合型に対して適用されます。

defaultOpenContent 要素はグローバル要素で、ドキュメント内 (インクルード、インポート、再定義の後、そしてコンテンツ定義の前) に一度だけ出現することができます (以下のスクリーンショットを参照)。要素には interleave または

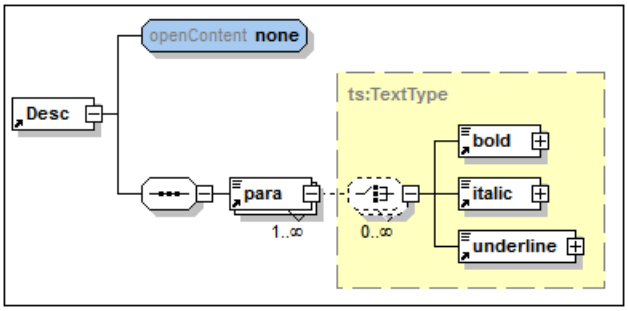
suffix という値を含めることのできるmode 属性が含まれており、デフォルトではinterleave 値が使用されます。



defaultOpenContent 要素にはopenContent 要素の定義と同様に、コンテンツ モデル ビューで編集することのできるコンテンツ モデルが含まれています（上記参照）。一旦 defaultOpenContent 要素が宣言されると、スキーマ内にあるすべての複合型に対して自動的に適用され、各複合型のコンテンツ モデルに表示されます。以下のスクリーンショットでは、defaultOpenContent がOfficeType とDesc という複合型に対して自動的に適用されているのが確認できます。

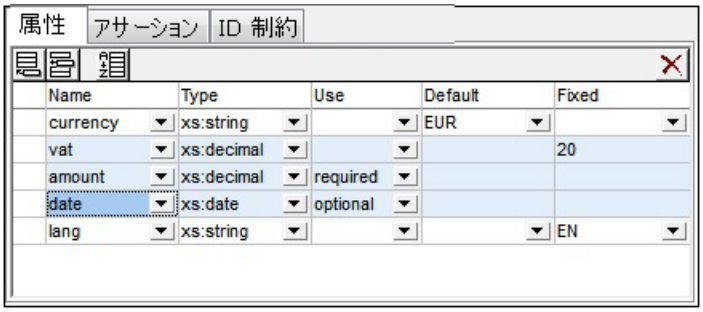


特定の複合型に対してdefaultOpenContent 要素をオーバーライドするには、openContent 子要素をその複合型へ追加してください。以下のスクリーンショットでは、defaultOpenContent が適用されたDesc 要素以下にopenContent 子要素が定義されており、defaultOpenContent 要素がオーバーライドされています。



2.3.4 属性、アサーション、同一性制約

属性 /アサーション/同一性制約 (AAIDC) ペインは、スキーマ概要メインペインならびにコンテンツ モデル ビューの下部に表示されます（以下のスクリーンショットを参照）。ペインとそのタブは固定されていますが、コンテンツ モデル ビューでは各タブの表示内容を（AAIDC タブではなく）コンテンツ モデル ビューに表示して、図の中で直接編集することができます。これら 3 つのタブの内容を図の中に表示するように指定すると、AAIDC ペインが表示されなくなります。



AAIDC ペイン図の中における表示は、スキーマ表示設定ダイアログ（**スキーマ設計 | 表示設定 | 要素タブ**）または（以下に示される）スキーマ設計ツールバーのアイコンから切り替えることができます。

- **属性を図に表示**: コンテンツモデルビューが表示されている際にスキーマ設計ツールバーで有効になります。属性の表示を図または属性タブで行うか指定します。
- **アサーションを図に表示**: コンテンツモデルビューが表示されている際にスキーマ設計ツールバーで有効になります。アサーションの表示を図またはアサーションタブで行うか指定します。
- **制約を図に表示**: コンテンツモデルビューが表示されている際にスキーマ設計ツールバーで有効になります。同一性制約の表示を図または同一性制約タブで行うか指定します。

タブの使用

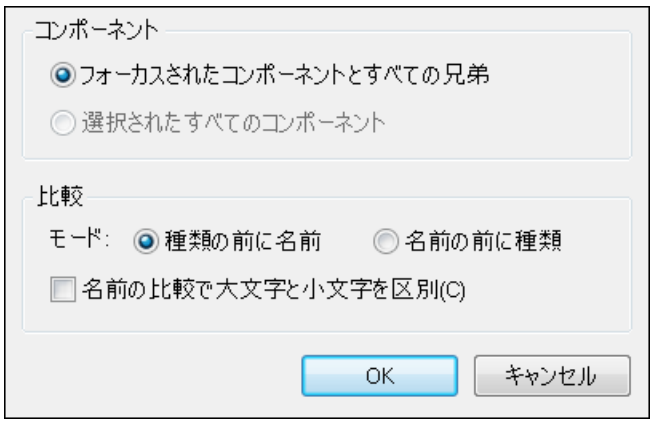
AAIDC ペイン内部のタブは、スキーマ概要またはコンテンツモデルビューにて選択されているコンポーネントの種類に対応しているタブが有効になります。例えば複合型に対して属性を追加することはできるため、メインペインにて複合型が選択されている時に属性タブが有効になります（タブが有効になるとは、そのタブの中でなんらかの操作ができることを意味します）。

各タブの使用方法については、このセクション以下にあるサブセクションを参照ください：

- 属性、属性グループ 属性ワイルドカード
- アサーション
- 同一性制約

属性と同一性制約の並べ替え

各タブのツールバーにある並べ替えアイコンをクリックすることで、属性と同一性制約の並べ替えを行うことができます。コンポーネントの並べ替えダイアログでは、選択されたコンポーネントとその兄弟、または選択されたコンポーネントだけを並べ替えの対象とするかを選択することができます。例えば上のスクリーンショットでは、3つの属性が選択（ハイライト）されています。**「Shift」**キーを押下しながらクリックすることで範囲の選択を、**「Ctrl」**キーを押下しながらクリックすることで、コンポーネントを選択対象に加えることができます。



コンポーネントのセットを選択した後には最初にアルファベット順で並べ替えを行い、その次に種類による並べ替えを行うか (種類の前に名前) またはその逆 (名前の前に種類) という、並べ替えの優先順位を選択することができます。並べ替えの結果はスキーマ ドキュメントのテキストへ直ちに反映されます。

属性、属性グループ、属性ワイルドカード

属性 アサーション同一性制約 (AAIDC) ペインの属性タブでは、以下の操作を行うことができます :

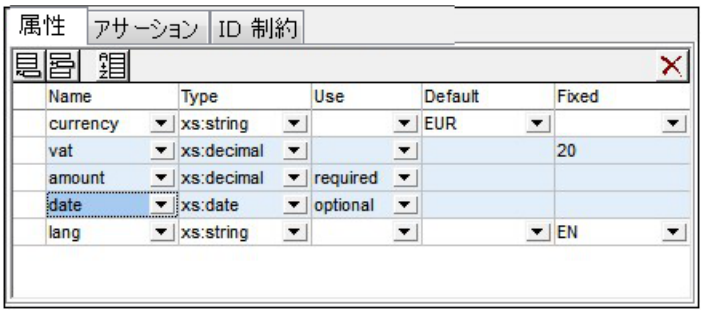
- 選択された複合型にて、ローカルの属性を宣言
- 選択された複合型で使用する属性グループの参照
- 選択された複合型にて属性ワイルドカードを定義

メモ: AAIDC ペインではなく、図の中に属性を表示するオプションを選択した場合 (**「スキーマ設計 | 表示設定」**)、属性、属性グループ参照、属性ワイルドカードのプロパティを、図の中、そして詳細入力ヘルパーにて編集することができます。

属性を図に表示: コンテンツ モデル ビューが表示されている際にスキーマ設計ツールバーで有効になります。属性の表示を図または属性タブで行うか指定します。

属性

属性 アサーション同一性制約 (AAIDC) ペインの属性タブでは、要素や複合型のローカル属性、または属性グループを構成する属性を宣言することができます (以下のスクリーンショットを参照)。



以下の操作により属性を作成することができます :

1. スキーマ概要にて複合型または属性を作成するための属性グループを選択します。
2. 属性タブの左上にある **追加** または **挿入** アイコンをクリックして、 **属性** を選択します。
3. 属性に対して作成された行にて属性の詳細 (Name、Type、Use、Default、Fixed の値) を入力します。 Name プロパティは必須の項目で、Use のデフォルト値はオプションです。データ型 (Type) とDefault/Fixed 値はオプションです。

メモ: 属性グループへの属性追加はスキーマ概要でのみ行うことができますが、複合型への属性追加はスキーマ概要とコンテンツ モデル ビューの両方で行うことができます。

デフォルト値と固定値

属性宣言で宣言されたデフォルト値 (Default) または固定値 (Fixed) は、インスタンス ドキュメント内にその属性が存在しない場合に (インスタンス ドキュメントに対して) 適用されます。指定できるのはデフォルト値と固定値の一方で、両方を指定することはできません (上のスクリーンショットを参照)。デフォルト値または固定値は属性のデータ型に対して妥当な値である必要があります。Use に `required` がセットされている場合、デフォルト値も固定値も定義することはできません。

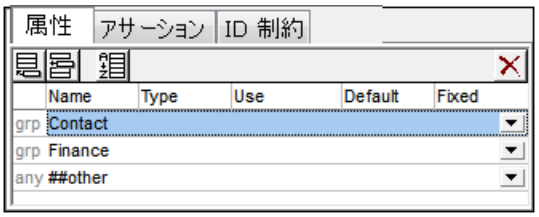
以下の点に留意してください:

- **デフォルト値**: 属性が存在しない時だけデフォルト値は挿入されます。属性が存在しており、値が与えられている場合、デフォルト値は挿入されません。インスタンス ドキュメント内の値が妥当ではない場合、エラーが返されます。
- **固定値**: 固定値は属性が存在しない場合だけではなく、インスタンス ドキュメントの値が属性の宣言にて指定された固定値と異なる場合にも挿入されます。

メモ: デフォルト値と固定値はローカルならびにグローバル属性に対しても指定することができます。ローカル属性の場合は AAIDC ペインの属性タブと詳細入力ヘルパーの両方で定義することができ、グローバル属性の場合、詳細入力ヘルパーにて定義することができます。

属性グループと参照

グローバルの属性グループが宣言されると、この属性グループに対する参照を複合型の定義内で追加することができます。スキーマ概要またはコンテンツ モデル ビューにて複合型を選択し、AAIDC ペインの属性タブ左上にある **追加** または **挿入** アイコンをクリックして、 **属性グループ** を選択してください。作成された属性グループ行にて参照される属性グループの名前を入力してください (以下のスクリーンショットには 2 つの属性グループ参照が含まれています)。複数の属性グループを追加することができます。

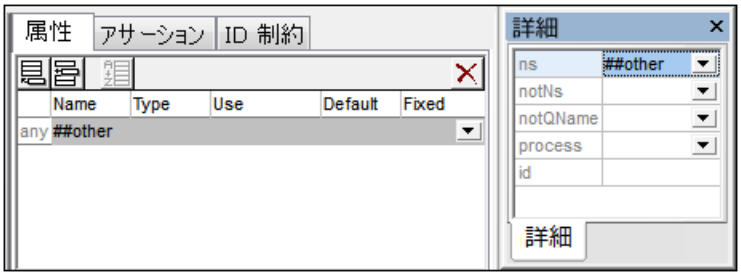


属性タブにて属性グループが選択されると、そのプロパティを詳細入力ヘルパーにて編集することができるようになります。

属性ワイルドカード：anyAttribute

属性ワイルドカードは、任意の属性を要素上で使用するのを許可するため複合型へ追加されます。属性ワイルドカードは単一の`anyAttribute` 要素により定義され、インスタンス ドキュメント内にある要素にて指定された名前空間から得られる任意の数の属性が出現するのを許可します。

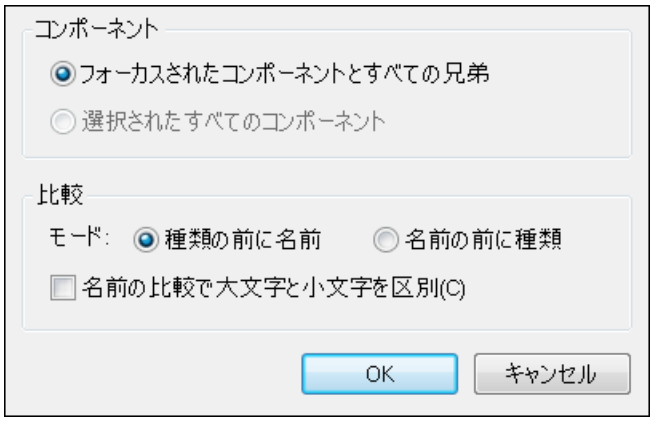
スキーマ概要またはコンテンツ モデル ビューにて複合型コンポーネントを選択し、AAIDC ペインの属性タブ左上にある追加または挿入アイコンをクリックし、**「任意属性」**を選択することで属性ワイルドカードは作成されます。属性ワイルドカード`anyAttribute` に対する行が作成されます（以下のスクリーンショットを参照）。



属性タブでは、`anyAttribute` の`namespace` プロパティをセットすることができます。その他のプロパティは、属性タブにて属性ワイルドカードが選択されている状態で詳細入力ヘルパーにてセットすることができます（上のスクリーンショットを参照）。`notNamespace` ならびに`notQName` プロパティはXSD 1.1 の機能で、XSD 1.0 モードでは利用できない点に注意してください。

属性と属性グループの並べ替え

タブのツールバーにある **「並べ替え」**アイコンをクリックすることで、属性タブに表示されている属性や属性グループを並べ替えることができます。コンポーネントの並べ替えダイアログでは、単一のコンポーネントとその兄弟コンポーネント、または選択されているコンポーネントのセットを並べ替えの対象とすることができます（以下のスクリーンショットを参照）。**「Shift」**または**「Ctrl」**キーを押下しながらコンポーネントをクリックすることで、複数のコンポーネントを選択することができます。



並べ替えを行う範囲を選択すると、属性すべてと属性グループをアルファベット順に並べ替え（種類の前に名前）、または属性と属性グループとアルファベット順で別々に並べ替えることができます（名前の前に種類）。

並べ替えの順序はスキーマドキュメントのテキストに直ちに反映されます。

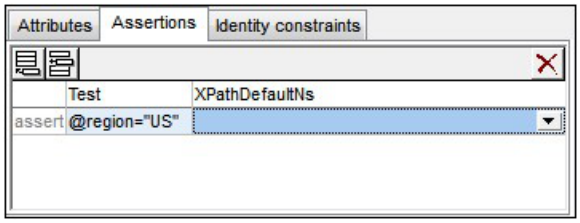
メモ: 属性ワイルドカードは常に複合型宣言の末尾に出現する必要があり、複合型の定義に対して属性ワイルドカードはそれぞれ 1 つしか許されていないため、並べ替えの対象に含まれません。

アサーション

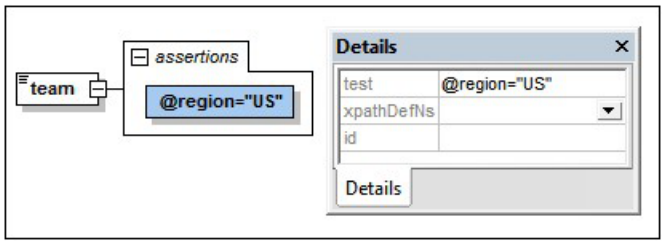
このセクションでは**複合型で使われるアサーション**について説明します。このようなアサーションは `xs:assert` 要素にて定義され、複合型の妥当性制約に使用されます（アサーションには `xs:assertion` 要素にて定義され、単純型のファセット入力ヘルパーにて作成ならびに編集されるアサーションも存在します）。

スキーマビューでは、以下の場所で複合型のアサーションを作成し、編集することができます：

- **スキーマ概要にて**: 属性 / アサーション / 同一性制約（AAIDC）ペインのアサーションタブにて（以下のスクリーンショットを参照）。



- **コンテンツ モデル ビューにて**: アサーションは図の中に表示されたアサーションタブにて編集することができます（以下のスクリーンショットを参照）。図の表示オプションを選択するには、スキーマデザインツールバーにある**「アサーションを図に表示」アイコン**をクリックしてください。複合型や複合型の要素にあるアサーションのボックスを選択し、詳細入力ヘルパーにてアサーションの定義を編集してください。



アサーションを図に表示: コンテンツ モデル ビューが表示されている際に選択可能になります。アサーションの表示を図またはアサーションタブで行うか指定します。

メモ: アサーションは XSD 1.1 に搭載されている機能で、アサーションの編集機能は XSD 1.1 モードでのみ有効になります。

アサーションのスコープ

アサーションの制約を定義するために使用される XPath 条件式は、定義されている複合型のスコープ内にある必要があります。XPath 条件式が特定のノードへアクセスする必要がある場合、そのノードの祖先ノードにてアサーションを定義する必要があります。

アサーションの追加と削除

複合型には複数のアサーションを含めることができます。インスタンス ドキュメント内にある要素に対してすべてのアサーションの XPath 条件式が `true` と評価されることで、ドキュメントも妥当なものと評価されます。以下の操作により、複合型に対してアサーションを追加することができます：

- **スキーマ概要にて**：複合型を選択し、AAIDC ペインにあるアサーション タブの左上にある**追加**または**挿入**アイコンをクリックします（上のスクリーンショットを参照）。追加することのできるアサーションの数に制限はありません。アサーションを選択し、タブの右上にある**削除**アイコンをクリックすることで、そのアサーションを削除することができます。
- **コンテンツ モデル ビューにて**（上のスクリーンショットを参照）：複合型を右クリックして、**「子の追加 | アサーション」**を選択します。図に表示されている既存のアサーションを右クリックして、コンテキスト メニューから**「追加 | アサーション」**または**「挿入 | アサーション」**を選択することもできます。追加することのできるアサーションの数に制限はありません。アサーションを選択して、**「Delete」**キーを押下することで、そのアサーションを削除することができます。

アサーションの XPath 条件式を定義する

複合型アサーションの XPath 条件式により、インスタンス ドキュメント内にある複合型要素に対して適用される妥当性制約が定義されます。例えば上のスクリーンショットでは、複合型要素の `team` とアサーションの XPath 条件式 `@region="US"` が定義されています。XML スキーマ ドキュメントで、アサーションは以下のように記述されます：

```
<xs:assert test='@region="US"'/>
```

アサーションにより、インスタンス ドキュメント内にある `team` 要素には、`US` という値を持った `region` 属性を含めなければならなくなります。この条件が満たされない場合、ドキュメントの検証に失敗します。

以下の点に留意してください：

- XPath 条件式は XPath 2.0 に従ったかたちで記述される必要があります。
- XPath 条件式で評価されるノードは、アサーションのスコープ内にある必要があります（上記を参照ください）。
- 条件式が `boolean` の `true/false` を評価しない場合、返された値が `boolean` 値に変換されます。空でないシーケンスは `true` に変換され、空のシーケンスは `false` に変換されます。
- 条件式における構文エラーは赤色で示されます。コンテキスト エラーは表示されません。例えば XPath 条件式にてある属性が評価され、その属性がスキーマ内部で定義されていなかったとしても、そのエラーは表示されません。

アサーション メッセージ

アサーションの詳細がこの定義と共に提供される場合とても役に立ちます。XML インスタンス ドキュメントが検証される際、アサーションが満たされていない場合、適切な メッセージが表示されます。XML スキーマ仕様は、このようなメッセージのために対応をしないため、 XMLSpy は、アサーションの定義と共に XML インスタンス ドキュメントの検証内で使用される Altova xml-schema-extensions 名前空間 `http://www.altova.com/xml-schema-extensions`（または他の名前空間）内でメッセージを許可します。例えば：

```
<xs:assert test="count(//MyNode) ge 1" altova:message="There must be at
least one MyNode element"/> or
<xs:assertion test="count(//MyNode) ge 1" altova:message="There must be at
least one MyNode element"/>
```

アサーション内で指定された制約が満たされない場合、XMLSpy の検証エンジンは、検証 エラー メッセージと共にヒントとしてのアサーションと関連したメッセージを表示します。 バリデーターは、`assert/@message` 属性の値または `message` 属性が存在する名前空間に関わらず、アサーション/`@message` 属性 を報告します。ですが、スキーマ ビュー内では、xml-schema-extensions 名前空間にある `message` 属性 のみを編集することができます。他の名前空間

内のmessage 属性 を編集するには、テキストビューを使用します。

詳細に関しては、アサーションメッセージを参照してください。

---

xpathDefaultNamespace の使用

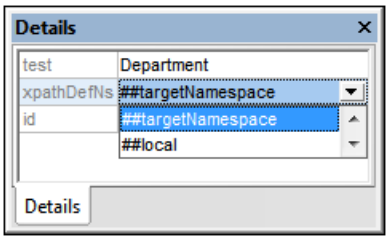
XML スキーマ ドキュメントで宣言されたデフォルト名前空間は、そのXML スキーマ ドキュメントのデフォルト名前空間で、スキーマ ドキュメント内の接頭辞のない要素名に対して適用されますが、スキーマ ドキュメント内の接頭辞のないXPath 条件式に対しては適用されません。

XSD 1.1 にて新たに追加されたxpathDefaultNamespace 属性は、XPath 条件式にて接頭辞のない要素の名前空間を指定するために使用されます。

XPath デフォルト名前空間は、それらが宣言されたXML スキーマ要素をスコープとします。
xpathDefaultNamespace 属性は、以下のXML Schema 1.1 要素内に記述することができます :

- xs:schema
- xs:assert とxs:assertion
- xs:alternative
- xs:selector とxs:field (同一性制約にて)

XSD 1.1 モードでは、xs:schema 上のxpathDefaultNamespace がスキーマ設定ダイアログ (**スキーマ設計 | スキーマ設定**) にてセットされます。上に示されるその他の要素の場合、xpathDefaultNamespace 属性が詳細入力ヘルパーにてセットされます (以下のスクリーンショットを参照)。



xs:schema 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、xpathDefaultNamespace 属性が許されている属性でオーバーライドすることができます (上のリストを参照ください)。

実際の名前空間を含める代わりに、xpathDefaultNamespace 属性には、以下のうちのどれかを含めることができます :

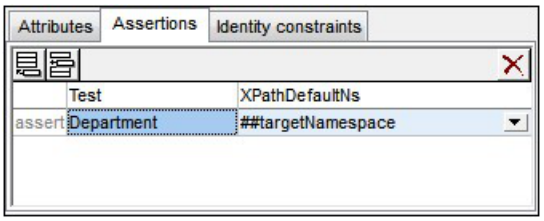
- ##targetNamespace: スキーマのターゲット名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- ##defaultNamespace: スキーマの名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- ##local: XPath デフォルト名前空間は指定されません。

ドキュメントに対してXPath デフォルト名前空間が宣言されなかった場合、XPath 条件式における接頭辞なしの要素は名前空間を持たない要素として扱われます。

メモ: XPath デフォルト名前空間宣言は属性に対して適用されません。

アサーション内にあるXPath 条件式アサーションの定義にてXPath デフォルト名前空間を定義することもできます。属性 / アサーション/同一性制約 (AAIDC) ペインのアサーション タブにて、XPathDefaultNs フィールドのドロップダウン リストか

ら目的のキーワードを選択してください。



選択された名前空間が、そのアサーションのスコープとなります。

同一性制約

同一性制約 (DC) はグローバル要素宣言にて定義されます。同一性制約によりノードの一意性が指定され、ユニークなノード間の正確な参照を行うことができます。

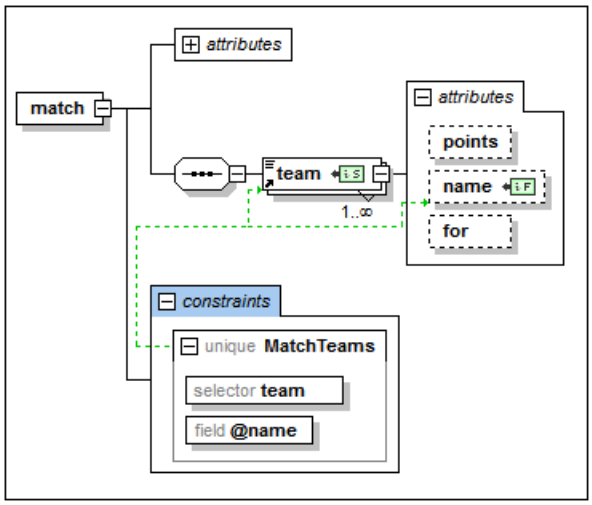
宣言のメカニズム

同一性制約 (`unique`、`key`、`keyref`) を定義するために以下のメカニズムを使用することができます：

- スキーマ概要に表示されたグローバル要素にて同一性制約を宣言することができます。グローバル要素を選択し、属性 / アサーション / 同一性制約 (AAIDC) ペインにある同一性制約タブをクリックします (以下のスクリーンショットを参照)。

同一性制約タブにある挿入または追加アイコンにより、同一性制約 (`unique`、`key`、`keyref`) を追加します。選択された同一性制約に対して `field` を追加することもできます。選択された `field` や同一性制約を削除するには、削除アイコンを使用してください。

- グローバル要素のコンテンツモデルビューでは、グローバル要素やローカルの子孫要素にて同一性制約を定義することができます。このビューでは、同一性制約のタブと図の中に表示された要素の制約ボックスで同一性制約を編集することができます (以下のスクリーンショットでは、`match` 要素に `team` セレクターが含まれるというユニークな制約が示されています)。**「スキーマ設計 | 表示設定」**により表示されるスキーマ表示設定にて、またはスキーマ設計ツールバーにある **「制約を図に表示」**アイコンをクリックすることで、表示方法を切り替えることができます。図に示された同一性制約に対してドラッグ アンド ドロップ編集機能を使用することもできます。



同一性制約 (unique、key、keyref) を図の中に追加するには、制約を行う要素を右クリックして、コンテキストメニューから **子の追加 | <同一性制約の種類>** を選択してください。同一性制約が図の中で選択されていると、フィールド アイテムが選択可能になります。 **Delete** キーを押下することで、選択された field または同一性制約を削除することができます。

以下にあるいずれかの方法で selector ならびに field ボックスに XPath 条件式を入力することができます: (i) キーボードから入力する、(i) selector または field ボックスをクリックすると表示されるドロップダウン リストから目的のノードを選択する、または (ii) ターゲット ノードを selector または field ボックスへドラッグして、ボックスがハイライトされたところでドロップする (XPath 条件式が自動的に作成されます)。

メモ: コンポーネント入力ヘルパーにある同一性制約のタブではスキーマ内にあるすべての同一性制約の概要 を確認することができます。詳細については、以下のコンポーネント入力ヘルパーを参照ください。

同一性制約アイコン

制約を図に表示: コンテンツ モデル ビューが表示されている際にスキーマ設計ツールバーで有効になります。同一性制約の表示を図または同一性制約タブで行うか指定します。

同一性制約の視覚化: コンテンツ モデル ビューにて選択可能になります。同一性制約の情報の表示 / 非表示を切り替えます。

セレクター ノード、フィールド ノード: ノードの図にて表示され、これら 2 つのアイコンにより、selector ならびに field の XPath 条件式により選択されたノードがそれぞれ表示されます。

同一性制約の可視化

同一性制約の可視化アイコン が有効になっている場合、同一性制約の情報が図の中で可視化されます。可視化が有効になっている場合、selector と field の XPath 条件式により選択されたノードがボックス内のアイコンにより示され、緑色の線によりセレクターとフィールドのボックスが接続されます (上のスクリーンショットを参照ください)。

同一性制約の可視化アイコンにより、スキーマビューにおける同一性制約の検証機能も切り替えられます。XPath 条件式または同一性制約が間違っている場合、エラーが赤色で、警告がオレンジ色で示されます。XML スキーマドキュメントの検証を行うと、エラーと警告がメッセージ ウインドウに表示されます。

同一性制約の可視化アイコン を無効にすることで、検証を無効にすることができます。

XML の内容

このセクションの下部に示される同一性制約の例は、以下の妥当なインスタンス ドキュメントをベースにしています：

```
<results xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="Scores.xsd">
   <!---- Groups ----->
   <group id="A">
         <team name="Brazil"/>
         <team name="Germany"/>
         <team name="Italy"/>
         <team name="Holland"/>
   </group>
   <group id="B">
         <team name="Argentina"/>
         <team name="France"/>
         <team name="England"/>
         <team name="Spain"/>
   </group>
   <!---- matches ----->
   <match group="A" date="2012-06-12" location="Munich">
         <team name="Brazil" for="2" points="3"/>
         <team name="Germany" for="1" points="0"/>
   </match>
   <match group="A" date="2012-06-12" location="Frankfurt">
         <team name="Italy" for="2" points="1"/>
         <team name="Holland" for="2" points="1"/>
   </match>
   <match group="B" date="2012-06-13" location="Munich">
         <team name="Argentina" for="2" points="3"/>
         <team name="France" for="0" points="0"/>
   </match>
   <match group="B" date="2012-06-13" location="Berlin">
         <team name="England" for="0" points="1"/>
         <team name="Spain" for="0" points="1"/>
   </match>
</results>
```

ユニーク制約 (unique)

ユニーク制約により、要素または属性 (要素や属性のセット) の値が定義されたスコープ内でユニーク (一意) でなければならないと指定されます。なんらかのリーグ戦を表している上のXML テキストでは、同じ名前のチーム同士が試合を行わないような制約が必要となるため、各 `match` 要素のスコープにて `team/@name` ノードの値がユニークであるような制約を作成します。以下の操作により制約を作成します :

1. スキーマ概要にて `match` 要素を選択します。`match` 要素が同一性制約のスコープとなります。
2. 同一性制約タブの左上にある **「追加」** または **「挿入」** アイコンをクリックして、メニューから **「ユニーク」** をクリックします。ユニーク制約の行が追加されます (以下のスクリーンショットを参照)。

3. 同一性制約の名前を入力します (上のスクリーンショットでは、`MatchTeams` という名前が与えられています)。
4. `team` 要素を選択するXPath 条件式を `Selector` 列に入力します。`match` 要素がコンテキスト ノードだということに注意してください。これで `team` 要素が同一性制約のセレクターとなり、ユニーク制約が適用されるノードとなります。
5. ユニークでなければならない `@name` ノードをField 列に入力します。ユニーク制約が入力されたノードの値に対して適用されます。

上記のユニーク制約により、各 `match` 要素のスコープ内にてすべての `team` 要素にユニークな `@name` 属性値が必要になります。

更にフィールドを追加して、ユニーク制約をチェックすることもできます。例えば `result` 要素にて定義されたユニーク制約により、ユニークな日付と場所の組み合わせがすべての `match` に含まれていることをチェックすることができます (同じ日付に同じ場所で複数の試合を行うことができなくなります)。ユニーク制約により、各 `match` 要素 (セレクター) に対して `@date` と `@location` の値を組み合わせた値が `result` 要素のスコープ内部でユニークとなる必要があります。

上記の方法により `result` 要素に対してユニーク制約を定義します。`match` がセレクターとなり、`@date` と `@location` がフィールドとなります (以下のスクリーンショットを参照)。追加アイコンをクリックして **「フィールド」** を選択することで、2番目のフィールドが追加されます。

メモ: 同一性制約タブのRefer 列はキー参照 (`keyref`) 制約でのみ有効になります。

キー制約 (key)

キー制約により、以下の制約が指定されます : (i) 要素や属性 (または属性やスキーマのセット) の値が定義されたスコープ内でユニークであり、(ii) これらフィールド要素や属性がインスタンスXML ドキュメントに含まれている (従って省略可能な要

素や属性をキー制約のフィールドとして選択すべきではありません)。キー制約により、ユニーク制約により与えられる制約に加え、指定されたフィールド要素/属性をXML ドキュメント内に含める必要があります。

既に登場した最初のユニーク制約と同様に、match 要素にて定義されたキー制約を以下のスクリーンショットに示します:

このキー制約により、各 match 要素のスコープ内部ですべての team 要素にユニークな @name 値が必要になります。更に、@name 属性がmatch/team 要素上に存在しなければならないと指定されます。

メモ: 同一性制約タブのRefer 列はキー参照 (keyref) 制約でのみ有効になります。

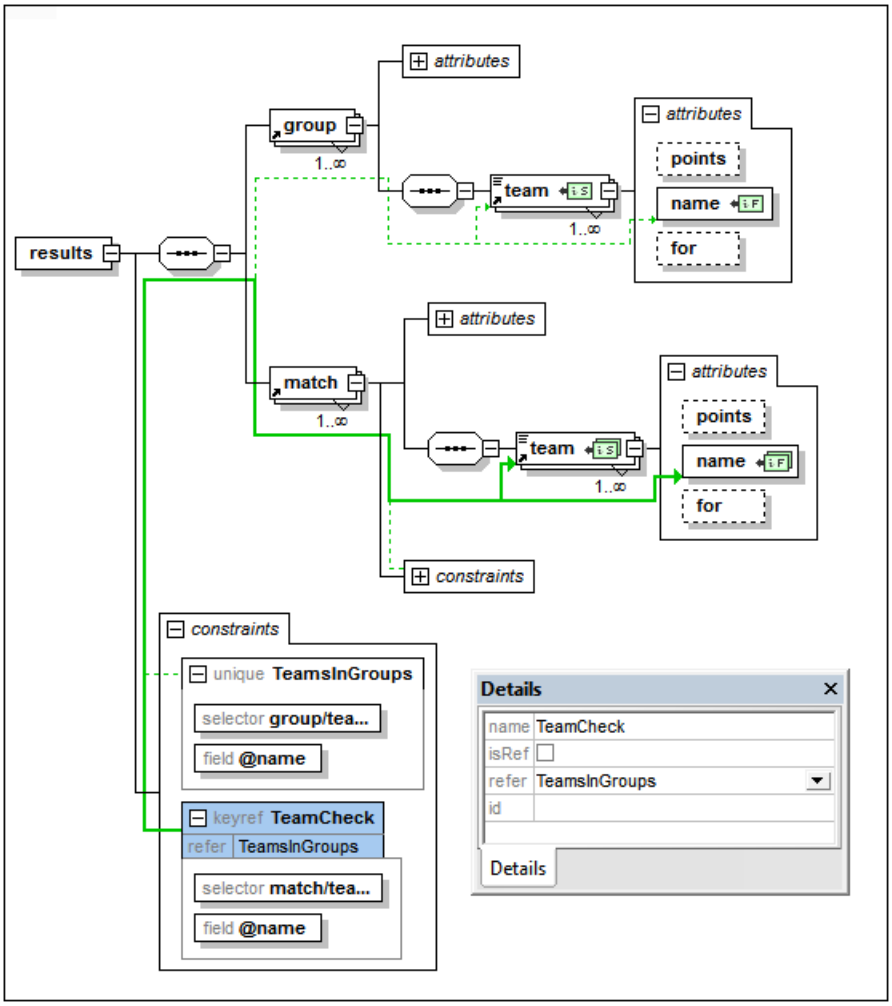
キー参照 (keyref)

キー参照により、インスタンス ドキュメント内にある値のセットがチェックされます。例えば上に示されるXML のテキストでは、キー参照を使うことにより同じグループに含まれているチーム同士で試合が行われるかをチェックすることができます。条件が満たされない場合、XML ドキュメントの検証に失敗します。

最初にユニーク制約またはキー制約を作成します。以下のスクリーンショットではTeamInGroups という名前のユニーク制約がresults 要素に対して作成されています。この制約により、group 要素内にある各 team にて、ユニークな @name 属性値が必要になります。

次に、match の子要素であるteam を選択し、@name の属性値がTeamInGroups により返された値に含まれるかどうかをチェックする、TeamCheck という名前のキー参照 (keyref) を作成します。

以下のスクリーンショットでは、(水色でハイライトされている)このキー参照と、詳細入力ヘルパーの内容が示されています。選択された同一性制約との関係は緑色の実線で示されており、選択されていない同一性制約は緑色の破線により示されています。それぞれの同一性制約にてselector とfield のXPath 条件式にて選択されたノードが、それぞれ と アイコンにより示されます。ノードが縮退されると、関係を表す線が破線状態で表示されます。



xpathDefaultNamespace の使用

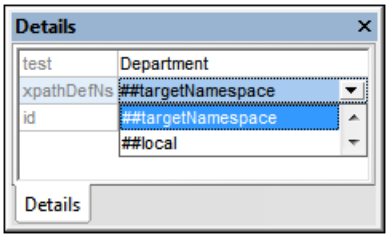
XML スキーマ ドキュメントで宣言されたデフォルト名前空間は、そのXML スキーマ ドキュメントのデフォルト名前空間で、スキーマ ドキュメント内の接頭辞のない要素名に対して適用されますが、スキーマ ドキュメント内の接頭辞のないXPath 条件式に対しては適用されません。

XSD 1.1 にて新たに追加された`xpathDefaultNamespace` 属性は、XPath 条件式にて接頭辞のない要素の名前空間を指定するために使用されます。

XPath デフォルト名前空間は、それらが宣言されたXML スキーマ要素をスコープとします。`xpathDefaultNamespace` 属性は、以下のXML Schema 1.1 要素内に記述することができます：

- `xs:schema`
- `xs:assert` と`xs:assertion`
- `xs:alternative`
- `xs:selector` と`xs:field` (同一性制約にて)

XSD 1.1 モードでは、`xs:schema` 上の`xpathDefaultNamespace` がスキーマ設定ダイアログ (**スキーマ設計 | スキーマ設定**) にてセットされます。上に示されるその他の要素の場合、`xpathDefaultNamespace` 属性が詳細入力ヘルパーにてセットされます (以下のスクリーンショットを参照)。



`xs:schema` 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、`xpathDefaultNamespace` 属性が許されている属性でオーバーライドすることができます (上のリストを参照ください)。

実際の名前空間を含める代わりに、`xpathDefaultNamespace` 属性には、以下のうちのどれかを含めることができます:

- `##targetNamespace`: スキーマのターゲット名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- `##defaultNamespace`: スキーマの名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- `##local`: XPath デフォルト名前空間は指定されません。

ドキュメントに対してXPath デフォルト名前空間が宣言されなかった場合、XPath 条件式における接頭辞なしの要素は名前空間を持たない要素として扱われます。

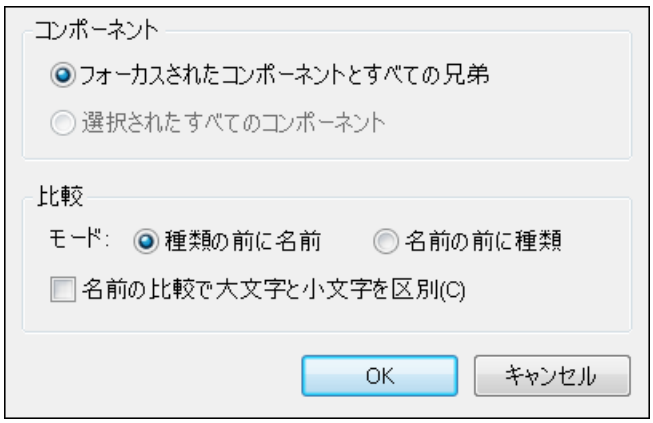
メモ: XPath デフォルト名前空間宣言は属性に対して適用されません。

同一性制約の ID

同一性制約とそのセレクター、フィールドに対してID を与えることができます。ID を割り当てるには必要なコンポーネントを選択し、詳細入力ヘルパーのID 行にてID を入力してください。

同一性制約の並べ替え

同一性制約タブのツールバーにある **「並べ替え」** アイコンをクリックすることで、同一性制約の並べ替えを行うことができます。コンポーネントの並べ替えダイアログでは、選択されたコンポーネントとその兄弟、または選択されたコンポーネントのセットを並べ替えの対象とすることができます。 **「Shift」** キーを押下しながらクリックすることで範囲の選択を、 **「Ctrl」** キーを押下しながらクリックすることで新たなコンポーネントを選択対象に含めることができます。



並べ替えの範囲を選択した後には、すべての対象をアルファベット順で並べ替えるのか (種類の前に名前)、または同一性制約の種類ごと (ユニーク制約、キー制約、キー参照の順) でアルファベット順に並べ替えるのかを選択することができます。

並べ替えの順序はスキーマのテキストに反映されます。

2.3.5 スキーマ ビューにおける入力ヘルパー

スキーマビューにて 3 つの入力ヘルパーが表示されます。このセクション以下にあるサブセクションでは、これら入力ヘルパーの詳細について説明されます :

- コンポーネント入力ヘルパー
- 詳細入力ヘルパー
- ファセット入力ヘルパー

表示される入力ヘルパーはスキーマ概要とコンテンツ モデル ビューで変わらず、スキーマ コンポーネントの定義を視覚的に追加ならびに編集することができます。入力ヘルパーからコンポーネントをドラッグすることができるほか、デザイン上でコンポーネントを選択して、そのプロパティを入力ヘルパーで定義することができます。

コンポーネント

スキーマビューにおけるコンポーネント入力ヘルパーでは、3 つの操作を行うことができます :

- コンポーネントの種類と名前空間により、ツリー内にあるグローバル コンポーネントを管理する (以下のスクリーンショットを参照) : これにより、名前空間に従うかたちですべてのグローバル コンポーネントの概要が表示されます。
- グローバル コンポーネントのコンテンツ モデル ビューへのナビゲーションを行う (コンポーネントにコンテンツ モデルが含まれている場合) : コンポーネントにコンテンツ モデルが含まれない場合、スキーマ概要にてコンポーネントがハイライトされます。他のスキーマからインクルードまたはインポートされたグローバル コンポーネントもコンポーネント入力ヘルパーに表示されます。
- スキーマ ドキュメント内で定義された同一性制約の概要を表示する : 同一性制約タブの詳細については、**同一性制約**」のセクションを参照ください。

メモ: 表示されるのが XSD 1.0 なのか 1.1 なのかは、選択されている XSD モード (XSD 1.0 または 1.1) により決定されます。

グローバル タブ

グローバル タブでは、グローバル コンポーネントがツリー構造でグループ化され表示されます (以下のスクリーンショットを参照)。スキーマ内にある各グローバル コンポーネントの種類の数が、各種コンポーネントの隣に表示されます。

ツリー構造では、グローバル コンポーネントが以下の 7 つの種類に分類されます。コンポーネントの種類がツリーに表示されるには、その種類のコンポーネントがスキーマ内部に最低 1 つ必要になります。

- 要素宣言 (要素)
- モデル グループ (グループ)
- 複合型
- 単純型
- 属性宣言 (属性)
- 属性グループ
- 記法

名前空間タブ

名前空間タブでは、名前空間とコンポーネントの種類により分類されたかたちでコンポーネントが表示されます (以下のスクリーンショットを参照)。

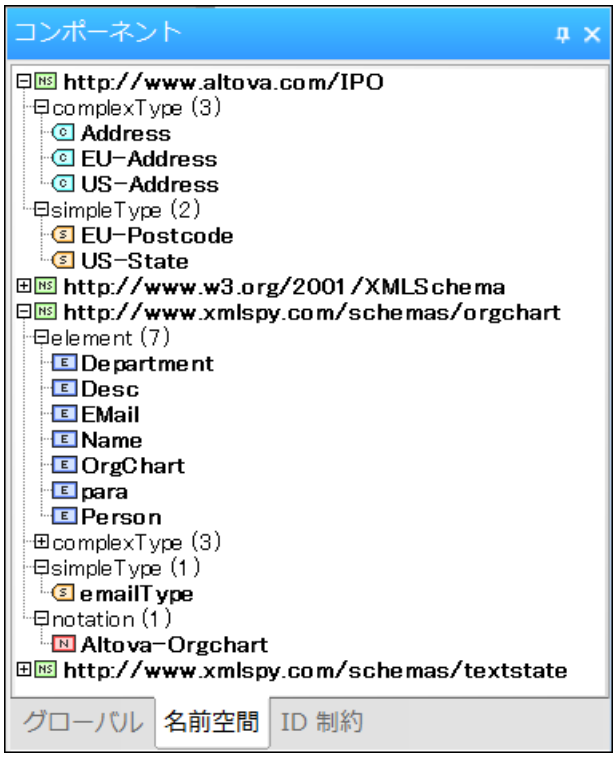
ツリーの表示では、グローバル コンポーネントが以下にある 7 つのグループに分類されます。コンポーネントの種類がツリーに表示されるには、その種類のコンポーネントがスキーマ内部に最低 1 つ必要になります。

- 要素宣言 (要素)

- モデルグループ (グループ)
- 複合型
- 単純型
- 属性宣言 (属性)
- 属性グループ
- 記法

グローバルと名前空間タブに表示されるコンポーネントのグループ

グローバルまたは名前空間タブにてコンポーネントのグループを展開すると、そのグループに含まれるすべてのコンポーネントが表示されます。これにより、ユーザー定義のコンポーネントへ簡単にアクセスすることができます。コンポーネント タブに表示されたコンポーネントをダブルクリックすると、その定義がメイン ウィンドウに表示されます。

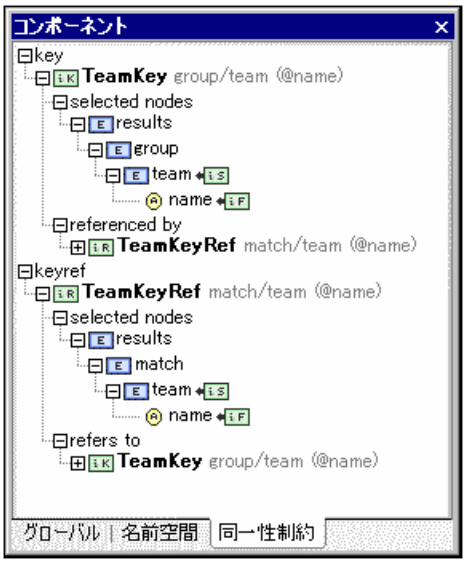


コンポーネントにコンテンツ モデルが含まれている場合 (つまり、コンポーネントが要素、グループ または複合型の場合)、コンポーネント名をダブルクリックすることで、コンポーネントのコンテンツ モデルが (メイン ウィンドウにある) コンテンツ モデル ビューに表示されます。コンテンツ モデルを含まないコンポーネントがダブルクリックされた場合、そのコンポーネントが (メイン ウィンドウにある) スキーマ概要でハイライトされます。

注意: インクルードまたはインポートされたスキーマの中にコンポーネントが含まれる場合、インクルード/インポートされたスキーマが開かれ、コンポーネントのコンテンツ モデルがコンテンツ モデル ビューに表示されるか、コンポーネントがスキーマ概要でハイライトされます。

同一性制約

コンポーネント入力ヘルパーの同一性制約タブでは、ドキュメントにある同一性制約の概要が表示されます (以下のスクリーンショットを参照)。このタブでは、同一性制約の種類 (ユニーク (unique)、キー (key)、キー参照 (keyref)) により分類されたかたちで同一性制約が表示されており、展開 縮退可能なツリーにより同一性制約が表示されます。



太字で表示されているエントリーはその同一性制約が現在アクティブなスキーマ内部にあることを表しており、そうでないエントリーはサブスキーマ内に含まれることを表しています。同一性制約タブに表示されたエントリーをダブルクリックすることで、そのスキーマ コンポーネントがコンテンツ モデル ビューにて表示されます。

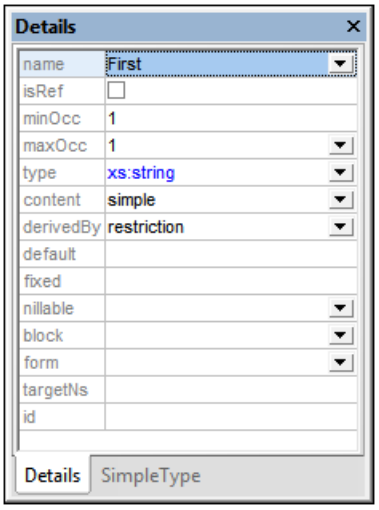
同一性制約タブにてアイテムが選択されている状態では、以下のコンテキスト メニュー コマンドを利用することができます :

- **図に表示**: スキーマ コンポーネントをコンテンツ モデル ビューに表示します。
- **図にセレクター/フィールド ターゲットを表示**: 同一性制約のセレクター (selector) またはフィールド (field) にて指定されているスキーマ コンポーネントをコンテンツ モデル ビューに表示します。複数のフィールドがある場合、目的のフィールドを選択するためのダイアログが表示されます。
- **同一性制約へ移動**: 同一性制約をスキーマ概要に表示します。
- **すべての子を展開 縮退**: ツリーを展開 縮退します。

同一性制約タブの詳細については、**「同一性制約」**のセクションを参照ください。

詳細

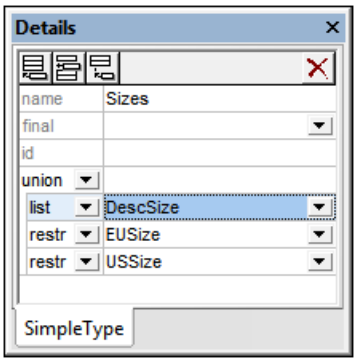
スキーマビューの詳細入力ヘルパーには、メイン ウィンドウにて現在選択されているコンポーネントやコンポジッターに関する編集可能な情報が表示されます。データベース拡張を含んでいるスキーマ ファイルを編集している場合、DB 拡張に関する情報を表示するためのタブも表示されます。



現在選択されているコンポーネントやコンポジッターのプロパティを変更するには、編集するフィールドをダブルクリックして、値を選択するか直接入力してください。編集するフィールドにてコンボボックスが利用できる場合、ドロップダウンリストから目的の値を選択してください。詳細入力ヘルパーにて行った変更は、直ちにデザインへ反映されます。

単純型の派生

詳細入力ヘルパーにて派生した単純型 (restriction、list、union) を素早く正確に作成することができます。デザインにて単純型が選択されると、詳細入力ヘルパーに単純型タブが表示されます (以下のスクリーンショットを参照)。



単純型タブにある派生型のコンボボックスにて派生型 (restriction、list、union) を選択し、右側にある対応するメンバーの型で利用可能な単純型から目的の単純型を選択してください。ツールバーにあるアイコンを使用することで、同じレベルに型を追加または挿入し、サブレベルに別の派生を追加したり、派生型を削除することができます。

ファセット

単純型の基底型 (既存の単純型) に制約を課すことで新たな単純型 (名前付き または匿名型) が作成されます。このような制約は、基底型の値にファセットを追加することで達成されます。スキーマビューにあるファセット入力ヘルパーでは、単純型のファセットを簡単かつ視覚的に編集することができます (以下のスクリーンショットを参照)。利用可能なファセットは、以下のテーブルで示されるようにファセット入力ヘルパーのタブにて管理されます。

タブ	利用可能なファセット

ファセット	minInclusive、maxInclusive、minExclusive、maxExclusive、length, minLength、maxLength、totalDigits、fractionDigits、whiteSpace、explicitTimezone
パターン	pattern
列挙	enumeration
アサーション	assertion
サンプル	altova:exampleValues はサンプルであり、**ファセットではありません。**サンプルは XMLSpy にて XML スキーマからインスタンス XML ドキュメントを生成する際のサンプル値として使用されます。

以下に、これらタブの詳細について記述します。

デザイン内にある単純型の選択

単純型（名前付きまたは匿名型）は、以下のデザイン環境にて選択することができます：

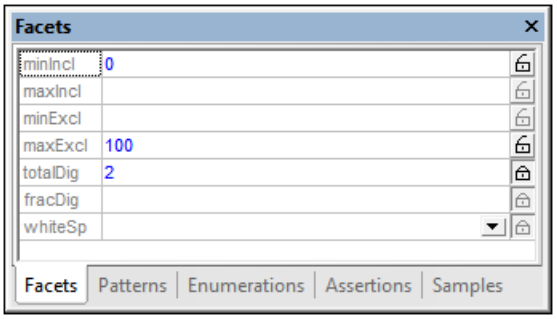
- スキーマ概要にて：グローバル コンポーネントのリスト、またはグローバル コンポーネントのリスト以下にある属性タブから。
- コンテンツ モデル ビューにて：図の中、またはデザイン下部にある属性タブから。

上記のデザイン環境にて単純型が選択されると、定義されているファセットがファセット入力ヘルパーに表示され、編集することができるようになります。

ファセット タブ

ファセット タブではデザインにて選択されている型に対して利用可能なファセットだけが表示されます。例えば制約を課されているのが xs:string 型の場合、totalDigits などの利用できないファセットは表示されません。

- 数値に関するファセット（minInclusive、maxInclusive、minExclusive、maxExclusive）は数値型と日付/時間型、そしてこれらの型から派生した型に対してだけ使用することができます。
- 長さに関するファセットは string を基底とした型、バイナリ型、そして anyURI に対してだけ使用することができます。
- totalDigits ファセットは xs:decimal ならびに整数型と、これらの型から派生したすべての型に対して使用することができます。fractionDigits ファセットは xs:decimal に対してのみ適用されます。



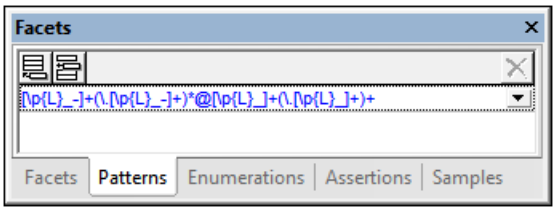
値を入力するには、（それが存在する場合には）コンボ ボックスのドロップダウン リストから値を選択、または値フィールドをダブルクリックして値を入力してください。無効な値や他の値と相反する値入力された場合、その値が赤色で表示され、妥当な

値は青色で表示されます。例えば、minInclusive ファセットとmaxInclusive ファセットは同時に定義することができないため、これらのうち後に入力された値が赤色で表示されます。

固定ファセット(fixed="true" の属性を持つファセット)を指定するには、ファセットの右側にある鍵のシンボルをクリックして、シンボルを閉じた鍵に変更してください。上のスクリーンショットでは、totalDigits ファセットが固定ファセットとして定義されています。固定することのできるファセットの数に制限はありません。ファセットの固定を解除するには、閉じた鍵のシンボルをクリックしてください。

パターン タブ

パターン タブでは、1つ以上のpattern ファセットを制約に追加することができます。pattern ファセットのパターンは正規表現の構文により指定されます。以下のスクリーンショットでは、電子メールアドレスのパターンが示されます。

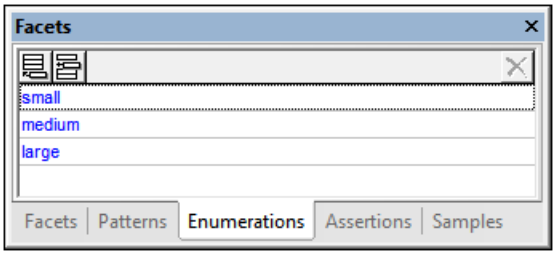


複数のpattern ファセットが指定されている場合、XML インスタンスの値が指定されたパターンの 1つにマッチする必要があります。例えば、米国と欧州の郵便番号にマッチするパターンが与えられている場合、XML インスタンスの値を、そのうち 1 つに対してマッチさせる必要があります。

左上にある **追加**」または **挿入**」アイコンをクリックして、目的のパターンを定義する正規表現を入力することで、pattern アセットを追加することができます。パターンを選択して、右上にある **削除**」アイコンをクリックすることで、そのパターンを削除することができます。

列挙タブ

列挙タブでは、enumeration ファセットを制約に加えることができます(以下のスクリーンショットを参照)。各enumeration ファセットにより、その型に対して妥当となる値が指定されます。enumeration ファセットのセットにより、許されている値のセットを指定することができます。以下のスクリーンショットでは、enumeration ファセットにより許されている文字列のセットを確認することができます。



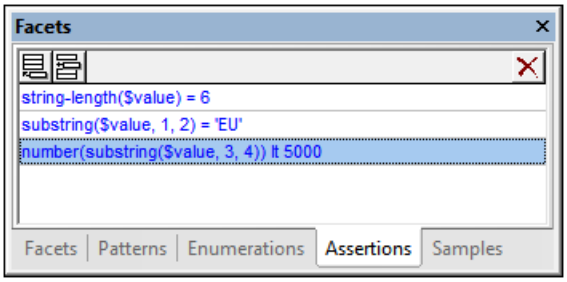
左上にある **追加**」 または **挿入**」アイコンをクリックしてenumeration の値を入力することで、enumeration ファセットを追加することができます。enumeration ファセットを選択し、右上にある **削除**」アイコンをクリックすることで、そのenumearation ファセットを削除することができます。

アサーション タブ

アサーションは XSD 1.1 にて新たに追加された機能で、XSD 1.1 モードでのみ表示されます。ファセット入力ヘルパーのアサーション タブにて定義されたアサーション ファセットは、**単純型に対するアサーション**です（複合型に対するアサーションは、ファセット入力ヘルパー**ではなく**スキーマ概要やコンテンツ モデル ビューにて定義ならびに編集されます）。

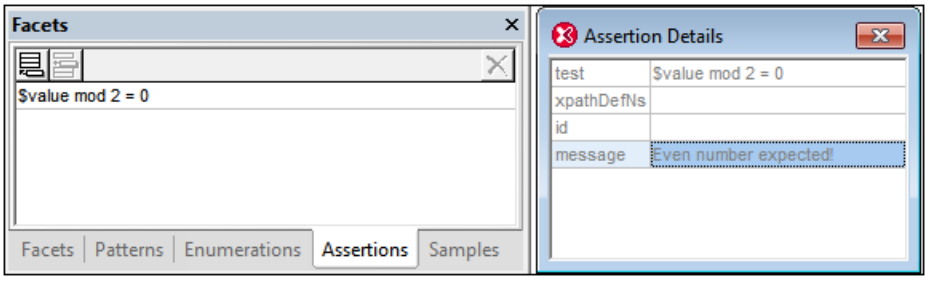
単純型（要素または単純コンテンツの属性）がデザインで選択されている際にアサーション タブへ切り替え、左上にある **追加** または **挿入** アイコンをクリックし、`assertion` を定義するのに使われる XPath 2.0 条件式を入力することで、アサーションの指定を行うことができます。XPath 条件式では、`$value` という特別な変数を使用することで、単純型の値を参照することができます（1 つの値以外に評価する要素はないため、通常使用される `self::node()` パス（またはその省略形であるピリオド `.`）を XPath 条件式内部で使用することはできません）。

例えば、`string-length($value) = 6` という XPath 条件式により、単純型に含まれている値が 6 文字から構成されるかをチェックすることができます。インスタンス ドキュメント内に含まれている要素や属性の値が 6 文字から構成される場合、アサーションに従った妥当な値ということになります。



メモ： XPath 条件式における構文エラーは赤色により示されます。しかしデータ型はランタイムにて決定されるため、型に関するエラーは XPath 条件式の入力中に報告されません。そのため期待されたとおりに型の構築を行う必要があります。例えば上のスクリーンショットにある 3 番目の条件式では、数値の比較を行う前に、文字列の値（アサーションが `xs:string` 単純型にて定義されたと仮定した場合）を数値型へ変換しています。

上のスクリーンショットの例にあるように、複数のアサーションを同じ単純型へ指定することもできます。この場合、指定されているすべてのアサーションに要素または属性の値をマッチさせる必要があります。上のスクリーンショットにあるアサーションでは、インスタンス ドキュメントの値が 6 文字から構成され、`EU` という文字から開始され、最後の 4 文字を `0000` から `4999` までの数値に収める必要があります。



アサーションの詳細がこの定義と共に提供される場合とても役に立ちます。XML インスタンス ドキュメントが検証される際、アサーションが満たされていない場合、適切な メッセージが表示されます。XML スキーマ仕様は、このようなメッセージのために対応をしないため、 XMLSpy は、アサーションの定義と共に XML インスタンス ドキュメントの検証内で使用される Altova xml-schema-extensions 名前空間 `http://www.altova.com/xml-schema-extensions`（または他の名前空間）内でメッセージを許可します。例えば：

```
<xs:assert test="count(//MyNode) ge 1" altova:message="There must be at
least one MyNode element"/> or
<xs:assertion test="count(//MyNode) ge 1" altova:message="There must be at
least one MyNode element"/>
```

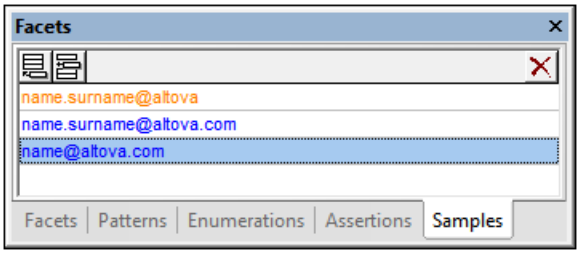
アサーション内で指定された制約が満たされない場合、XMLSpy の検証エンジンは、検証 エラーメッセージと共にヒントとしてのアサーションと関連したメッセージを表示します。 バリデーターは、 assert/@message 属性の値または、 message 属性が存在する名前空間に関わらず、アサーション/@message 属性 を報告します。ですが、 スキーマビュー内では、xml-schema-extensions 名前空間にあるmessage 属性 のみを編集することができます。他の名前空間内のmessage 属性 を編集するには、テキストビューを使用します。

詳細に関しては、アサーションメッセージを参照してください。

メモ: 可能な場合はアサーションよりもその他の種類のファセット使用する方が推奨されます。例えば、上のスクリーンショットにある最初のアサーションにより指定された制約は、(ファセット タブにある)length ファセットにより指定された方が良いでしょう。

サンプル タブ

サンプル タブでは、XML スキーマからXML インスタンス ファイルを生成する際のサンプル値として使用される値を指定することができます (以下のスクリーンショットを参照)。XML インスタンス ファイルの生成は、メニュー オプションの **DTD/スキーマ | サンプル XML ファイルの生成** から行うことができます。サンプル値が無効の場合、警告としてサンプル値がオレンジ色で表示されます。以下のスクリーンショットでは最初のサンプルが電子メールのpattern ファセットにマッチしないため、無効な値となります (上記のパターン タブを参照ください)。



メモ: ツールバーにある **検証の警告を表示** アイコン をクリックすることで、無効なサンプル値の警告を確認することができます。無効なサンプル値によりXSD ファイルの検証に失敗することはありません。

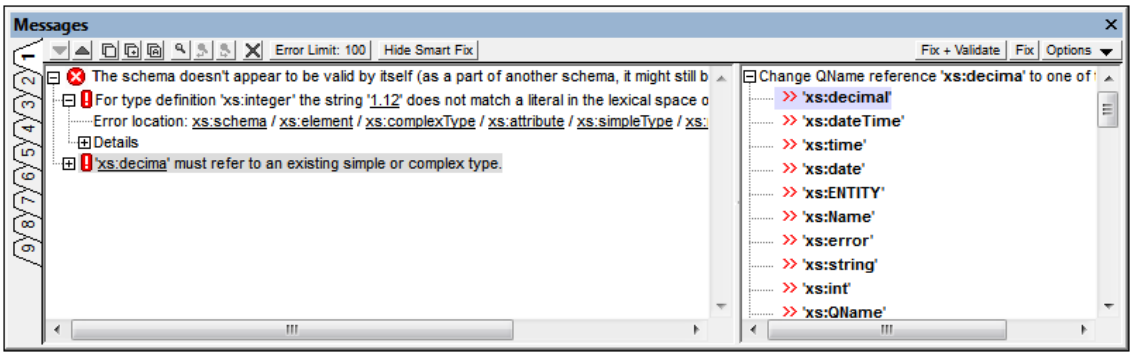
左上にある **追加** または **挿入** アイコンをクリックしてhttp://www.altova.com/xml-schema-extensions内のサンプル値を入力することで、altova:example へサンプル値が入力されます。サンプル値を選択し、右上にある **削除** アイコンをクリックすることで、altova:example の値が削除されます。

2.3.6 検証とスマートフィックス

XML スキーマ ドキュメントの正確性の確認のために検証を行うことができます。これはメニューコマンド **[XML | XML の検証 (F8)]** をクリックして行います。

ドキュメントが有効な場合、この効果についてのメッセージがメッセージウィンドウに表示されます。

ドキュメントが無効な場合、メッセージ ウィンドウは、2 つの ペインの表示をするように変更されます (下のスクリーンショット参照)。左側の ペイン (エラー ペイン)は、最初の x エラーまたは すべての エラーをリストします。右側のペインは、スマート フィックス ペインです。 このペインは左側の ペインで選択されたえらのためのフィックスの選択肢のリストを含みます。下のスクリーンショットの例では、エラー ペイン内の2番目のエラーの選択は、右側の スマートフィックス ペインにこのエラーのフィックスの選択肢の表示をします。 フィックスの1つを選択してを **[フィックス+検証]** または **フィックス** 選択すると、ドキュメント内のエラーはこの特定のフィックスと共に修正されます。



エラー ペイン

ウィンドウの ツールバーには、次の機能があります :

- **[Up]** および**[Down]** 矢印を使用して、エラーをスクロールします。
- メッセージ、または メッセージとその子孫、または すべての メッセージをクリップボードにコピーします。
- 検索を使用して、用語を検索します。複数のエラーが報告されている場合、次を検索、および前を検索 機能は、役に立ちます。
- エラー ペイン空すべてのエラーをクリアします。
- 検索され表示されるエラー数の制限を設定します (1 から999)。デフォルトの値は100 です。ボタンをクリックして制限を編集します。
- 表示 /非表示 スマートフィックス ペイン スマートフィックス ペインが非表示の場合、**[スマートフィックスを表示]** ボタンがツールバーに表示されます。クリックするとスマートフィックスペインが表示され、ボタンが**[スマートフィックスを非表示]** に変更されます。**[表示 /スマートフィックスを非表示]** ボタンが、有効化されていない場合、スマートフィックスを使用することができません。

スマートフィックス ペイン

ウィンドウのツールバー で次の機能を使用することができます :

- **[フィックス+検証]** ボタンは、選択されたエラーを選択された スマートフィックスで修正します。そしてドキュメントを再検証します。他の エラー は、エラーペインで再度開かれます。
- **フィックス** ボタンをクリックすると、エラー はフィックスされますが、再検証は行われません。
- **オプション** ボタンは、スマートフィックスをダブルクリックすることにより作動する振る舞いの選択を含むリストをドロップします :ダブルクリックにより**[フィックス+検証]** または**[フィックス]** を実行するかどうか。

2.3.7 アサーションメッセージ

XML Schema 1.1 内では、 複合型 (xs:assert 要素を使用して)および単純型 (xs:assertion 要素を使用して)のためにアサーションは、定義されることができます。

アサーションの詳細がこの定義と共に提供される場合とても役に立ちます。XML インスタンス ドキュメントが検証される際、アサーションが満たされていない場合、適切な メッセージが表示されます。XML スキーマ仕様は、このようなメッセージのために対応をしないため、 XMLSpy は、アサーションの定義と共にXML インスタンス ドキュメントの検証内で使用される Altova xml-schema-extensions 名前空間 http://www.altova.com/xml-schema-extensions (または他の名前空間)内でメッセージ を許可します。例えば :

```
<xs:assert test="count(//MyNode) ge 1" altova:message="There must be at
least one MyNode element"/> or
<xs:assertion test="count(//MyNode) ge 1" altova:message="There must be at
least one MyNode element"/>
```

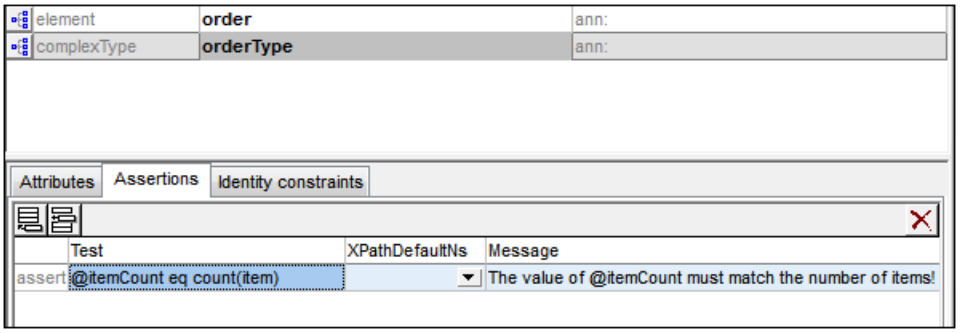
アサーション内で指定された制約が満たされない場合、XMLSpy の検証エンジンは、検証 エラーメッセージと共にヒントと

してのアサーションと関連したメッセージを表示します。 バリデーターは、 assert/@message 属性の値または message 属性が存在する名前空間に関わらず、アサーション/@message 属性 を報告します。ですが、スキーマビュー内では、xml-schema-extensions 名前空間にある message 属性 のみを編集することができます。他の名前空間内の message 属性 を編集するには、テキストビューを使用します。

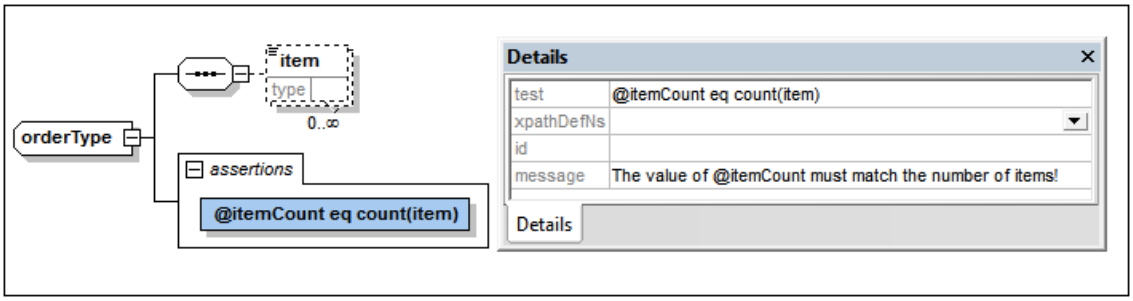
---

xs:assert メッセージの編集

スキーマビュー内では、(複合型のための) xs:assert 要素 を作成して、属性 アサーション/ID 制約 (AAIDC) ペイン または関連する複合型の詳細入力ヘルパー 内で編集することができます。下のスクリーンショットは、複合型 orderType のためのアサーションを表示します。 (この場合は、 xs:assert) は、(スキーマ概要の) アサーション タブでアサーション メッセージと共に定義されています。

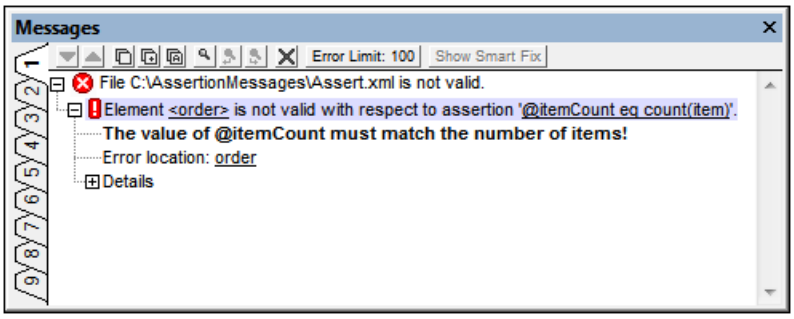


ダイアグラム内でアサーションを表示する オプションが選択された場合、複合型 のアサーションが作成され、コンテンツモデルビュー内で編集されます。アサーション メッセージを追加または編集するには、アサーションを選択して、詳細入力ヘルパー内のアサーション メッセージに入力してください (*下のスクリーンショット参照*)。



この方法で作成されるアサーション メッセージは、Altova xml-schema-extension 名前空間 http://www.altova.com/xml-schema-extensions 内にあります。 XML スキーマ ドキュメント内に最初のアサーション メッセージを、AAIDC ペイン または 詳細入力ヘルパーを介して追加すると、Altova xml-schema-extension 名前空間が自動的に xs:schema 要素に対して宣言されます。

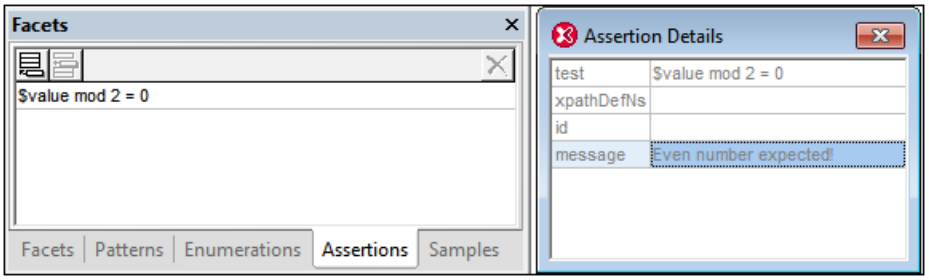
XML ファイルが検証されてアサーション テストが満たされない場合、アサーションのために定義された、エラー メッセージと共に表示されます (*下のスクリーンショット参照*)。



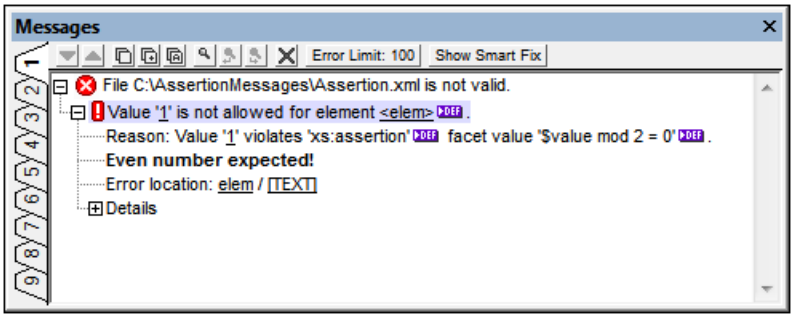
---

xs:assertion メッセージの編集

スキーマビューの xs:assertion 要素（単純型）は、関連する単純型のファセット入力ヘルパー内で作成および編集されることができます。アサーションメッセージを編集するには、ファセット入力ヘルパー（*下のスクリーンショット参照*）内のアサーションを右クリックして、ポップアップするメニューの**[詳細]**をクリックし、アサーション詳細モーダル ウィンドウのメッセージを編集します（*下のスクリーンショット参照*）。メモ この方法で作成される、アサーションメッセージは、Altova xml-schema-extension 名前空間 http://www.altova.com/xml-schema-extensions 内にあります。アサーション詳細モーダル ウィンドウを介して、XML スキーマ ドキュメント内のメッセージに最初のアサーションを追加する際には、名前空間は、自動的に xs:schema 要素 に対して宣言されます。



XML ファイルが検証されて、アサーションテストが満たされない場合、アサーションのために定義されたメッセージがエラーメッセージと共に表示されます（*下のスクリーンショット参照*）。

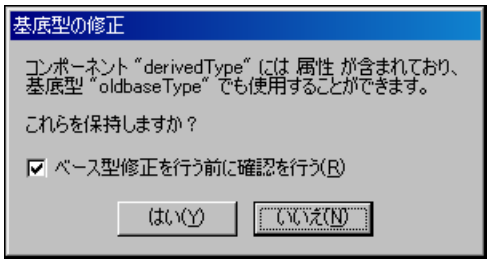


2.3.8 基底型の修正

派生した型の基底型がスキーマビューで変更されると、派生型にあるコンテンツ、属性、ファセット、サンプル値を、以下のいずれかの方法で処理することができます：

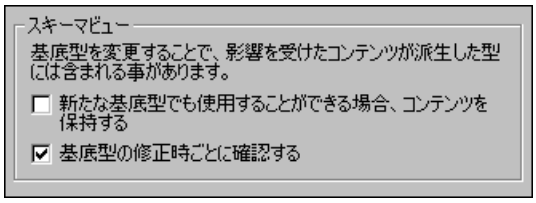
- 新たな基底型でも使用できる場合、これらの値を保持する。
- 新たな基底型でも使用できるかどうかにかかわらず、これらの値を自動的に削除する。

コンテンツ、属性、ファセット、またはサンプル値が含まれる派生型の基底型を切り替えると、基底型の修正ダイアログが表示されます (以下のスクリーンショットを参照)。



基底型の修正を行う前に確認」チェックボックスを無効にした場合、オプションダイアログのビュータブ (**「ツール | オプション | ビュー」**) にてこの設定を再度有効にすることができる旨のメッセージが表示されます。

オプションダイアログのビュータブ (**「ツール | オプション | ビュー」**) にあるスキーマビューペインでは、コンテンツを保持するか、または基底型の修正を行うたびにユーザーへの確認を行うかを指定することができます (以下のスクリーンショットを参照)。



コンテンツを保持して、確認を行うチェックボックスを有効にすることで、これらの動作がデフォルトのオプションになります。

2.3.9 スマート制約

複合型に対して制約を課す場合、基底型に含まれるコンテンツモデルの一部が派生型にて書き換えられることになります。派生型の編集を行いながらその基底型のコンテンツモデルを正確に把握するのは大変な作業です。

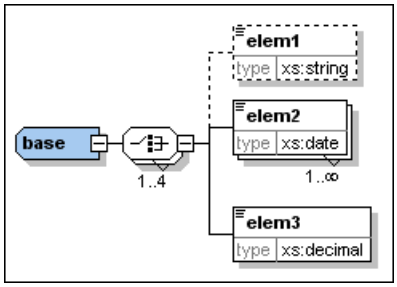
スマート制約により、(基底型と派生型の) 2つのコンテンツモデルが派生した型のコンテンツモデルに表示されます。派生した複合型では、基底複合型にあるすべての項目と、それらが派生した型とどのように関連付けられているのかを確認することができます。更にスマート制約では、基底型のコンテンツモデルへ課すことのできるすべての制約が表示されます。これで派生した型を簡単かつ正確に制約することが可能になります。

以下の操作によりスマート制約を有効にすることができます：

- スキーマ デザイン ツールバーにあるスマート制約アイコン をクリックします。

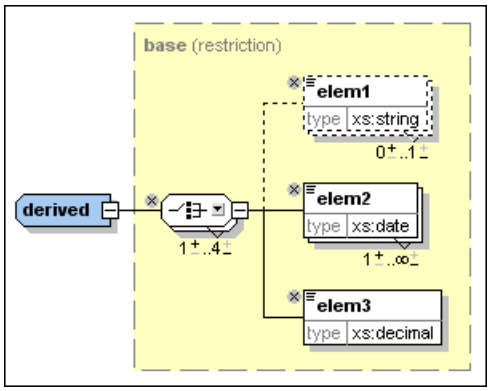
以下のサンプルはスマート制約の機能を説明しています。

この例では以下に示される複合型が基底型として使用されます



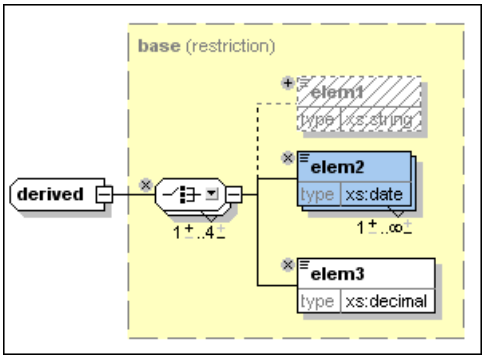
上に示される基底型のbase から、以下のように複合型のderived を派生させることができます：
1. スキーマにてderived という名前の複合型を新たに作成します。
2. 詳細入力ヘルパーにてbase ドロップダウンリストからbase を選択し、derivedBy ドロップダウンリストから restriction を選択します。

スマート制約が有効になっている状態で、派生型が以下のように表示されます

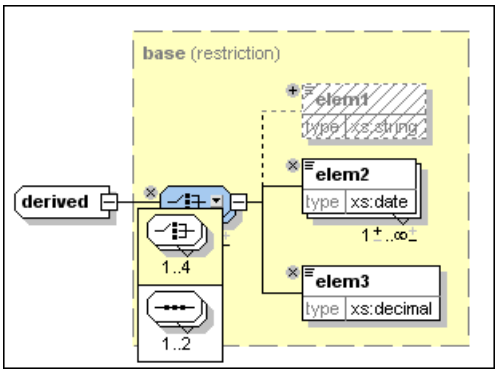


この例で派生型の制約に使用されるコントロールを以下に示します：

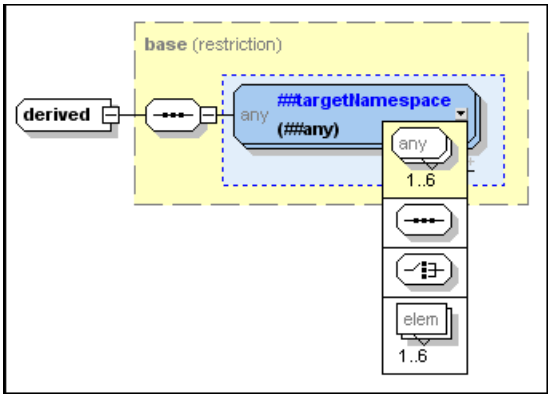
- このアイコン により、基底型にある要素を削除することができます。ここでは `elem1` が削除されます。要素を追加するには、アイコン をクリックしてください。



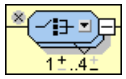
- 選択コンポジッターに表示された矢印 ▣ をクリックすることで、以下のスクリーンショットに示されるリストが表示されます。選択モデル グループをシーケンス モデル グループへ変更することができます。

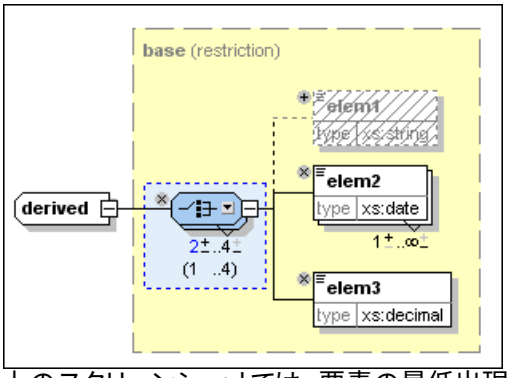


同様に、ワイルドカードを変更することもできます



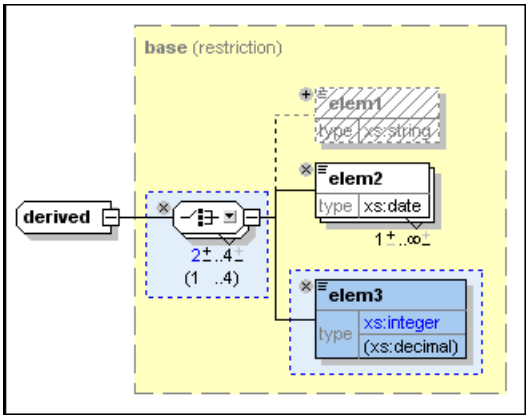
どの項目をどの項目により置き換えることができるかの詳細については、XML スキーマの仕様 (英文) を参照ください。

- モデル グループの出現回数は、  にある 1の隣にあるプラス記号により最低出現回数を増やし、4の隣にあるマイナス記号により最大出現回数を減らすことができます。基底型にて定義された出現の回数が特定の数 (例えば 4)ではなく範囲の場合 (例えば 2-5)の場合にこれらのコントロールが表示されます。出現の範囲が間違っていてもこれらのコントロールは表示されます。

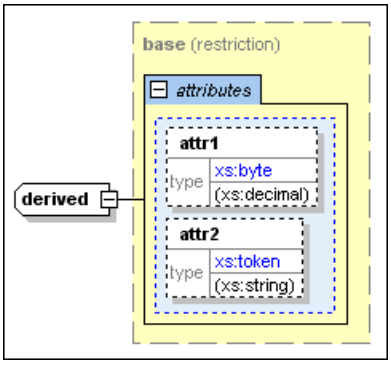


上のスクリーンショットでは、要素の最低出現回数が 2に変更されています。モデル グループの背景が青色に変更され、基底複合型にあるモデル グループとは違うものであることを表している点に注目してください。基底型にて定義された出現範囲も表示されます。

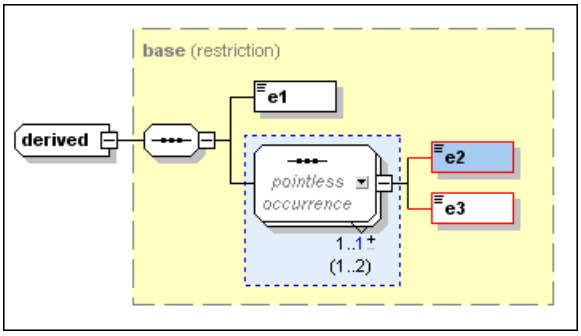
- それが XML スキーマの仕様で定義された基底型の制約に対して妥当な変換である場合、属性または要素のデータ型を変更することができます。例えば、derived 複合型にある elem3 データ型を decimal から integer へ変更することができます。これにより、要素の背景が青色に変わり、基底型のデータ型とは違っている点が示され、基底型で定義されている要素のデータ型がかっこの中に表示されます：



派生した複合型にて値のデータ型が制限された属性を以下のスクリーンショットに示します



- スマート制約はコンテンツ モデルで**意味のない出現**の警告を行います。意味のない出現とは、例えばコンテンツ モデルで表示されている、実際には必要ないシーケンスのことです。以下の図に意味のない出現の例を示します：



メモ: 意味のない出現は、コンテンツ モデルがエラーを含んでいる時にだけ表示されます。コンテンツ モデルが意味のない出現を含んでいながら妥当である場合もあり、その場合は混乱を避けるために意味のない出現は明示的に表示されません。

意味をなさない表示の詳細については、XML スキーマの仕様を参照ください。

2.3.10 xml:base、xml:id、xml:lang、xml:space

名前空間 `http://www.w3.org/XML/1998/namespace` は XML 名前区間の仕様により与えられたもので、`xml:` 接頭辞に紐付けられています。これにより、`xml:` 接頭辞とともにこの名前空間が使用されることになります。この名前空間では 4 つの属性が定義されており、任意の XML ドキュメント (スキーマかインスタンスかを問わず) にあるすべての XML 要素にて使用することができます:

- `xml:base`: 要素のベース URI をセットするのに使用
- `xml:id`: 要素のユニークな ID を指定するのに使用
- `xml:lang`: その要素内にあるコンテンツが記述されている言語を識別するために使用
- `xml:space`: 要素内にある空白スペースをどのように処理するかを指定するために使用

スキーマ ビューにて、XML スキーマ ドキュメントへこの XML 名前空間がインポートされると、これら 4 つの `xml:` 属性をスキーマ内にあるすべての要素にて使用することができます。

以下の操作により、これら属性のうち 1 つを要素にて宣言することができます:

1. そのスキーマ ドキュメントに対する XML Namespace 名前空間を宣言し、名前空間を `xml:` 接頭辞に割り当てます。4 つある `xml:` 属性のうちのどれかがドキュメントにて使用されると、その名前が正しい名前空間を含めるように展開されます。
2. XML Namespaces 名前空間をインポートします。XMLSpy のバリデーターが名前空間を認識し、4 つの `xml:` 属性がスキーマ内部から参照可能なグローバル属性で利用できるようになります。
3. 目的の `xml:` 属性を要素の子として挿入します。インポートされたグローバル属性への参照として属性は宣言されます。

XML Namespaces 名前空間の宣言

そのスキーマに対して宣言されているすべてのスキーマを確認ならびに編集することができるスキーマ設定ダイアログでは、XML Namespaces 名前空間 (`http://www.w3.org/XML/1998`) を入力することができます。名前空間は `xml:` 接頭辞へ関連付ける必要があります (`xml:` 接頭辞とともに、テキスト ビューで表示された `xs:schema` 要素にて名前空間を宣言することもできます)。

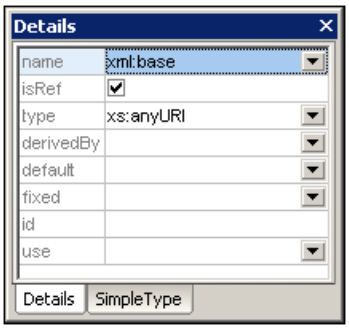
XML Namespaces 名前空間のインポート

スキーマ概要にてXML Namespaces 名前空間に対するグローバル要素宣言を作成します。スキーマ概要ウィンドウの上部にある挿入 または追加 アイコンをクリックし、メニューから **「インポート」** を選択します。XML Namespaces 名前空間をインポートする名前空間として入力します。テキスト ビューではインポート宣言が以下のように表示されているはずです :

```
<xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/XML/1998/namespace"/>.
```

xml: 属性の追加

スキーマ概要にて `xml:` 属性を追加する要素を選択し、その要素に対して属性を追加します。詳細ウィンドウにて name コンボ ボックスにある矢印ボタンをクリックし、目的の `xml:` 属性 (例えば `xml:base`) を選択します (以下のスクリーンショットを参照)。グローバル属性を参照するか尋ねられた場合、 **「はい」** を選択します。属性が参照として追加されます。



XInclude とxml:base

XInclude の `include` 要素がインクルード要素の `href` 属性にて指定されたXML ファイルにより置き換えられた場合、解析されたXML ドキュメントの最上レベルにある要素に `xml:base` 属性が与えられます。このXML ドキュメントを検証する必要がある場合、`xml:base` 属性を関連する要素に対して定義する必要があります。

2.3.11 戻ると進む :位置の移動

スキーマビューの **「戻る/進む」** コマンドにより、スキーマビューで直前に閲覧した位置に戻ることができます。スキーマビューにてスキーマ コンポーネント間を移動しながら、既に見たコンポーネントを再度閲覧するのに便利な機能です。ツールバーにある **「戻る」** ボタンを一度クリックすることで、直前に閲覧した位置に移動します。 **「戻る」** ボタンを繰り返し押すことで、既に閲覧したポジション (最大 500) まで移動することができます。既に閲覧した位置に移動した後、 **「進む」** ボタンをクリックすることで先の位置に移動することができます。

これら 2 つのショートカット キーを以下に示します :

- **戻る:Alt + 左矢印**

- **進む :Alt + 右矢印**

戻る /進むと元に戻す /やり直し

戻る/**進む**コマンドは**元に戻す（Ctrl」+Z)** や**やり直し（Ctrl」+Y)** とは違うものだということに注意してください。**戻る**」コマンドをクリックすることで、最後に見た状態のまま直前に閲覧したコンポーネントに戻ります。元に戻すコマンドを一度選択すると、最後に行った変更を（その変更がどこで行われたかに関わらず）取り消します。

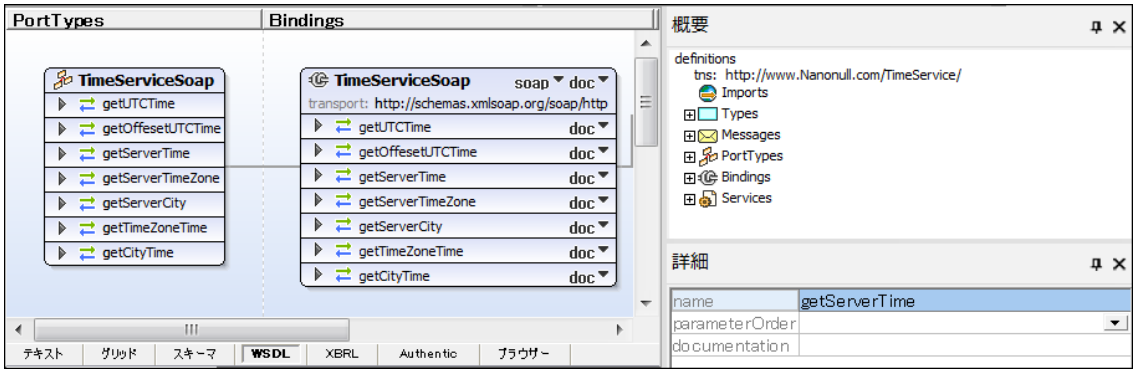
追加メモ

以下の点に注意してください：

- **戻る**」ボタンにより、最大 500回まで最後に閲覧した場所に移動できます。
- 戻る進む機能はスキーマ全体に対して有効になります。スキーマが閉じられたり、他のビューで開かれていたとしても、スキーマビューにて再度開かれます。
- 前に閲覧したコンポーネントが削除されていた場合、コンポーネントを閲覧することはできません。そのようなコンポーネントが前の位置の一部であった場合、削除されていないコンポーネントとともに以前の画面が表示されます。削除されたコンポーネントが以前閲覧した位置のすべてを含んでいた場合、その位置を表示することはできず、**戻る**」ボタンをクリックしても、削除されたコンポーネントのさらに 1つ前の位置に移動します。

2.4 WSDL ビュー

WSDL ビュー (下のスクリーンショット) はWSDL 1.1 とWSDL 2.0 ドキュメントをグラフィカルに編集するインターフェースを提供します。WSDL ビューはWSDL ドキュメントがアクティブになっているときにWSDLビュータブにて有効になります。グラフィカルなWSDL ドキュメントの構造とコンポーネントがメインウィンドウで作成され、入力ヘルパーには特別な編集機能が有効になります。



メインウィンドウ (ポートタイプ (WSDL 1.1) またはインターフェース (WSDL 2.0)、バインディングとサービス) と入力ヘルパー (概要と詳細) についてはこのセクションのサブセクションにて説明されます (オブジェクトに関する詳しい情報についてはリファレンスセクションのオブジェクトメニューを参照下さい)。

WSDLビューで使用することのできる機能

WSDL ビューでは以下の機能を使用することのできます:

- メインウィンドウ内ですべてのWSDL 要素 (ポートタイプ (WSDL 1.1) またはインターフェース (WSDL 2.0)、バインディング、サービス) のグラフィカルな表示
- ドラッグアンドドロップによるWSDL 要素の直接操作
- 動的なグラフィカルビューによる任意のWSDL 要素の追加、操作、削除機能
- 詳細入力ヘルパーによる値の入力と編集
- W3C Working Draft に従ったWSDL の検証機能
- 全てのWSDL 要素を、ポートタイプ (WSDL 1.1) またはインターフェース (WSDL 2.0)、バインディング、サービスによりグループ化されたかたちでメインウィンドウに表示
- ドラッグアンドドロップによりWSDL 要素を直接操作
- 参照またインポートしたスキーマが不正である場合のスキーマビューへの自動的な切り替え
- WSDL ビュー内からスキーマ型の編集
- MS Word またはHTML 形式でのWSDL ドキュメンテーション生成
- メインウィンドウのWSDL ドキュメントのダイアグラム (PNG イメージ) 生成
- WSDL ウィンドウのビューを印刷

ファイル閲覧

ファイルの閲覧に関する以下の点に注意して下さい:

- WSDL ファイルを開いたときには、自動的にWSDL ビューで開かれます。
- WSDL ドキュメントをテキストビューや拡張グリッドビューで閲覧することも可能です。対応するタブをクリックして下さい。
- WSDL ファイルがXML スキーマに対する参照を含んでいる場合、メニューコマンド **「WSDL | 型 | スキーマビューでスキーマを編集」** によりそのスキーマを閲覧、編集することができます。
- 関連付けられているスキーマファイルが開かれている場合、WSDL ファイルのビューを変更することはできません (例えば、WSDL ビューからテキストビューへの変更)。WSD L ファイルのビューを変更する前に、スキーマファイルを保存して閉じたことを確認して下さい。

WSDL ドキュメント編集の助けになる2つの入力ヘルパーがあります: 概要と詳細入力ヘルパーです。両方の入力ヘルパーともタイトルバーをダブルクリックすることにより切り離し/ドッキングを行うことができます。ドッキングされているときには、タイトルバーにあるピンアイコンをクリックすることで自動的に隠す機能を有効にできます。自動的に隠す機能が有効になっていれば、入力ヘルパーはアプリケーションウィンドウの端にあるタブとして最小化されます。隠れた入力ヘルパーはマウスポインターをタブに移動して、タイトルバーのピンアイコンを再度クリックすることで元の状態に戻すことができます。

関連項目: WSDL ドキュメントの使用に関する詳細については、WSDL チュートリアルならびにユーザーリファレンスの WSDL メニューを参照ください。

2.4.1 メインウィンドウ

メインウィンドウによりWSDL ドキュメントの編集を行います。メインウィンドウには垂直の点線で分けられた3つのセクションが存在します: (i) ポートタイプ (WSDL 1.1) またはインターフェース (WSDL 2.0); (ii) バインディング; (iii) サービス。ポートタイプとバインディングの関係、またバインディングとサービスの関係はそれぞれ接続線で示されます。これら3つのセクションについては以下で説明します。

メインウィンドウのシンボル

以下のシンボルがメインウィンドウでは使用されます:

- WSDL 1.1 ではポートタイプ, WSDL 2.0 ではインターフェース
- バインディング
- サービス
- Fault
- オペレーション。緑の矢印はインプットを表し、青の矢印はアウトプットを表します。オペレーションの種類により適切なシンボルが使用されます。
- メッセージ
- メッセージパート (パラメタ)
- XSD要素
- XSD単純型または複合型
- ポート

新しいポートタイプ、インターフェース、バインディング、サービスを追加する

新しいポートタイプ (WSDL 1.1 ドキュメント)、インターフェース (WSDL 2.0 ドキュメント)、バインディング、またはサービスを追加するには、メインウィンドウのどこか (ただしコンポーネントボックスの外) を右クリックして、現れるコンテキストメニューから適切なコマンドを選択します。

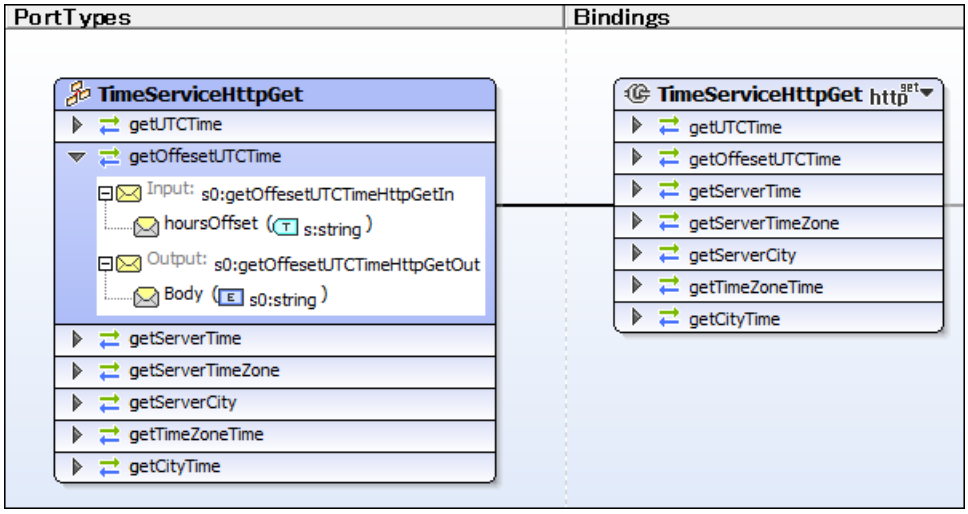
ドラッグアンドドロップ機能

以下のドラッグアンドドロップ機能が使用可能です：

- メインウィンドウにて、ドラッグアンドドロップによりポートタイプ (WSDL 1.1) またはインターフェース (WSDL 2.0)、バインディング、そしてサービス間の関係を記述することができます。
- WSDL 2.0 ドキュメントでは、概要入力ヘルパーの要素をメインウィンドウと概要入力ヘルパーのインターフェース fault にドラッグすることができます。

ポートタイプ (WSDL 1.1)、インターフェース (WSDL 2.0)

ポートタイプセクション (WSDL 1.1 ドキュメント) にはWSDL ドキュメントで定義された全てのポートタイプが含まれます (下のスクリーンショットではポートタイプセクションのポートタイプ1つだけを表示している)。インターフェースセクション (WSDL 2.0 ドキュメント) にはWSDL ドキュメントで定義された全てのインターフェースが含まれます。



各ポートタイプまたはインターフェースは定義されたオペレーション を含むボックス として表現されます。コンポーネントはボックス内で直接編集することができます。ポートタイプとインターフェースボックスの主な機能は以下の通りです：

- オペレーションはオペレーション名の左にある アイコンをクリックすることで、メッセージ を閲覧できるよう展開することができます。
- WSDL 1.1 では、メッセージにメッセージパート が含まれることがあります。このようなメッセージはメッセージパートを閲覧できるよう展開することができます。
- ポートタイプボックス (ポートタイプ、メッセージ、またはメッセージパート) のコンポーネントを右クリックすることで、コンテキストメニューが表示され、そこから関連するアクションを選択することができます。例えば、ポートタイプ名を右クリックすることで、新しいポートタイプや追加、選択したポートタイプに対してオペレーションを追加したり、バインディングを作成することができます。
- オプションであるWSDL 2.0 インターフェースプロパティーである、*extends*、*styleDefault*、そして *documention* は空の場合隠されます。これらの項目はインターフェースのコンテキストメニューの編集コマンドにより編集することができます。
- WSDL 2.0 ドキュメントにて、オペレーションのプロパティーはオペレーションのコンテキストメニューの編集コマンドにより編集することができます。
- コンポーネントが選択されたとき、その詳細を詳細入力ヘルパーにて編集することができます。
- ポートタイプとインターフェースのドキュメンテーションは個々のボックスの下部に表示されます。

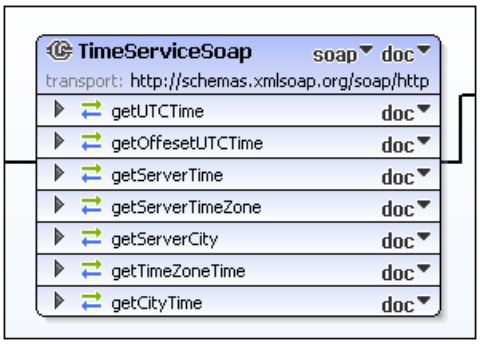
ポートタイプまたはインターフェースとバインディングの関連性はメインウィンドウに表示される *ポートタイプボックス*または*インターフェースボックス*と*バインディングボックス*を繋ぐ黒色の接続線で表されます。*バインディングボックス*はメインウィンドウの*バインディングセクション*に表示されます。

バインディング

バインディングは以下のメッセージフォーマットとプロトコルの詳細を定義します：

- 特定のポートタイプ (WSDL 1.1) で定義されたオペレーション、または
- 特定のインターフェース (WSDL 2.0) で定義されたオペレーション

WSDL 1.1 では、SOAP 1.1 または SOAP 1.2 のエンドポイント、または HTTP 1.1 の GET と POST verb 属性に対してバインディングが作成されます。WSDL 2.0 において、バインディングは SOAP 1.1 または SOAP 1.2 エンドポイント、または HTTP に対して作成されます。各バインディングはメインウィンドウの*バインディングセクション*にある*バインディングボックス*により表示されます (下の*スクリーンショット*を参照)。*バインディングボックス*には全てのオペレーションと、関連した*ポートタイプ*または*インターフェース*が表示されます (下の*スクリーンショット*を参照)。

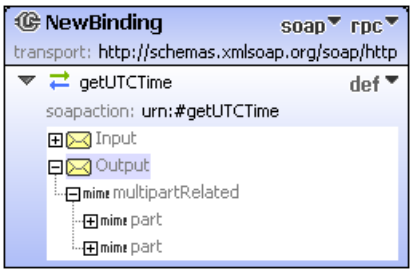


バインディングは以下のどれかの方法でポートタイプまたはインターフェースと関連付けられます：

- ポートタイプまたはインターフェースを右クリックして、**「バインディングのポートを作成」**または**「バインディングのインターフェースを作成」**を選択します。
- WSDL 1.1 バインディングを右クリックして、ポートタイププロパティーを編集します。
- WSDL 2.0 バインディングを右クリックして、**編集 | インターフェース**を選択します。

バインディングを定義するには、バインディング名の隣にあるコンボボックス (下の*スクリーンショット*) から必要なプロトコルを選択します。WSDL 1.1 では、soap 1.1、soap 1.2、http-get、または http-post によりバインディングの種類を決定します。SOAP プロトコルを選択した場合、他にも (2 つ目のコンボボックスを使用して) スタイルを doc か rpc に設定できます。WSDL 2.0 ドキュメントの場合、wsoap:protocol プロパティーをバンディングのコンテキストメニューから**「編集」**コマンドにより追加することができます。

WSDL 1.1 では、MIME エンコーディング (MIME バインディング) がメッセージレベルで定義されます。MIME エンコーディングを定義するには、メッセージを右クリック (下の*スクリーンショット*) して、適切な MIME 定義を追加します。下の*スクリーンショット*では、`Output` メッセージの MIME 宣言が作成されています。



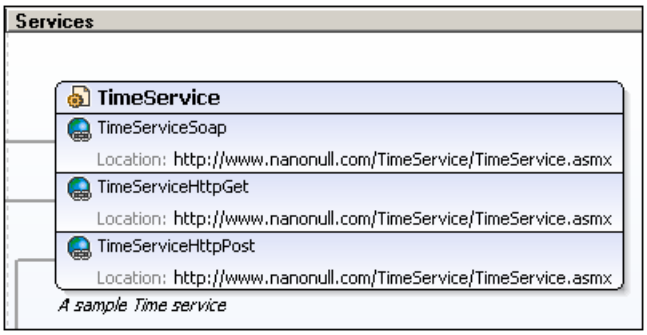
バインディングボックスにある特定のアイテムを右クリックしてコンテキストメニューを開きます。コンテキストメニューを使用して、例えば、バインディングの追加や削除、アイテムの定義や編集を行うことができます。バインディングボックスまたはその中のアイテムが選択されたとき、その定義が詳細入力ヘルパーに表示され、編集を行うことができるということにも注目して下さい。

バインディングのポートはバインディングボックスのタイトルバーを右クリックして、(WSDL 1.1 ドキュメントでは) **『バインディングのポートを作成』**コマンド、または (WSDL 2.0 ドキュメントでは) **『バインディングのエンドポイントを作成』**コマンドを選択することで行うことができます。関連するポートまたはエンドポイントが (メインウィンドウのサービスセクションにある) サービスボックスの中に作成されます。バインディングとポートの関連は黒色の接続線により表されます。

バインディングのドキュメンテーションは各バインディングボックスの下部に表示されます。

サービス

サービスグループでは複数のポート (WSDL 1.1) またはインターフェース (WSDL 2.0) のリレーションがまとめられます。メインウィンドウのサービスセクションにあるサービスボックス (下のスクリーンショット) により表示されます。1つのサービスボックスには1個以上のポートまたはエンドポイント宣言が存在します。



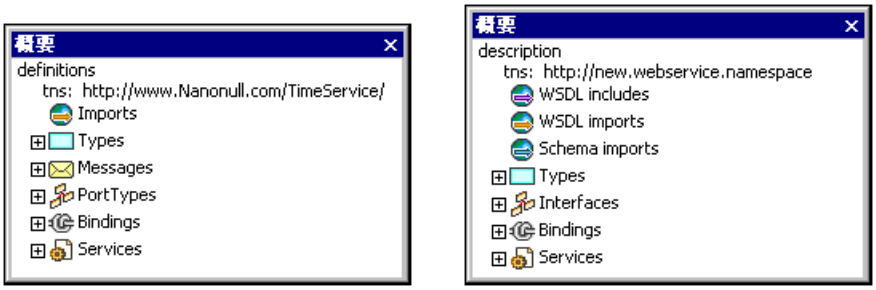
サービス名、ポートまたはエンドポイントの名前、ポートまたはエンドポイントに関連付けられたバインディング、そしてポートまたはエンドポイントのアドレス情報は詳細入力ヘルパーにて直接編集することができます。サービスボックスかその中にある特定のアイテムを右クリックして表示されるコンテキストメニューから、サービスもしくはそのアイテムに関係するコマンドを選択します。

サービスに関するドキュメンテーションは個々のサービスボックス下部に表示されます。

2.4.2 概要入力ヘルパー

概要入力ヘルパー (下のスクリーンショット) にはWSDL ドキュメントの概要がコンポーネントがターゲット名前空間、インポートされたスキーマ、インクルード/インポートされたWSDL ドキュメントごとにカテゴライズされて表示されます。ポートタイプ (WSDL 2.0 ではインターフェース) に加えて、メッセージ (WSDL 1.1)、バインディング、そしてサービスなどのドキュメントで定義された様々な型が表示されます。

概要入力ヘルパーでは、インポートならびにインクルードされたXML スキーマやWSDL ファイルを管理することもできます。



WSDL 1.1 (左) とWSDL 2.0 (右) の概要入力ヘルパー

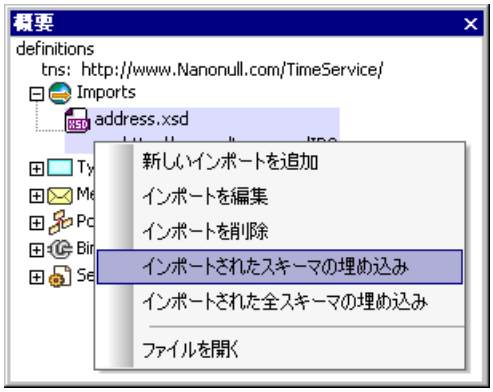
各カテゴリでは、コンポーネントがツリービューで表示されます。ツリーアイテムは展開したり折りたたんだりして、内容を表示したり隠したりすることができます。概要入力ヘルパーのコンポーネントを選択することで、プロパティーの編集を行うことができる詳細入力ヘルパーにその情報が表示されます。ツリーに表示されているWSDL とスキーマコンポーネントの名前は直接編集することができます。外部で定義されたコンポーネント (インクルードやインポートされたWSDL ドキュメントやスキーマでグレーで表示されます) を編集することはできません。概要入力ヘルパー内の個々のカテゴリについての説明は以下でなされます。

ターゲット名前空間 (WSDL 1.1 ならびに 2.0)

ツリーが**tns** で表示され、概要入力ヘルパーにて編集することができます。その他の名前空間は全てテキストビューで編集する必要があります。

Imports (WSDL 1.1)

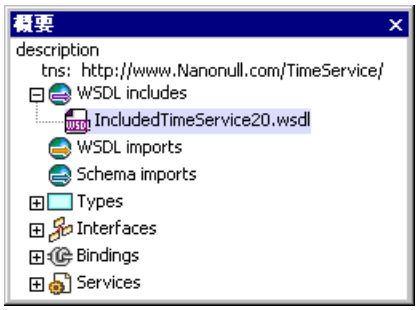
XML スキーマ (XSD) ファイルならびにWSDL ファイルをアクティブなWSDL ドキュメントにインポートすることができます。XML スキーマまたはWSDL ファイルをインポートするには、Imports アイテム、またはImports リストに表示されている既にインポートされたアイテムを右クリックして、**新しいインポートを追加**」を選択します。Imports リストにあるインポートされたファイルを右クリックすることでコンテキストメニューが表示され、新たなインポートを加えたり、選択されたファイルを置き換えるファイルを選択 (**インポートを編集**) したり、インポートされたファイルを削除することができます。ファイルをロケーションから開くこともできます。ファイルは (`wsdl` ファイルの場合) WSDL ビュー、または (`xsd` ファイルの場合) スキーマビューで開かれ、そのビューで編集することができます。



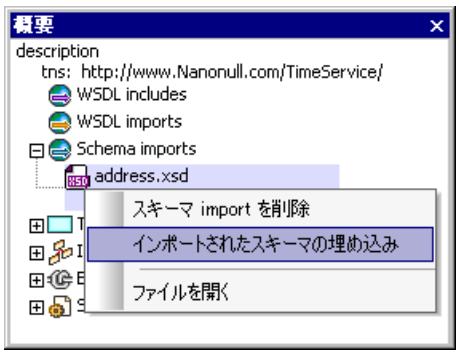
インポートされたXML スキーマはWSDL ファイルに埋め込むことができます。インポートされたスキーマを埋め込むと、そのスキーマが型 要素内のインラインスキーマとして作成され、`import` 要素が削除されます。インポートされたスキーマを埋め込むには、**Imports** リスト内にあるスキーマを右クリックして、**「インポートされたスキーマの埋め込み」**または**「インポートされた全スキーマの埋め込み」**コマンドを選択します。インポートされた全てのスキーマに対して適用される後者のコマンドは、**Imports** アイテムのコンテキストメニューからもアクセスすることができます。

WSDLインクルード、WSDLインポート、スキーマインポート (WSDL 2.0)

XML スキーマ (XSD) ファイルをインポートしたり、WSDL ファイルをアクティブなWSDL ドキュメントにインポートすることができます。ファイルをインクルードまたはインポートするには、インクルードまたはインポートするアイテム (WSDL includes、WSDL imports、Schema imports) を右クリックして、インクルードまたはインポートを加えます。インポートされたファイルの名前空間はインポートされたファイルのターゲット名前空間から自動的に生成されます。



インクルードまたはインポートされたファイルを右クリックすることで表示されるコンテキストメニューから、ファイルを削除したりXMLSpy で開いたりすることができます。ファイルはWSD L ビュー (`wsdl` ファイルの場合) またはスキーマビュー (`xsd` ファイルの場合)、そこで編集作業を行うことができます。その後、インポートされたXML スキーマをWSDL ファイルへ埋め込むことができます (以下のスクリーンショットを参照)。



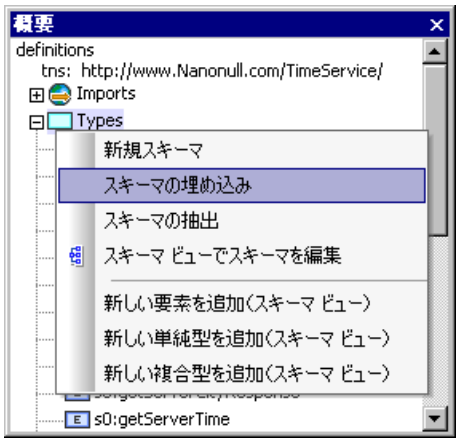
インポートされたスキーマを埋め込むと、そのスキーマが types 要素内のインラインスキーマとして作成され、import 要素が削除されます。インポートされたスキーマを埋め込むには、**Imports** リスト内にあるスキーマを右クリックして、**「インポートされたスキーマの埋め込み」**または**「インポートされた全スキーマの埋め込み」**コマンドを選択します。インポートされた全てのスキーマに対して適用される後者のコマンドは、**Imports** アイテムのコンテキストメニューからもアクセスすることができます。

Types (WSDL 1.1 ならびに 2.0)

WSDL ドキュメントで定義された全ての型を黒色で表示します。インポートされたスキーマや WSDL ドキュメント内で定義された型はグレーで表示されます。

以下の関数を利用することができます：

- **新規スキーマの作成**: **Types** アイテムを右クリックして、**「新規スキーマ」**を選択します (以下のスクリーンショットを参照)。WSDL ファイル内に埋め込まれた空のスキーマが新たに作成され、埋め込まれたスキーマとともにスキーマビューに表示されます。スキーマドキュメントの編集を完了した後に変更をスキーマファイルへ保存することで、変更が WSDL ドキュメントに書き込まれます。WSDL ドキュメントに埋め込まれたスキーマを新たに作成する際にこの機能を利用することができます。



- **スキーマの埋め込み**: **Types** アイテムを右クリックして、**「スキーマの埋め込み」**コマンドを選択します。埋め込みを行うファイルを選択するためのダイアログが表示されます。**「OK」**をクリックすることで、スキーマが types 要素内にインラインスキーマとして作成されます。選択されたスキーマが既にインポートされている場合、既にインポートされているスキーマを埋め込むか選択するように促されます。
- **スキーマの抽出**: **Types** アイテムを右クリックして**「スキーマの抽出」**を選択します。埋め込まれた各スキーマファイルが一時ファイルとしてスキーマビューにて開かれ、各々のファイルに対して名前を付けて保存ダイアログが表示されます。スキーマファイルを保存する場合、スキーマが抽出され、指定された場所に保存された後で WSDL ファイルへインポートされます。インラインスキーマとして処理されていたスキーマファイルは削除され、インポートされた外部スキーマとして処理されます。

- **スキーマビューでスキーマを編集**：埋め込まれたスキーマをスキーマビューにて編集することができます。Types アイテムまたはTypes リスト内にあるスキーマコンポーネントの名前を右クリックして、**スキーマビュー**で **「スキーマを編集」**または **「編集」（スキーマビュー）**を選択します。WSDL ドキュメントの `types` 定義から一時的なXSD ファイルが生成されます。このXSD ドキュメントはスキーマビューにて編集することができます。XSD ドキュメントの編集後にファイルを保存することで、変更点がWSDL ドキュメント内にある `types` の定義へ保存されます。変更点を保存すること無くXSD ドキュメントを閉じた場合、WSDL ドキュメントが修正されることはありません。
- **スキーマコンポーネントの追加**：XML スキーマ要素（WSDL 1.1 ならびに 2.0）、単純型（WSDL 1.1）、または複合型（WSDL 1.1）を追加することができます。**Types** アイテムまたはTypes リスト内にあるスキーマコンポーネントの名前を右クリックして、**追加**コマンドを選択します。WSDL ドキュメントの `types` 定義から一時的なXSD ファイルが生成され、スキーマビューにて表示されます。このファイルには新規コンポーネントが名前の無い状態で含まれています。このコンポーネントを編集し、ファイルを保存することで、新規コンポーネントがWSDL ドキュメントの `types` 定義へ書き込まれます。
- **スキーマコンポーネントの削除**：**Types** リスト内にあるスキーマコンポーネントを右クリックして、コンテキストメニューから **「削除」**を選択することで、スキーマコンポーネントを削除することができます。

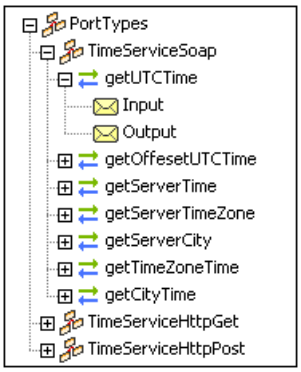
メッセージ（WSDL 1.1）

メッセージまたはそのサブコンポーネントが選択されたとき、そのプロパティーの編集を詳細入力ヘルパーにて行うことができます。その他にも、コンテキストメニューを使用して以下のことを行えます：

- 概要入力ヘルパーにてメッセージが選択されている場合、メッセージパートの追加や削除、またはメッセージの追加を行うことができます。
- 概要入力ヘルパーにてメッセージパートが選択されている場合、そのメッセージパートを削除したり、メッセージに新たなメッセージパートを追加したりすることができます。
- **「同期」**コマンドにより選択されたメッセージまたはメッセージパートに対応するポートタイプボックスをハイライトすることができます。

PortType（WSDL 1.1）

コンテキストメニューからポートタイプに関する以下の機能を使用することができます：



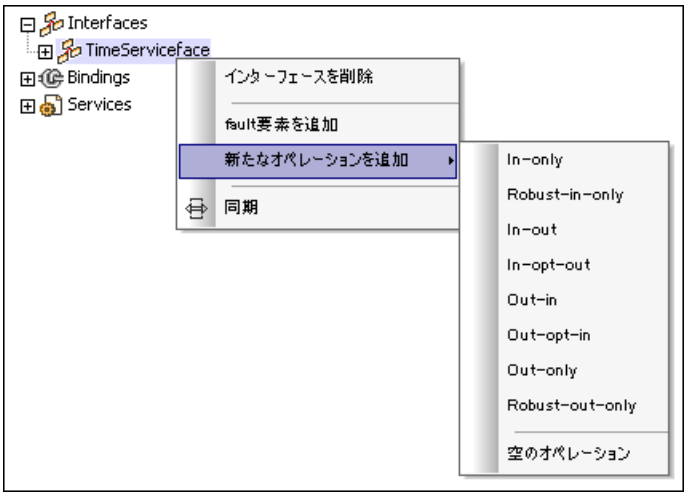
- PortTypes アイテムが選択された状態で、ポートタイプを追加することができます。
- ポートタイプが選択された状態で、ポートタイプを追加したり、選択されたポートタイプを削除したり、選択されたポートタイプにオペレーションを追加することができます。
- オペレーションが選択された状態で、新たなオペレーションを追加したり、選択されたオペレーションを削除したり、要

素 (入力、output、または fault) を選択されたオペレーションに追加することができます。
- メッセージ要素 (入力、output、または fault) が選択された状態で、新たなメッセージを追加したり、選択されたメッセージを削除することができます。
- **「同期」**コマンドにより、選択されたポートタイプ、オペレーション、メッセージがハイライトされます。

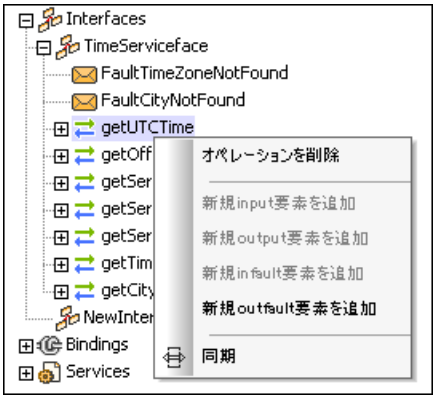
Interfaces (WSDL 2.0)

インターフェースの管理はコンテキストメニューから行うことができます。

- インターフェースを追加するには、*Interfaces* アイテムを右クリックして、メニューコマンドから **「新たなインターフェースを追加」**を選択します。
- インターフェースを右クリックすることで、(以下のスクリーンショットにあるような) ポップアップメニューが表示され、選択されたインターフェースを削除したり、選択されたインターフェースに fault やオペレーションを追加することができます。追加されるオペレーションの種類は **「新たなオペレーションを追加」**コマンドのサブメニューから選択することができます。対応するバインディングオペレーションも全て、Baindings 内に追加されます。同じように、オペレーションが削除された際にも、それを参照しているバインディングオペレーションも削除されます。



- オペレーションを右クリックすることで、(以下に表示される) コマンドが表れ、選択されたオペレーションを削除したり、(`infault` や `outfault` といった) 要素をオペレーションに追加することができます。



- メッセージ要素を右クリックすることで、選択されたメッセージを削除するポップアップメニューが表示されます。
- **「同期」**コマンドにより、デザイン内で選択されたインターフェース、オペレーション、またはメッセージがハイライトされます。

Bindings (WSDL 1.1 および 2.0)

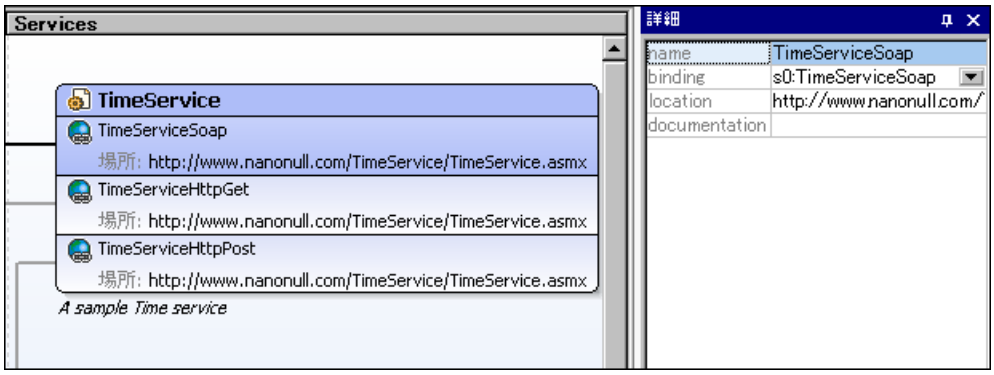
バインディングが選択された状態から、新たなバインディングを既存のバインディングに追加したり、選択されたバインディングを削除したり、選択されたバインディングにオペレーションを追加することができます。オペレーションやメッセージが選択された状態からは、PortTypes (WSDL 1.1) や Interfaces (WSDL 2.0) カテゴリー内のオペレーションやメッセージに対して利用可能なオプションを利用することができます。**同期**」コマンドをクリックすることにより、選択されたバインディング、オペレーション、そしてメッセージがハイライトされます。

Services (WSDL 1.1 および 2.0)

サービスが選択された状態から新たなサービスを追加したり、選択されたサービスを削除したり、選択されたサービスにポートを追加することができます。**同期**」コマンドをクリックすることにより、選択されたサービスやポートがハイライトされます。

2.4.3 詳細入力ヘルパー

詳細入力ヘルパーではメインウィンドウまたは概要入力ヘルパーで選択されたアイテムのプロパティが表示されます (以下のスクリーンショットを参照)。これらプロパティは詳細入力ヘルパーで編集することができます。



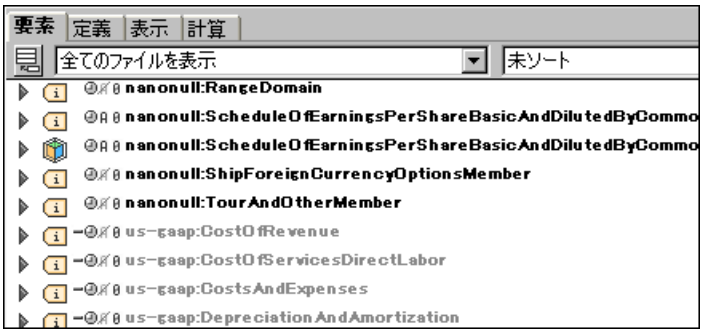
例えば上のスクリーンショットで示される様に、メインウィンドウ内で (`TimeService` サービスの) `TimeServiceSoap` ポートが選択されている場合、`TimeServiceSoap` のプロパティが詳細入力ヘルパーにて表示され、編集することができます。`name` や `description` のテキストフィールドを編集するには、フィールドをダブルクリックして、テキストの編集を行ってください。上のスクリーンショットの例にある `binding` のようなプロパティの場合、コンボボックスをクリックして、現れた選択肢の中からオプションを選ぶことができます。

2.5 XBRL ビュー

XBRL ビューはXBRL タクソノミの編集を行うことができるグラフィカルなビューです。素早い検証とエラーを発見した際にはその旨を表示し、タクソノミの開発を素早く正確に行うことができます。

XBRL ビューには以下のパーツが含まれています：

- メインウィンドウには4つのタブが表示されます：要素、定義、表示、計算 (以下のスクリーンショットを参照)。これらのタブの詳細については、この項以下のサブセクションを参照ください。



- 3つのパワフルな入力ヘルパー：概要、グローバル要素、詳細。これら入力ヘルパーにより、タクソノミファイルの管理やメインウィンドウでの編集が行ないやすくなります。
- メッセージウィンドウではタクソノミの妥当性に関するメッセージが表示されます。

このセクションではXBRL ビューのメインウィンドウならびに入力ヘルパーに関する記述と、それらの使用する方法について記述します。より詳しい情報を知るには、XBRL とXBRL メニューにあるコマンドの解説を参照ください。

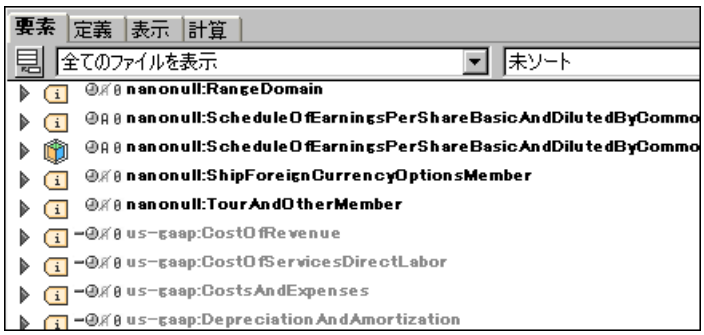
XBRL ビューにあるその他の機能

このセクションで記述される編集機能以外にも、以下の便利な機能を使用することができます：

- ドキュメントの生成によりHTML、Word、またはRTF フォーマットの詳細なドキュメントを作成することができます。
- XMLSpy の **「ファイル| 印刷」** コマンドにより、現在使用しているビューの印刷を行うことができます。

2.5.1 メインウィンドウ :要素タブ

以下のスクリーンショットに示されるメインウィンドウの **「要素」** タブには、タクソノミのコンセプトが、デフォルトではインポートされたタクソノミ内のコンセプトと共に表示されます。現在のタクソノミ内のコンセプトが黒色で表示され、インポートされたタクソノミのコンセプトは灰色で表示されます。

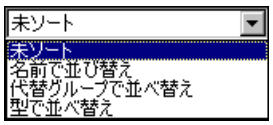


表示するタクソノミを選択

メインウィンドウタブの下にある *ファイル* コンボボックス (以下のスクリーンショットを参照) を使えば、現在のタクソノミにあるコンセプトだけを表示するのか、それともインポートされたタクソノミ内にあるコンセプトも一緒に表示するのか選択することができます。インポートされた数多くのタクソノミを表示させないだけでも編集スピードが格段に向上します。

要素の並び替え

メインウィンドウのタブ以下の右側にある *ソート* コンボボックス (以下のスクリーンショット参照) では、メインウィンドウ内に表示されている要素の並び替えを行うことができます。



並べ替えの条件は以下から選ぶことができます：

- *未ソート*：要素はドキュメントに記述されている通りに表示されます。
- *名前*：名前空間プレフィックスを要素名の一部として見なします。
- *代替グループ*：代替グループは4つあります：項目、タプル、*hypercubeItems*、*demensionItems*。
- *型で並べ替え*：XBRL 要素の Type 属性を参照します。

メインウィンドウ内での要素の検索

メインウィンドウ内で要素を検索するには、**Ctrl** + **F** のキーコンビネーションを使用します。検索ダイアログが表示されるので、目的の文字列を入力して検索を行うことができます。タクソノミで使用されている名前空間プレフィックスも検索されます。

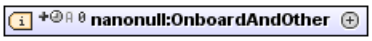
コンセプト (<element> 要素) について

各タクソノミコンセプトは XML スキーマの <element> 要素にて定義されています (以下を参照)。この定義はタクソノミドキュメントをテキストビューで見ることにより確認することができます。(インポートされたタクソノミドキュメントの定義を見るには、そのドキュメントを一度 XMLSpy で開かなければなりません)

```
<xs:element id="icuiUnrealizedHoldingLoss"
```

```
name="UnrealizedHoldingLoss"
substitutionGroup="xbrli:item"
type="xbrlim onetaryItem Type"
xbrli:balance="credit"
xbrli:periodType="instant"
abstract="false"
nillable="true"/>
```

要素タブに表示される各要素（またはコンセプト）は、その代替グループを意味するアイコン（item、tuple、hypercube、またはディメンション）と共に表示されます。更に、コンセプトの`balance`、`periodType`、`abstract`、そして`nillable`属性の値を意味するアイコンがコンセプト名の左に表示されます（以下のスクリーンショットを参照）。



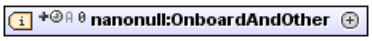
コンセプトの名前を変更するには、コンセプト名をダブルクリックして名前の編集を行ないます。

代替グループ

コンセプトの代替グループ値はコンセプトのアイコンで表示されます：

- `xbrli:item`
- `xbrli:tuple`
- `xbrldt:hypercubeItem`
- `xbrldt:dimensionItem`

以下のスクリーンショットは`substitutionGroup`値が`xbrli:item`の要素を表しています。



---

balance、periodType、abstract、nillable 属性

要素名の左にあるアイコン（上のスクリーンショットを参照）にはコンセプトのメイン属性が左から右の順序で表示されます：

- `xbrli:balance`：＋のアイコンが資産（`credit`）、－のアイコンが負債（`debit`）を表します
- `xbrli:periodType`：グレーで表示された時計のアイコンは期間（`duration`）を表し、白の時計アイコンは時点（`instant`）を表します
- `xs:abstract`：黒色の`A`アイコンは`true`で、グレーは`false`を表します
- `xs:nillable`：黒色の`0`アイコンは`true`で、グレーは`false`を表します

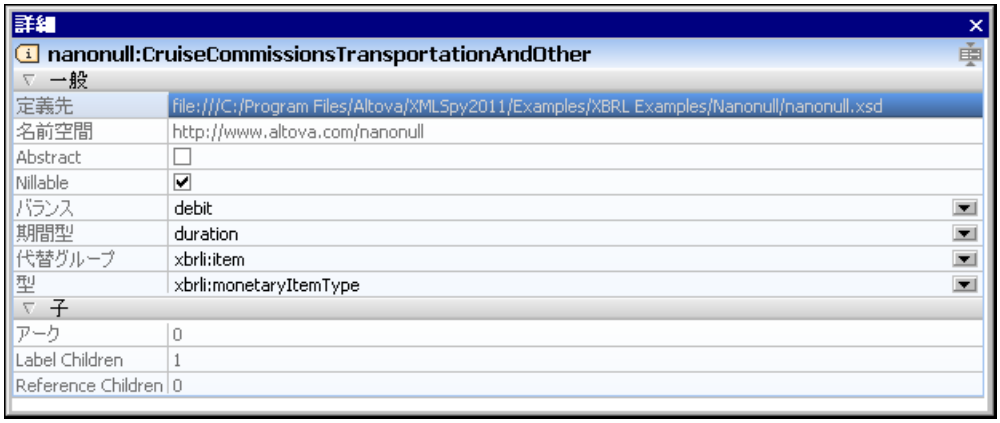
上に記された`xbrli:`属性はXBRLスキーマからのもので、XMLスキーマの`xs:`属性とは異なるものということに注意してください。

上のスクリーンショットでは、＋アイコンにより`xbrli:balance`属性の値が`credit`（貸方）であることが示されます。スクリーンショットに示されるその他の値（`xbrli:periodType`、`xs:abstract`、`xs:nillable`）は、それぞれ`duration`、`false`（abstract ではない）、`true`（nillable である）を表しています。

これら4つのアイコンのどれかにマウスカーソルを合わせると、4つ全ての属性値が表示されます。アイコンをクリックすることで、その属性に許されている値と共にリストボックスが表示され、マウスクリックにより選択を行うことで、コンセプト属性の値を素早く

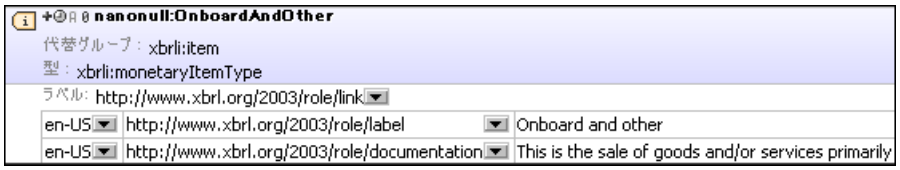
変更することができます。

これらの属性値は詳細入力ヘルパーでも編集することができます。



属性 substitutionGroup と type

要素名の左にある矢印シンボルをクリックすることで、要素の表示が展開され、`xs:substitutionGroup` と `xs:type` の2つの要素が画面に表示されます（以下のスクリーンショットを参照）。



上のスクリーンショットでは、`xs:substitutionGroup` と`xs:type` 属性の値はそれぞれ`xbrli:item` と `xbrli:monetaryItemType` となっています。これらの値は対応するコンボボックスのドロップダウンリストから別の値を選択することで編集可能です。詳細入力ヘルパー内の対応する値をダブルクリックし、キーボードから値を入力することで、両方の属性の値を変更することができます。

ラベルリンク

ラベルの子を加えるには、要素を右クリックし、コンテキストメニューから **「ラベルリンクロールの追加」** を選択します。これでラベル行が追加（以下のスクリーンショットを参照）され、ラベルプロパティ（言語、ラベルの種類、そして値）を追加、指定することができます。ラベルの展開や縮退は、ボックスの右端にある+ または－ボタンをクリックすることで行えます。

各ラベルの言語とラベルの種類プロパティはコンボボックスから値を選択することで編集することができます（上のスクリーンショットを参照）。

参照リンク

参照の子を加えるには、要素を右クリックして、コンテキストメニューから **参照リンクロールの追加** を選択します。これで参照ボックスが作成され、参照フィールドをクリックすることで参照プロパティの値を編集することができます。参照の展開や縮退は、ボックスの右端にある＋または－ボタンをクリックすることで行えます。

2.5.2 メインウィンドウ: 定義、プレゼンテーション、計算 タブ

メインウィンドウにはコンセプト間のリレーションシップを表す3つのタブが含まれています：

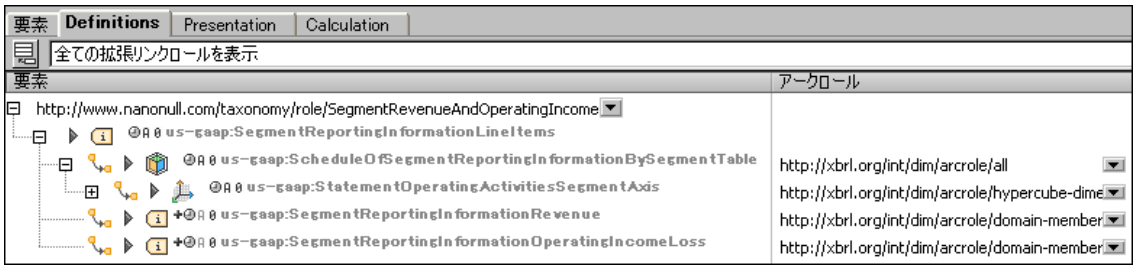
- 定義 タブにて表示される定義 リレーションシップ
- プレゼンテーション タブにて表示されるプレゼンテーション リレーションシップ
- 計算 タブにて表示される計算 リレーションシップ

各タブ (定義、プレゼンテーション、計算) では、対応するタクソノミ リレーションシップが表示され、リレーションシップをグラフィカルに修正することができます。2つのコンセプト間のリレーションシップは、あるコンセプトから別のコンセプトへアークを構築することで作成されます。この *from-to* アークは曲がった矢印として表示されます。2つのコンセプト間のリレーションシップは from から to への方向と共にアークロールとして指定されます。リレーションシップの to-end にて要素が選択されているときにアークのアークロールが詳細入力ヘルパーに表示され、定義 リレーションシップの場合、定義 タブのアークロール列に表示されます。

リレーションシップアークは、全てのリレーションシップ (定義、プレゼンテーション、計算) において同一に表示されます。このセクションでは 定義 タブにおいてリレーションシップがどのように表示されるかの簡単な説明と共に、各タブが個別に扱われます。プレゼンテーションならびに 計算 アークに関するより詳細な情報は、各セクションを参照ください。リレーションシップに関するより詳細な情報は、リレーションシップの作成: パート 1 ならびにリレーションシップの作成: パート2を参照ください。

定義 タブ

定義 タブ ではタクソノミの定義全てが表示されます。これらの定義は 定義 リレーションシップ (`.xml`) ファイル内に収められている定義 アークにより指定されます。XBRL ビューの 定義 タブには 定義 アークのセットから得られた構造が展開/縮退可能なツリー形式で表示されます (以下のスクリーンショットを参照)。



このグラフィカルな表示では、2つのエンドポイント (*from* エンドポイントと *to* エンドポイント) を持つ曲がった矢印として各 定義 アークが表示されます。2つの要素のリレーションシップの種類は要素の *アークロール* 列に表示されます。例えば、`hypercube-dimension` リレーションシップでは、リレーションシップ (またはアークロール) は対となる要素の ディメンション パートと一緒に表示されます。アークロール URI は詳細入力ヘルパーからも入力できます。

定義 リレーションシップに関する詳細は、リレーションシップの作成: パート1を参照ください。

プレゼンテーション タブ

プレゼンテーション アークには 定義 アークと同一の属性があり、同様に機能します (上の 定義 タブの項を参照ください)。アークロール URI は詳細入力ヘルパーにて入力することができます。重要な プレゼンテーション属性の1つに親要素から派生した子供の要素が表示される順序を決定する `order` 属性があります。プレゼンテーション タブではこれら子供要素は正しい順序で表示されます。子要素をリスト内の他の場所にドラッグすることで `order` 属性の値を素早く変更することができます。詳細入力ヘルパーを使っても `order` 属性の値を変更することができます (以下のスクリーンショットを参照)。

プレゼンテーション リレーションシップに関する詳細は、リレーションシップの作成 : パート 2 を参照ください。

計算 タブ

計算アークも定義 アークと同様の属性を持ち、同じように機能します (上の 定義 タブの項を参照ください)。アークロール URI は詳細入力ヘルパーを使って入力することができます。アークロールには以下の2種類があります :

- 要素の値を合計して他の要素に受け渡すアークロール
- 合計リレーションシップではなく、等価リレーションシップとして機能するアークロール

前者の場合、`weight` 属性により、どれだけ要素の値が集計要素に合計されるかが決定されます。`1.0` という属性値により、要素の値がそのまま合計されます。負の数値により、値が合計から差し引かれます。`weight` 属性の値は 計算タブまたは詳細入力ヘルパーから修正することができます。

フォーミュラ タブ

XBRL フォーミュラは フォーミュラ タブで定義および管理することができます。フォーミュラ タブは概要入力ヘルパーおよび詳細入力ヘルパーと共にフォーミュラを作成および編集することができます。 フォーミュラ間の定義とリレーションシップはダイアグラム で実行することができます。詳細はXBRL フォーミュラ エディターのセクションを参照してください。

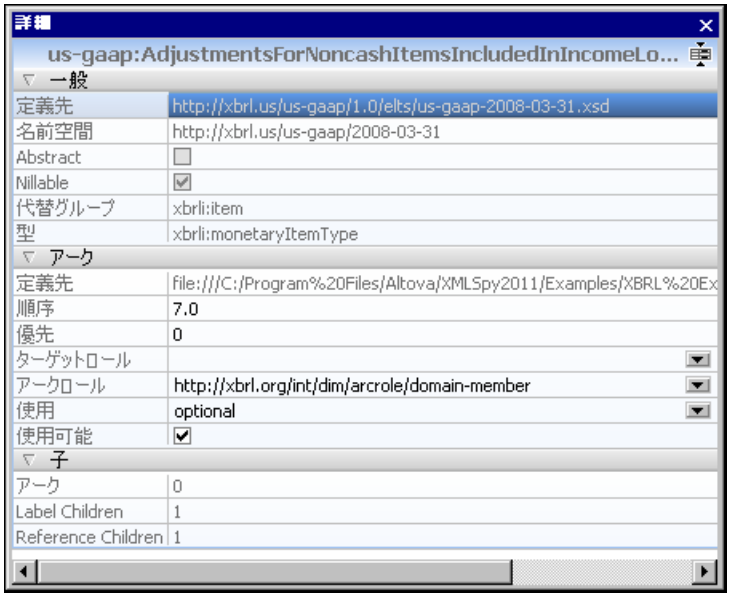
グラフィカルなビューを使った編集

以下の様な編集機能が利用可能です :

- グローバル要素入力ヘルパーから要素をドラッグして、ツリーにある要素にドラッグしてリレーションシップを作成することができます。
- アークロールの作成や編集は、 (アークロール列にある) アークロールを選択することで行えます。アークロールにて定義されたリレーションシップは `from-to` リレーションシップで、選択された要素がリレーションシップの `to-endpoint` となります。
- 要素をドラッグすることでツリー内の位置を変更することができます。これにより `order` 属性の値が変更されます。
- 要素のプロパティの編集を行うには、そのプロパティシンボルをクリックした後行うか、プロパティボックスを展開した後で、ボックス内部のプロパティを編集することができます。

詳細入力ヘルパーでの編集作業

メインウィンドウにて要素が選択されていれば、定義 アークのプロパティが詳細入力ヘルパーのアークセクションに表示されます (以下のスクリーンショットを参照)。値フィールドをダブルクリックして目的の値を入力するか、それができるのであれば) コンボボックスにて表示されるドロップダウンリストの値を選択することでこれらプロパティの値の編集を行うことができます。



その他にも、選択されている要素のアークのプロパティが詳細入力ヘルパーのアークセクションに表示されます。現在アクティブになっているタブに従って、表示されるアークは 定義 アーク プレゼンテーション アーク または 計算アークとなります。アークロールはアークロールフィールドにて入力することができます。

ラベルならびに参照リファレンスは子セクション以下に表示されます。これらのリレーションシップは要素タブにおいて編集できます。

2.5.3 XBRL ビューの入力ヘルパー

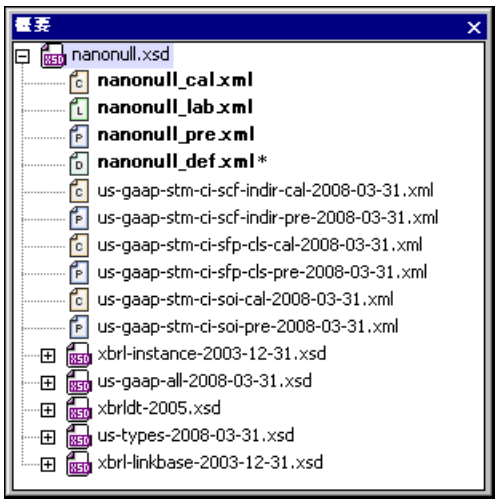
XBRL ビューには以下の入力ヘルパーが含まれています:

- 概要入力ヘルパー
- グローバル要素入力ヘルパー
- 詳細入力ヘルパー

検証情報はメッセージウィンドウに表示されます。

概要入力ヘルパー

概要入力ヘルパーにはタクソノミファイルがツリー構造で表示されます (以下のスクリーンショットを参照)。



ツリーは以下のように管理されます：

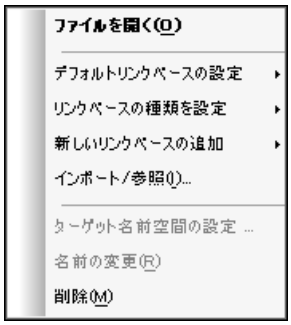
- メインコンセプトファイル (XML スキーマドキュメント) はツリーのルートに表示される現在アクティブなファイルです。
- リレーションシップリンクベースファイル (XML ファイル) はリレーションシップの種類を表す以下のような文字と色により表示されるファイルです。

  は定義リンクベースを表します。
  はラベルリンクベースを表します。
  は計算リンクベースを表します。
  は表示リンクベースを表します。
  は参照リンクベースを表します。

- インポートされたスキーマは階層の下部に表示されます。XML スキーマ (`xsd`) は全て XSD アイコンで表示されます。

概要入力ヘルパーに表示される上の情報はメインコンセプトファイルの `/schema/annotation/appinfo` 要素から取得されます。より詳細な情報については[タクソノミファイル]を参照ください。

概要入力ヘルパーにあるファイルを右クリックすると (以下のスクリーンショットに表示される) コンテキストメニューが表示され、以下のようなコマンドが使用できます：

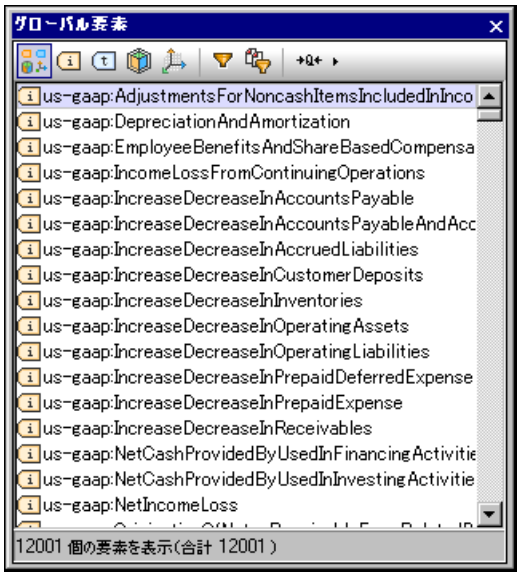


- *ファイルを開く*：選択された XML スキーマまたは XML ファイルを XMLSpy で開きます。
- *デフォルトリンクベースの設定*：単一のリレーションシップ種類 (例えば プレゼンテーション リレーションシップ) に対して複数のファイルが存在するとき、これらのうち1つをデフォルトリンクベースとしてセットすることができます。新たに作成されたリレーションシップは、そのリレーションシップのデフォルトリンクベース内に保存されます。リレーションシップのデフォルトリンクベースは太字で表示されます。
- *リンクベースの種類を設定*：選択されたリンクベースのリレーションシップの種類を指定します。目的のリレーションシップの種類をサブメニューから選択してください。**すべて**により、選択されたリンクベースは全てのリレーションシップの種類において使用されるようになります。

- *新しいリンクベースの追加* ：5つあるリレーションシップ定義、presentation、calculation、ラベル、resource）のうちの1つを含むリンクベースファイルを新たに作成します。追加されたファイル名は右クリックして表示されるコンテキストメニューの **「名前の変更」** コマンドにより変更することができます。新たに追加されたリンクベースを参照する `linkbaseRef` 要素がコンセプトファイルに新たに追加されます。
- *インポート参照* ：標準タクソノミをインポートするか、既存のXML スキーマまたはリンクベースに対する参照を作成します。標準タクソノミオプションを選択した場合、標準タクソノミのリストがダイアログに表示されます。インポートしたいタクソノミを選択してください（詳細はタクソノミのインポートを参照ください）。XML スキーマまたはリンクベースへの参照を作成した場合、コンセプトファイルに `linkbaseRef` 要素が新たに作成されます。XML スキーマまたはリンクベースオプションをクリックすると、目的のファイルを参照するためのダイアログが表示されます。
- *ターゲット名前空間の設定* ：ターゲット名前空間をセットし、`xs:schema` スキーマ要素（つまりタクソノミ全体）の中にその名前空間を加えます。「XBRL | ターゲット名前空間の設定」コマンドの詳細を参照ください。
- *名前の変更* ：選択されたファイルの名前を変更します。
- *削除* ：概要にて選択されたファイルを削除し、コンセプトファイルから `linkbaseRef` 要素を削除します。

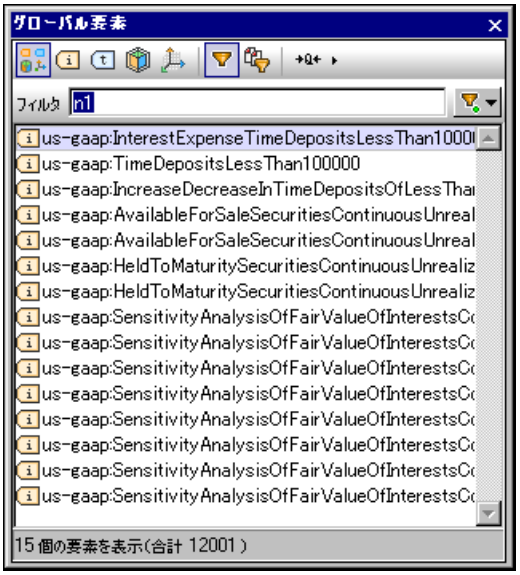
グローバル要素入力ヘルパー

（以下のスクリーンショットで示される）グローバル要素入力ヘルパーにはタクソノミ内にある全てのアイテム、Tuple、ハイパーキューブ、そして次元が表示されます。グローバル要素入力ヘルパーからメインウィンドウへ要素をドラッグすることもできます。



グローバル要素入力ヘルパーでは以下の機能を利用することができます：

- *種類によるフィルタリング* ：ツールバーにある各アイコンをクリックすることで、グローバル要素入力ヘルパーにて表示される種類（アイテム、タプル、ハイパーキューブ、次元）を切り替えることができます。*全ての要素を表示*アイコン により、全ての要素を表示します。選択されているフィルターがハイライトされます。
- *テキストによるフィルタリング* ：テキストフィルターアイコン により、テキストフィルターが表示されます（以下のスクリーンショットを参照）。



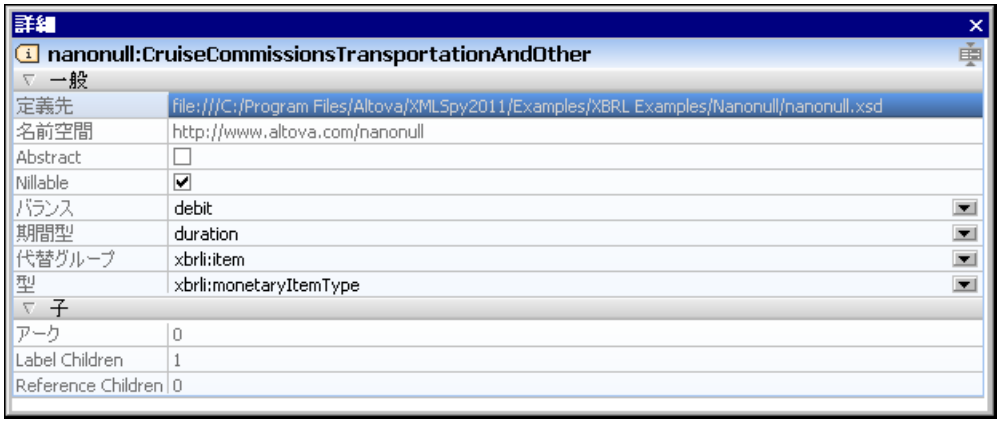
テキストフィルターが表示された後、テキストフィルターバーの右側にあるコンボボックスを使ってフィルタリングを行う対象を選択することができます。選択することのできるオプションには*含む*、*含まない*、*開始文字*があります。上のスクリーンショットの例では、*含む*オプションが選択されています。次に、フィルタリングを行う文字列をテキストフィルターボックスに入力します。入力された値に従って、フィルタリングされた結果が表示されます。上の例では、`n1` を含む要素のリストが表示されています。

フィルタリングではエントリ名をテキストとして扱うのと、エントリ名はフォーマットアイコン（下記参照）で変更することができ、フィルタリングの結果もそれによって変化することに注意してください。

- *ソースによるフィルタリング*：目的の*ソース*をポップアップから選択し、選択された*ソース*に存在する要素だけを表示することができます。
- *フォーマット*：要素名が表示される形式には3つあります：*短い有修飾名*、*展開有修飾名*、ラベル。短い有修飾名では、対応する名前空間に割り当てられたプレフィックスが使用されます。展開有修飾名（アイコン： ）では名前空間全体が使用されます。ラベルでは、要素に関連付けられたラベルが使用されます。フィルタリングは、現在表示されている要素のリストから、フィルタリングテキストにマッチしたものを表示するものなので、フォーマットの選択によりテキストによるフィルタリングオプションの結果が変わることに注意してください。
- *ドラッグアンドドロップ機能*：グローバル要素入力ヘルパーにある要素をは（定義、Presentation、Calculationといった*リレーションシップビュー*の種類に関わらず）メインウィンドウの*リレーションシップ*にドラッグすることができます。

詳細入力ヘルパー

メインウィンドウにて要素が選択されている場合、（以下のスクリーンショットにある）詳細入力ヘルパーには、そのプロパティが表示されます。



要素のプロパティによっては詳細入力ヘルパーにて編集を行うことができます：*Abstract*、*Nillable*、*バランス*、*期間型*、*代替グループ* そして*型*。

2.6 Authentic ビュー

Authentic View には、ウィンドウの上部にメニューバーとツールバーがあり、残りのインターフェイスは以下の3つの部分から構成されています： プロジェクトウィンドウ、メインウィンドウ、入力ヘルパー。これらの部分は以下に表示されています。

Authentic ビューインターフェイス の詳細は**Authentic | Authentic ビューインターフェイス**のセクションで説明されています。

2.7 ブラウザービュー

ブラウザービューは、通常以下を閲覧するために使用されます：

- 関連したXSLT ファイルをもつXML ファイル。ブラウザービューに切り替えると、XML ファイルは、関連したXSLT スタイルシートを使用して素早く変換され、結果は直接ブラウザービューに表示されます。
- HTML ファイルは、HTML として直接作成されるか、またはXML ファイルのXSLT 変換 を介して作成されます。

ブラウザービュー内のXML およびHTML ファイルを閲覧するには、**ブラウザー** タブをクリックします。

Microsoft Internet Explorer および XSLT に関する注意点

ブラウザービューは、まMicrosoft Internet Explorer 5.0 または以降を必要とします。XSLT スタイルシートにより変換されたXML ファイルを閲覧するためにブラウザービューを使用する場合は、XSLT 1.0 標準を完全にサポートするXML パーサであるMSXML 3.0 をサポートするInternet Explorer 6.0 または以降を使用することが奨励されます。MSXML 4.0.のインストールを行う場合は、ダウンロードセンターで更に詳しい情報を確認してください。 (IE 5 内のXSLT へのサポートは、公式のXSLT 勧告と 100% 互換性があるわけではありません。IE 5 を使用して、ブラウザービュー内で問題が発生した場合、IE 6 または以降にアップグレードしてください。使用中のInternet Explorer のバージョンのXSLT に対するサポートを確認してください。

ブラウザービュー 機能

次の機能がブラウザービューで使用することができます。ブラウザーメニュー、ファイルメニュー および編集メニュー を使用してアクセスすることができます。

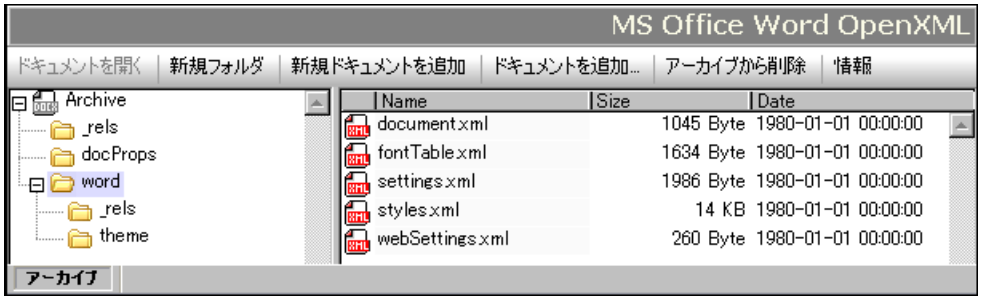
- **個別のウィンドウで開く:** ブラウザービューが個別のウィンドウである場合、同じドキュメントの編集ビューで横に並べて表示することができます。このコマンドは、**ブラウザー** メニュー内にあり、個別のブラウザービューウィンドウをタブされたウィンドウとして返す切り替えコマンドです。オプションダイアログのビュータブによりブラウザービューがデフォルトで個別のウィンドウで表示されるかを設定することができます。
- 前へ および 戻る コマンド ブラウザー コマンドは、ブラウザービュー内でロードされたページをナビゲートするために使用されます。これらのコマンドは、**ブラウザー** メニュー内にあります。
- **フォントのサイズ : ブラウザー** メニューコマンド を介して、調節することができます。
- **停止、更新、印刷 :** 標準 ブラウザー コマンドは、**ブラウザー** および **ファイル** メニュー内にあります。
- **検索 :** テキスト文字列の検索を有効化します。このコマンドは、**編集** メニュー内にあります。

2.8 アーカイブビュー

Archive View では Office Open XML (OOXML) ファイルや ZIP ファイル、または EPUB ファイルを開くことができます。OOXML や ZIP ファイル、EPUB ファイルの構造を変更するだけでなく、Archive View ではアーカイブ内にある個々のファイルを開き、XMLSpy の編集ビューにより編集を行ない、アーカイブ内に直接保存しなおすことができます。

アーカイブファイルとアーカイブビュー

(OOXML や ZIP、EPUB ファイルといった) アーカイブファイルが XMLSpy で作成または開かれたとき、ファイルは (以下のスクリーンショットにある) Archive View にて開かれます。アーカイブファイルは同時に複数個開くことができ、各アーカイブファイルが個別の Archive View ウィンドウにて表示されます。アーカイブファイルの種類は Archive View の右上に表示されます。以下のスクリーンショットに表示されているアーカイブファイルの種類は Microsoft Office Word Open XML です。



フォルダービュー

フォルダービューは Archive View ウィンドウの左側に配置され、圧縮されたアーカイブのフォルダー構造を表示します。各階層においてフォルダーはアルファベット順で表示されます。サブフォルダーを参照するにはフォルダーの左にある+シンボルをクリックしてください。フォルダーの横に+シンボルが無い場合、サブフォルダーは存在しません。フォルダー内にあるドキュメントファイルを閲覧するには、フォルダーを選択することで、メインウィンドウにファイルが表示されます。上のスクリーンショットでは、メインウィンドウにて表示されているドキュメントは `word` フォルダー内部のもので、`_rels` と `theme` の2つのサブフォルダーを含んでいます。

メインウィンドウ

メインウィンドウにはフォルダービューで選択されているフォルダー内に含まれるドキュメントが表示されます。ドキュメントはアルファベット順で表示され、圧縮していない状態でのファイルサイズと更新日時が表示されます。Archive View で表示されているドキュメントをダブルクリックすることで開くことができます。ドキュメントは別の XMLSpy ウィンドウにて開かれます。

コマンドボタン

コマンドボタンは Archive View ウィンドウの上部に位置しています。

- ***ドキュメントを開く*** : メインウィンドウ内でドキュメントが選択されているときに有効になります。クリックすることで選択されたドキュメントが開かれます。メインウィンドウに表示されているドキュメントをダブルクリックすることでもドキュメントを開くことができます。
- ***新規フォルダー*** : フォルダービューにて現在選択されているフォルダーに新たなフォルダーを追加します。フォルダービューにてフォルダーが作成されたら、すぐに名前を付けなければなりません。新しいフォルダーは、アーカイブファイルが保存されたときにアーカイブに保存されます。
- ***新規ドキュメントを追加*** : フォルダービューにて現在選択されているフォルダーに新たなドキュメントを追加します。このボタンをクリックすることで XMLSpy の新規ドキュメントを作成ダイアログが表示されます。新しく作成されたドキュメントは別の XMLSpy ウィンドウにて表示されます。フォルダービューにてファイルが選択されたら、すぐにそのファイルの名前を付けなければなりません。アーカイブファイルが保存されたとき、または編集ウィンドウにてドキュメントが保存された時に作成されたドキュメントは保存されます。
- ***ドキュメントを追加 ...*** 参照ダイアログを起動し、追加するドキュメントを選択することができます。メインウィンドウ

で現在選択されているフォルダーにドキュメントが追加され、別のXMLSpy ウィンドウにてドキュメントが開かれます。アーカイブ内のドキュメントを保存するには、ドキュメントの編集ウィンドウから保存を行うか、アーカイブファイルを保存する必要があります。

- ***アーカイブから削除*** :メインウィンドウ、またはフォルダービューにて選択されたドキュメントやフォルダーをアーカイブから削除します。アーカイブファイルが保存されるまで、変更は有効になりません。
- ***情報*** :情報ウィンドウの表示を切り替えます。以下を参照ください。

情報ウィンドウ

情報ボタンをクリックすることで、情報ウィンドウの表示を切り替えることができます。情報ウィンドウでは、アーカイブに含まれるファイルの数や、圧縮時と非圧縮時のファイルサイズ、また圧縮率といったアーカイブファイルに関する一般的な情報が表示されます。

3 XML

このセクションではXMLSpy にてXML ドキュメントを操作する方法が記述されています。XML に関する以下のような側面がカバーされます：

- XML ドキュメントを作成、開く 保存する方法。このセクションではファイルの作成に関するXMLSpy の重要な設定についても説明されます。
- XML ドキュメントはテキストビュー、グリッドビュー、またはAuthentic ビューにて編集することができます。使いやすいビューを選択して、編集時にビューの切り替えを行うことができます。各ビューではそれぞれのビューが持つ利点を活かすことができます。
- XML ドキュメントの入力ヘルパーが持つ独自の機能に関する解説。
- XSLT やXQuery を含むXML ドキュメントの操作方法。これらの処理を行うのに関連したXMLSpy が持つ様々な機能が説明されます。PDF フォントに関するセクションでは、PDF 出力を生成する際のフォントの処理方法について説明されます。
- XML ドキュメントで作業をする際に使用できるその他の機能を紹介します。

Altova Web サイト： XML エディター

3.1 XML ドキュメントを作成、開く、保存する

XML ドキュメントを作成、開く、そして保存するときには、以下のことがらが関連します：

- テキストビュー、グリッドビュー、Authentic ビューのうち、どのビューで XML ドキュメントは開かれているのか
- 新規 XML ドキュメントが作成されたとき、スキーマ（XML スキーマまたは DTD）が自動的に、もしくは手動で割り当てられるか、または割り当てられないのか
- XML ドキュメントにスキーマが割り当てられている場合、ドキュメントを開いたり保存するときに検証は自動的に行われるのか

デフォルトビュー

アプリケーション全体で有効になる設定により、新規または既存に関わらず）XML ファイルが開かれるビューを指定することができます。これらの設定はオプションダイアログ **「ツール | オプション」** に収められています。

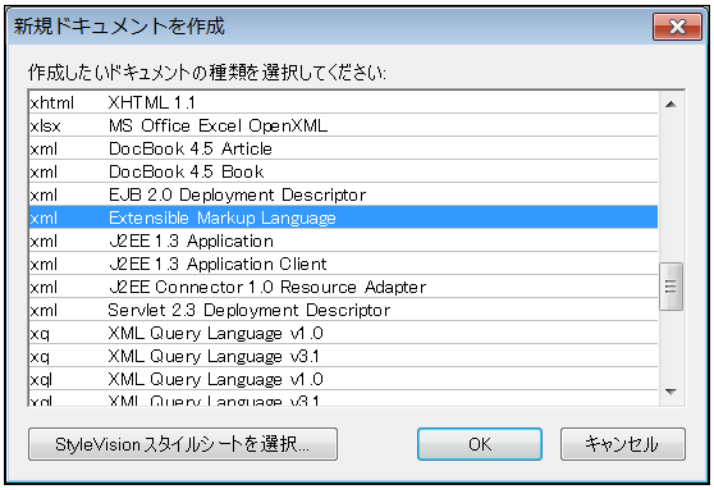
オプションダイアログの ***ファイルタイプ*** タブにて .xml を選択し、デフォルトビューから目的の編集ビュー（テキストビューまたはグリッドビュー）を選択します。スキーマビューならびに WSDL ビューは XML スキーマと WSDL ドキュメントでしか選択できないことと、ブラウザービューは表示のためのビューであり、編集ビューでは無いことに注意してください。

ファイルタイプ タブでは、選択されたファイルの種類に対して XMLSpy をデフォルトのエディターに設定することもできます。

StyleVision Power Stylesheet (SPS) ファイルが割り当てられている場合、Authentic ビューでも XML ドキュメントを編集することができます。割り当てられた SPS ファイルと一緒に XML ファイルが開かれた場合、Authentic ビューにで直接開く様に設定することができます。オプションダイアログの ***ビュー*** タブにある、*常にファイルを Authentic ビューで開く* にチェックを入れてください。このオプションにチェックが入っていない場合、***ファイルタイプ*** タブの .xml ファイルで指定されたデフォルトビューにてファイルが開かれます。

スキーマの割り当て

新規 XML ファイルを作成するには、**「ファイル | 新規作成」** メニューコマンドを選択してください。新規ドキュメントを作成ダイアログ（以下のスクリーンショット）が表示されます。



XML ドキュメントとして選択できるアイテムが複数あることに注意してください。*Extensible Markup 言語* として記述されているオプションにより一般的な XML ドキュメントが作成されます。その他のオプションでは、例えば DocBook といったスキーマと、XML ドキュメントが既に関連付けられていたりします。関連付けされているこれらのオプションを選択すると、() 対

応するスキーマが自動的に割り当てられ、(i) 割り当てられたスキーマにより妥当なドキュメントのスケルトンがデフォルトで XML ドキュメントに作成されます。XML ドキュメントのスケルトンは独自に作成することができます。アプリケーションフォルダーの `Template` フォルダーに保存すれば、スケルトンドキュメントが新規ドキュメントを作成ダイアログにて使用できるようになります。

一般的な Extensive Markup 言語 をドキュメントの種類として選択した場合、ドキュメントにスキーマ (DTD または XML スキーマ) を割り当てるかどうか質問されます。ここでは、スキーマを参照するか、スキーマを割り当てること無く XML ドキュメントの作成にとりかかることができます。

勿論、**DTD / スキーマ** メニューを使えば、ドキュメントの編集中いつでもスキーマを割り当てることができます。

自動検証

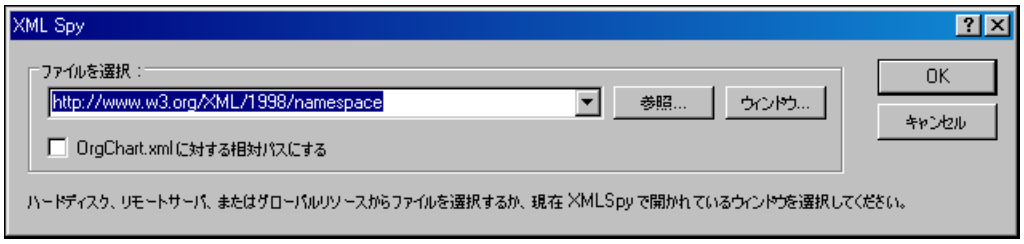
既存の XML ドキュメントにスキーマが割り当てられている場合、ドキュメントを開いたり保存する際に自動的に検証を行うことができます。オプションダイアログ（**ツール | オプション**）の**ファイル** タブから設定を行うことができます。

ファイル タブ内の自動検証設定をファイルタイプタブの設定と組み合わせることで、特定のファイルタイプを自動的に検証することができます。これら2つのタブにある設定を行うことで、

3.2 スキーマの割り当てと検証

Altova ウェブサイト： XML バリデーター、XML 検証

XML ドキュメントの作成時にスキーマ (DTD または XML スキーマ) の割り当てを行うことができます。スキーマの割り当てや変更は、**「DTD / スキーマ」**メニュー以下にある**「DTD の割り当て」**または**「スキーマの割り当て」**コマンドにより、いつでも行うことができます。



ダイアログ内の相対パスにチェックを入れることで、XML ドキュメントへ挿入されるスキーマファイルへのパスを XML ファイルから見た相対パスにすることができます。

スキーマのグローバルリソース

グローバルリソースとはファイルやフォルダーへのエイリアスです。ターゲットファイルまたはフォルダーは、（**「ツール | アクティブな構成」**により）グローバルリソースのアクティブ構成を変更することでグラフィカルに変更することができます。グローバルリソースを使えば割り当てられた複数のスキーマの切り替えを瞬時に行うことができ、試験を行うときなどに力を発揮します。グローバルリソースの使用方法については、Altova グローバルリソースを参照ください。

XML スキーマと DTD

XML スキーマには無い DTD の便利な機能の 1 つにエンティティがあります。もし検証されたスキーマにてエンティティを使用したい場合は、DOCTYPE 宣言を XML ドキュメント内で行ない、その中でエンティティの宣言を行う必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrgChart [
  <!ENTITY name-int "value">
  <!ENTITY name-ext SYSTEM "extfile.xml">
]>
<OrgChart xmlns="http://www.xyz.com/org"
        xsi:schemaLocation="http://www.xyz.com/org OrgChart.xsd">
  ...
</OrgChart>
```

DTD 内部でエンティティの宣言を行った後に、それらを XML ドキュメント内で使用することができます。ドキュメントは整形式ならびに妥当なものとなります。しかし外部で解析されたエンティティは Authentic ビューでサポートされません。

スキーマの定義へ移動

XML ドキュメントが開かれていれば、**「DTD/スキーマ」**メニュー以下の**DTD へ移動**または**スキーマへ移動**コマンドをクリックすることで、ベースとなっている DTD または XML スキーマを直接開くことができます。更に、XML ドキュメント内のノードにカーソルを移動させ、**「DTD/スキーマ | 定義へ移動」**コマンドを使うことで、ノードのスキーマ定義に直接移動することができます。

検証と整形式のチェック

ドキュメントの検証や整形式のチェックを行うには、**検証 (F8)** や**整形式のチェック (F7)** コマンドを XML メニューまたはツールバーから使用します。エラーや警告がある場合は、メッセージウィンドウに表示されます。XML ドキュメントが不正の場合、スキーマ内にある情報をベースに、スマートフィックスによるエラーの修正を行うことができます。

3.3 テキストビューでの XML 編集

XMLSpy のテキストビューでは、編集ビューセクションのテキストビューにて記述されている一般的に使用可能な編集機能と共に、XML に特化したテキスト編集機能を利用することができます。

- テキストのコメントアウトやアンコメント
- XML 文字のエスケープとエスケープ解除
- ファイルパスの挿入
- XInclude による XML フラグメントの挿入
- XPath ならびに XPointer 表現をクリップボードにコピー

テキストのコメントアウトとアンコメント

XML のコメント開始ならびにコメント終了文字列である <!-- と --> を使用して、XML ドキュメント内のテキストをコメントアウトすることができます。XMLSpy では **編集 | コメントの IN/OUT** メニューコマンドを使用することで、これらのコメント文字列を簡単に挿入することができます。

コメントアウトしたいテキストを選択して、**編集** メニューまたは選択したテキストを右クリックして表示されるコンテキストメニューにある **「コメントの IN/OUT」** を選択することで、テキストのブロックをコメントアウトすることができます。コメントアウトされたテキストはグレーで表示されます（以下のスクリーンショットを参照）。

```
<Department>
    <Name>Administration</Name>
    <Person>
    <Person>
    <Person>
    <!--<Person>
        <First
        <Last></Last>
        <PhoneExt></PhoneExt>
        <EMail></EMail>
        <LeaveTotal></LeaveTotal>
        <LeaveUsed></LeaveUsed>
        <LeaveLeft></LeaveLeft>
    </Person>-->
</Department>
```

コメントの区切り文字を**除いて**コメントアウトされたブロックを選択し、**編集** メニューまたは選択したテキストを右クリックして表示されるコンテキストメニューにある **「コメントの IN/OUT」** を選択することで、コメントアウトされたテキストをアンコメントすることができます。コメントの区切り文字が取り除かれ、テキストが元の色に戻ります。

空の行に対する注意

XML ドキュメントにおいて、空の行はビューの変更またはドキュメントの保存時に破棄されます。空の行を保持したいのであれば、コメント内に記述してください。

XML 文字のエスケープやエスケープの解除

（以下のテーブルに示される）XML で使用される5個の特殊な文字を、対応するエンティティ参照にエスケープ、またはエスケープの解除を行うことができます。テキストのブロックをハイライトして、**「XML 文字のエスケープ」** または **「XML 文字のエスケープ解除」** コマンドをコンテキストメニューから選択します。選択されたコマンドに従い、そのテキストブロック内にある XML 特殊文字がエスケープ、またはエスケープ解除されます。

<	<
>	>
&	&
'	'
"	"

例：

<a> という文字列に対して **「XML 文字のエスケープ」** コマンドを適用すると、 <a> という文字列への変換が行われ、
<a> という文字列に対して **「XML 文字のエスケープ解除」** コマンドを適用することで、文字列が <a> に変換されます。

ファイルパスの挿入

編集 | ファイルパスを挿入 コマンドにより、目的のファイルを参照し、そのファイルパスを XML ドキュメント内で選択されている場所に挿入することができます。このコマンドによりファイルパスを素早く正確に入力することができます。詳細については コマンドの記述を参照ください。

XInclude により XML フラグメントを挿入

編集 | XInclude を挿入 により、XInclude により XML ドキュメント全体のコンテンツまたはフラグメントを編集中の XML ドキュメントに挿入することができます。
このコマンドにより、XML ドキュメント全体、または XML ドキュメントのフラグメントを XInclude メカニズムの XPointer 拡張を介して素早く正確に入力することができます。詳細については、 コマンドの記述を参照ください。

XPath ならびに XPointer 表現をクリップボードにコピー

編集 | XPath をコピー および **編集 | XPointer をコピー** を使うことで、選択されているノードの XPath と XPointer 表現をそれぞれクリップボードにコピーすることができます（ノードの識別は XML ドキュメント内の位置により行われます）。これにより目的とするノードの正確な XPath ならびに XPointer 表現を取得することができます。

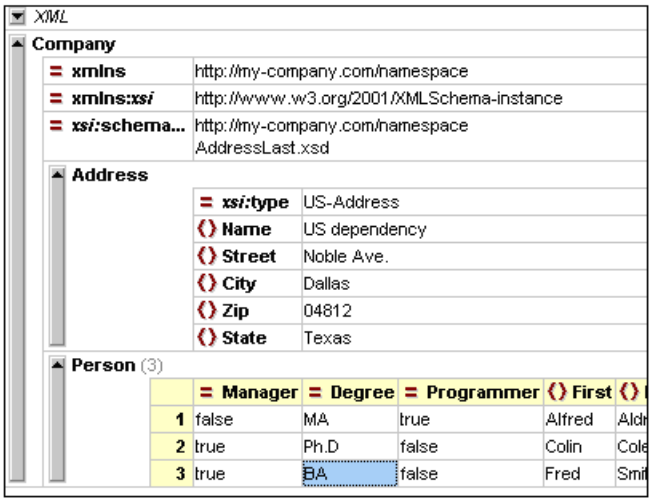
例えば、 Offices というドキュメント要素以下にある3番目の Office 要素がテキストビューまたはグリッドビューにて選択されている場合を考えてみましょう。この場合 /Offices/Ofice[3] という XPath 条件式がコピーされます。3番目の Office 要素の前に、別の名前の兄弟要素が出現しない場合、 element(/1/3) という XPointer がコピーされます。

コピーされた条件式をお任意の場所に挿入することができます。例えば、XPath 条件式を XSLT スタイルシートへ挿入したり、XPointer 条件式を xinclude 要素の href 属性内に挿入することができます。

コマンドのより詳細な情報については、ユーザーリファレンス内の対応する項を参照ください。

3.4 グリッドビューでの XML 編集

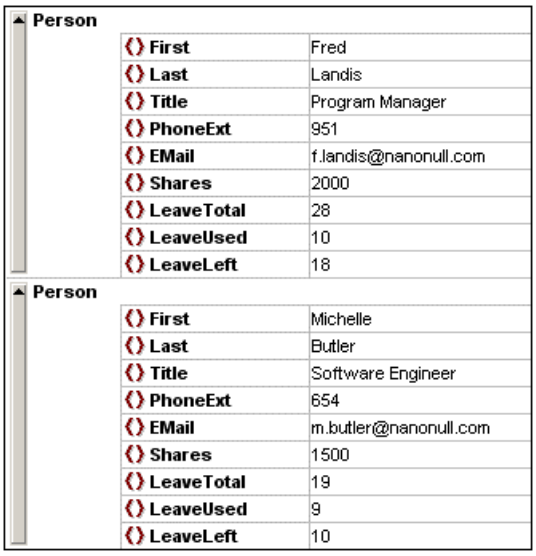
グリッドビューには**XML ドキュメント**が階層構造で表示され、ネストされているコンテナーは展開/縮退することができます。グリッドビューによりドキュメントの構造を確認することができ、ドキュメントの構造やコンテナーを簡単に編集することができます。



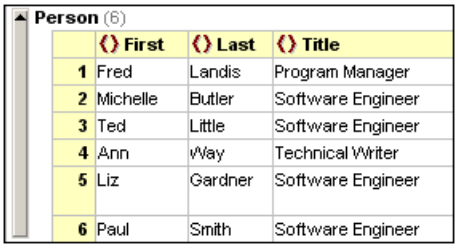
上のスクリーンショットでドキュメントが階層的に表示されていることに注目してください。ノードにコンテンツが収められている場合、名前とコンテンツの 2 つのフィールドに分割、表示されます。名前は太字で表示され、ノードコンテンツは標準の書体で表示されます。

テーブルとして表示

要素内で複数のインスタンスが繰り返し現れるような場合、標準のグリッドビューでは各インスタンスがドキュメント内の順序に従い 1 つずつ表示されます（以下のスクリーンショットを参照）。



このような複数のインスタンスが出現するような構造は（以下のスクリーンショットの様に）テーブルとして表示することもできます。各インスタンスの子要素が列として、複数のインスタンスが行として表示されます。



Person (6)

	First	Last	Title
1	Fred	Landis	Program Manager
2	Michelle	Butler	Software Engineer
3	Ted	Little	Software Engineer
4	Ann	Way	Technical Writer
5	Liz	Gardner	Software Engineer
6	Paul	Smith	Software Engineer

テーブルビューによりユニークで高度な編集機能が提供され、テーブル内にある行ならびに列を他の列や行と連動する形で操作することができます。これにより、インスタンスの子要素内にある値を元にデータの並べ替えを行ったりすることができるようになります。上のスクリーンショットの例では、`Last` 子要素に格納されている値を元に、6つある`Person` 要素を、マウスを使った操作により並べ替えることができます。このような操作により、通常 XML ノードセットの並び替えに使われるXSLT による変換と比較して、とてもシンプルに目的を達成することができます。

ドキュメントの構造を編集

グリッドビューでは、XML ドキュメント構造をグラフィカルに編集することができます。例えば、ドキュメント構造の各構造セグメントを展開/縮退したり、ノードの挿入、追加、削除、ノードをドラッグアンドドロップすることで位置を変更したり、ノードの型を変換したりといった操作を行うことができます。

「XML」メニューにより空の子ノードを挿入、追加することができます。例えば、ドキュメント内の要素を選択して子を追加するメニューオプションから空の子要素を追加することができます。新しく追加されたノードの適当なフィールドをダブルクリックすることで、要素の名前とコンテンツを入力することができます。

「要素」ならびに **「属性入力ヘルパー」**により、選択されている場所で許されている子ノードの挿入、追加を行うことができます。例えば、メインウィンドウにてノードを選択すると、その場で
挿入、追加してもドキュメントの妥当性を損なわない要素ならびに属性ノードが、要素、そして属性入力ヘルパーにそれぞれ表示されます。

選択されたノードに対して使用することのできるコマンドは、XML メニューならびに入力ヘルパーの他にもコンテキストメニューからも使用することができます。

コンテンツの編集

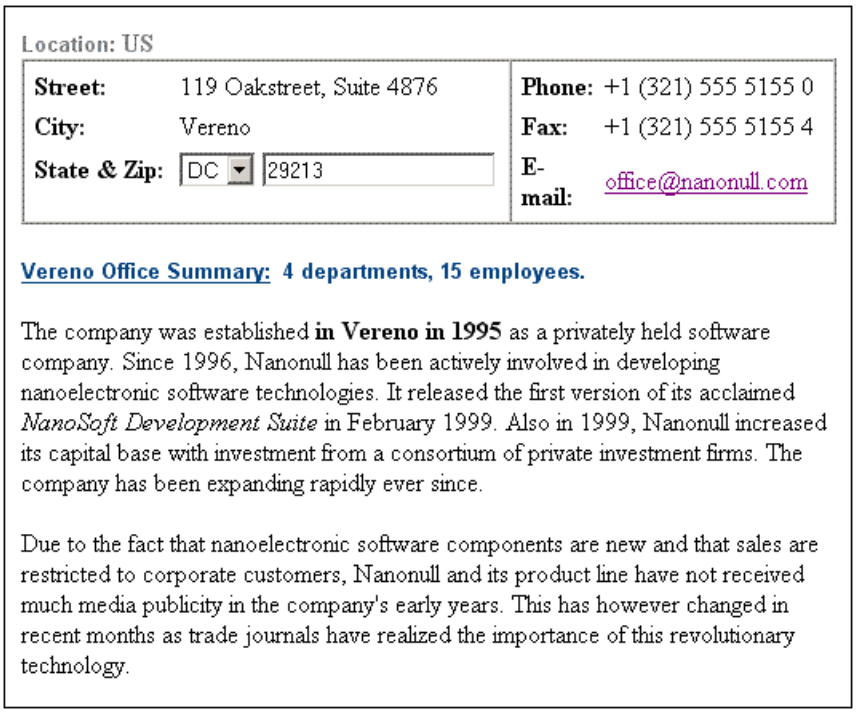
コンテンツを編集するには、コンテンツフィールドをダブルクリックし、目的のコンテンツテキストを入力してください。エンティティの挿入はエンティティ入力ヘルパーから行うことができます。

グリッドビューについて更に詳しく

グリッドビューに関する更に詳しい情報は編集ビュー以下のグリッドビューを参照ください。

3.5 Authentic ビューでの XML 編集

Authentic ビューではXML ドキュメントを (以下のスクリーンショットにあるような) テキストドキュメントの様に編集することができます。XML マークアップやその他のコンテンツには関係ないテキストはドキュメントの編集時に隠すことができます。これによりXML に慣れ親しんでいない人がドキュメントのコンテンツだけに集中しながらも、XML ドキュメントの妥当性を保つことができます。



ドキュメントをAuthentic ビューで表示するには、StyleVision Power Stylesheet (SPS) がXML ドキュメントに割り当てられている必要があります。SPS はXML ドキュメントが参照しているスキーマと同一のスキーマをベースにしており、XML ドキュメントの構造を定義します。SPS ではAuthentic ビューで表示されるドキュメントのレイアウトやフォーマットも定義されます。例えば、上のスクリーンショットに表示されているドキュメントには、以下のAuthentic フォーマットならびに編集機能が使用されています：

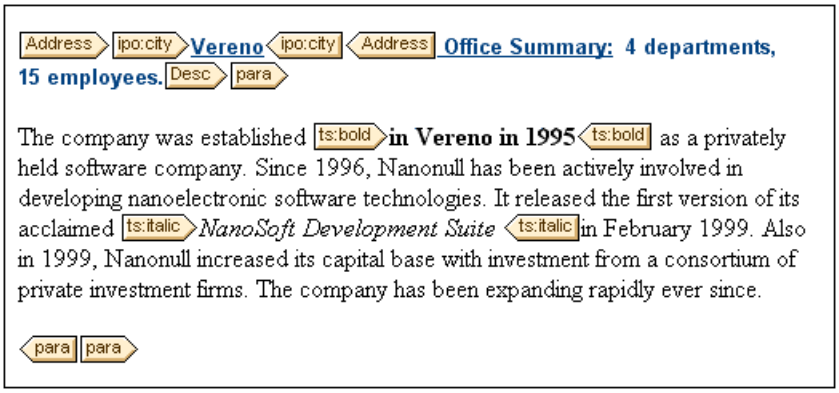
- パラグラフなどのブロックフォーマット
- テーブル構造
- 色やフォントの種類といったテキストフォーマット
- (State & Zip フィールドにある) コンボボックスにより、妥当な選択肢の中から値を選択することができます。選択肢は上で記述されているように、スキーマ内にある列挙 から取得されます。
- ドキュメント内のデータから計算を行ない表示することができます。上の例では、Office Summary の情報はユーザーによって入力されたものではなく、ドキュメント内のデータから計算して得られたものです。

SPS はXML ドキュメントをAuthentic ビューにて閲覧、そしてXML から (HTML、PDF、RTF、そしてWord 2007 といった) 標準出力を生成するためのファイルです。SPS ファイルはAltova StyleVision により作成されます。

ドキュメントの構造を編集

位置を選択し、(要素や属性) 入力ヘルパー、またはコンテキストメニューから目的のノードを選択することで、妥当なノードをいつでもドキュメントに挿入することができます。カーソルの位置を変更することで、選択できるノードがその場所において兄弟または子要素として妥当性を失わずに挿入することができるものに制限されます。例えば、カーソルがパラグラフ内に位置している場合、スキーまでそれが許されていれば、新たなパラグラフを追加することができます。

XML ドキュメントの構造をAuthentic ビューで編集する場合、ドキュメントのマークアップを参照できれば便利です。メニューオプションの **「Authentic | 大きなマークアップを表示」** (または対応するツールバーアイコン) を使用することで、以下のスクリーンショットにあるように、マークアップをタグとして表示することができます。



コンテンツの編集

ドキュメントのノード内に入力することでコンテンツの作成や編集を行うことができます。エンティティや CDATA セクションはコンテキストメニュー (そしてエンティティはエンティティ入力ヘルパーからも) から追加することができます。

Authentic ビューにおける編集の詳細

Authentic ビューにおける編集に関する詳細については、Authentic ビューセクションを参照ください。

3.6 XML ドキュメントの入力ヘルパー

XML ドキュメントでは、3つの入力ヘルパーが用意されます：要素入力ヘルパー、属性入力ヘルパー、そしてエンティティ入力ヘルパーです。要素入力ヘルパーにより要素が追加された場合、オプションダイアログの編集タブで指定された設定に従い、必須の子要素や必須の属性、または全ての子要素を同時に自動的に追加する（または全く追加しない）ことができます。空の属性が追加された場合、引用句（""）と共に追加されます。

他のビューでは入力ヘルパーの表示も、そのビューが持つ機能に合わせて変化することにご注意してください。

要素入力ヘルパー

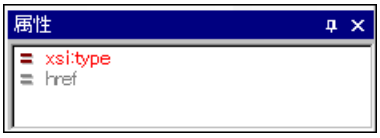
以下の点に留意してください：

- *テキストビュー*：カーソルの挿入ポイントに要素は挿入されます。使用されていない要素は赤色で、使用されているものはグレーで表示されます。必須要素は感嘆符（!）が要素名の前に表示されます。
- *グリッドビュー*：選択された要素の後に追加、前に挿入、または子要素として追加することができます。そのため追加することのできる要素は3つの異なるタブに表示されます。使用されていない要素は黒色で、使用された要素はグレーで表示されます。
- *Authentic ビュー*：選択された要素の後、前、そして中に要素を挿入することができます。更に、ドキュメント内で現在選択されている要素がドキュメントのツリー構造にて表示されます。Authentic ビュー内での詳しい編集方法については、Authentic ビューの項を参照ください。

属性入力ヘルパー

以下の点に留意してください：

- *テキストビュー*：カーソルが要素の開始タグ内にあるスペースの後にある場合、その要素に対して定義された属性が表示されます。使用されていない属性は赤色で、使用された属性はグレーで表示されます。必須属性は感嘆符（!）が属性名の前に表示されます。



目的の属性をダブルクリックすることで、属性を挿入することができます。カーソルの位置に属性が、等号記号（=）と引用句（""）と共に挿入されます。カーソルは等号句の間に配置され、属性値を直ちに入力することができます。

- *グリッドビュー*：要素が選択されているとき、子として追加することのできる属性が「子を追加」タブに表示されます。属性が選択されたときには、使用可能な属性が「追加」ならびに **「挿入」** タブに表示されます。使用されていない属性は黒色で、使用された属性はグレーで表示されます。
- *Authentic ビュー*：要素が選択されているとき、その要素に対して宣言されている属性が表示されます。入力ヘルパーにて属性の値を入力してください。

エンティティ入力ヘルパー

インライン（XML ドキュメント内）または外部 DTD で宣言されたエンティティはエンティティ入力ヘルパーに表示されます。3つのビュー（テキスト、グリッド、Authentic）にて、入力ヘルパー内のエンティティをダブルクリックすることで、そのエンティティがカーソル位置に挿入されます。グリッドビューでは、エンティティは追加、挿入、子を追加タブに表示されます。

内部エンティティを追加した場合、そのドキュメントを保存して再度開かないと、エンティティはエンティティ入力ヘルパーに表示されません。

3.7 XSLT とXQuery による処理

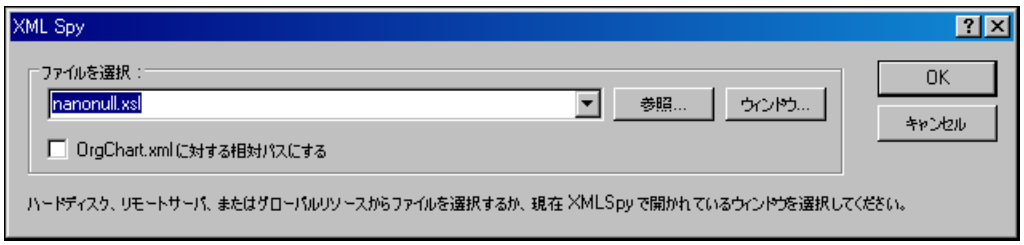
XML ドキュメントとXSLT またはXQuery ドキュメントを組み合わせることで、出力ドキュメントを生成することができます。XMLSpy にはXSLT 1.0、XSLT 2.0、そしてXQuery 1.0 プロセッサーが内蔵されており、GUI から以下のような機能を利用することができます：

- XSLT スタイルシートの割り当て
- XSLTへ移動
- XSLT パラメーターとXQuery 変数
- XSLT 変換
- XQuery 実行
- RaptorXMLを使ったXML 処理の自動化

XSLT スタイルシートの割り当て

「XSLT/XQuery | XSL の割り当て」コマンドを使用（以下のスクリーンショットにあるように、ダイアログにてファイルを指定）して、XSLT スタイルシートをXML ドキュメントに割り当てることができます。割り当てはW3C により定義された `xml-stylesheet` 処理命令（PI）としてXML ドキュメント内に記述されます。
XSLT 変換が呼び出された際（**「XSL/XQuery | XSL 変換」**）に、この割り当ては使用されます。

更に、**「XSL/XQuery | XSL:FO 変換」**コマンドにより（以下のスクリーンショットにあるようにダイアログにてファイルを指定して）XSLT-FO スタイルシートを割り当てることもできます。割り当てはAltova により定義されたaltova_xslfo 処理命令（PI）としてXML ドキュメント内に記述されます。
XSLT-FO 変換が呼び出された際（**「XSL/XQuery | XS:FO 変換」**）に、この割り当ては使用されます。



XSLT ファイルのリソースとしてグローバルリソースを選択することもできます。グローバルリソースはファイルまたはフォルダーに対するエイリアスです。アクティブなグローバルリソースの構成をマウス操作（メニューコマンド**「ツール | アクティブな構成」**）により変更することで、ターゲットファイルまたはフォルダーを変更することができます。グローバルリソースを使うことで、割り当てられたXSLT ファイルを切り替え、試験などに利用することができます。詳しいグローバルリソースの使用方法に関しては、Altova グローバルリソースの項を参照ください。

以前の割り当てにより作成された上記の処理命令が存在する場合、それに既存の割り当てを上書きしても良いか確認されます。

XSLT へ移動

「XSL/XQuery | XSL へ移動」コマンドにより、XML ドキュメントに割り当てられたXSLT ファイルを開くことができます。

XSLT パラメーターとXQuery 変数

「XSL/XQuery | XSLT パラメーター/XQuery 変数」コマンドを選択することで表示されるダイアログにより、XSLT パラメーターならびにXQuery 変数の定義、編集、そして削除を行うことができます。ここで定義されたパラメーター、変数の値は、XMLSpy における全てのXSLT 変換ならびにXQuery 実行により使用されます。しかし、これらの値はMSXML の様な外部エンジンには受渡されません。この機能の詳細については、ユーザーリファレンスの項を参照ください。

XSLT 変換

2種類のXSLT 変換を利用することができます：

- 標準 XSLT 変換（**「XSL/XQuery | XSL 変換」**）変換の出力は新規ウィンドウに表示されるか、スタイルシートで指定されている場合、指定された場所にファイルが保存されます。変換のために使用されるエンジンはオプションダイアログ（**「ツール | オプション」**）のXSL タブにて指定することができます。
- XSL-FO 変換（**「XSL/XQuery | XSL:FO 変換」**）：XML ドキュメントは2段階の処理によりPDF へと変換されます。最初のステップでは、オプションダイアログ（**「ツール | オプション」**）のXSL タブで指定されたXSLT プロセッサーによりXML ファイルがFO ドキュメントに変換されます。オプションタブの下部にてFOP の様なFO プロセッサーに同封されたXSLT エンジンも使用することができます。2番目のステップでは、オプションダイアログ（**「ツール | オプション」**）のXSL タブで指定されたFO プロセッサーによりFO ドキュメントからPDF 出力が生成されます。

注意： XSL: FO 変換コマンドにより、（XML ドキュメントの一種である）FO ドキュメントをPDF に変換することができます。変換のソースドキュメントがFO ドキュメントである場合、上で紹介された 2段階処理の 2番目のステップが直接実行されます。

XQuery 実行

「XSL/XQuery | XQuery 実行」コマンドをクリックすることで、アクティブなXML ドキュメント上でXQuery ドキュメントを実行することができます。XQuery ファイルを入力するよう促され、実行結果のドキュメントは新しいウィンドウに表示されます。

RaptorXML を使ったXML 処理の自動化

Altova RaptorXML はXML 検証、XSLT 変換、およびXQuery 実行を搭載したアプリケーションです。コマンドライン、COM インターフェース、Java プログラム、そして .NET アプリケーションから使用することができ、XSLT 変換タスクはRaptorXML により自動化することができます。例えば、RaptorXML によりドキュメントのセットを変換するバッチファイルを作成することができます。詳細についてはRaptorXML ドキュメンテーションを参照ください。

3.8 PDF フォント

フォーマッターとPDF ビューアーに対するフォントの使用

例えばFOP といった) フォーマッターによりPDF が作成され、PDF ビューアー 例えばAdobe Reader) にて確認することができます。

PDF のレイアウトを行うには、ドキュメント内で使用されているフォント、その中でも特に、使用されている文字の幅に関する詳細をフォーマッターへ伝える必要があります。この情報を使用することで行の長さやハイフンの位置、行端揃えなどの計算が行われます。この情報はフォントメトリックと呼ばれ、各フォントとともに保管されます。フォーマッターによっては、システムのフォントフォルダーからメトリックを直接読み込むことができますが、(FOP のような) それ以外のフォーマッターに対しては、定められた形式でメトリックを渡す必要があります。フォントのメトリックがフォーマッターに対して与えられると、期待されるかたちでPDF のレイアウトが作成されることになります。そのため、ドキュメントにて使用されている全てのフォントに対するメトリックファイルが、使用しているフォーマッターに対して渡されるようにする必要があります。

フォーマッターはフォントを参照することができるほか、PDF ファイル内部にフォントを埋め込むことができます。フォントが参照された場合、PDF ビューアー 例：Adobe Reader) は通常 (base 14 フォントが含まれる) 独自のリソースフォルダーからそのフォントの検索を行い、フォントがそこに無い場合はシステムのフォントフォルダーを検索します。フォントが利用できる場合、PDF が表示される際にそのフォントが使用されます。フォントが見つからない場合、リソースフォルダー上にある代替フォントが使用されるか、エラーが表示されます。代替フォントには異なるメトリックが指定されている場合もあり、表示エラーが生成される可能性もあります。

フォーマッターによりフォントがPDF ファイル内部に埋め込まれた場合、埋め込まれたフォントがPDF ビューアーにより使用されます。フォーマッターはフォントの全文字セットを埋め込むことができるほか、ドキュメント内部で使用されているフォントのサブセットだけを埋め込むことができます。フォントの埋め込み方法により、PDF ファイルの大きさが変わってくるだけではなく、フォントの使用に関する版権にも影響を与える可能性があります (以下のメモを参照ください)。フォーマッターのオプションを指定する際に、これらオプションの選択を指定できる場合があります。

XMLSpy とPDF フォント

XMLSpy では、XSL-FO ドキュメント (以下 FO ドキュメント) からPDF が生成されます。FOP といった外部 FO プロセッサーにより、XSL-FO ドキュメントが処理されます。オプションダイアログでは、FO プロセッサーの場所を指定することができます。これによりXMLSpy のGUI からFO の処理を開始することができます。

XSL-FO ドキュメントそのものはXML ドキュメントをXSLT スタイルシートで処理することにより生成されます。(XMLSpy に搭載されている) Altova のXSLT エンジンを使用することができるほか、外部 XSLT エンジンを使用することができます。

全テキストのフォントプロパティを含むPDF ドキュメントのフォーマットは、XSL-FO ドキュメントにて指定されます。使用しているフォーマッターがフォントのメトリックをフォントから直接読み取ることができる場合、フォントにアクセスするためにフォーマッターのセットアップを行う必要があります。FOP をフォーマッターとして使用している場合、base-14 フォントに含まれないフォントに対して、適切なフォントメトリックファイルを指定する必要があります。

フォーマッターがフォントを利用できるようにする

(FOP を含む) ほとんどのフォーマッターでは既にbase 14 フォントを利用することができ、これらのフォントをフォーマッターへ指定するために使用される名前を知ることが重要になります。base 14 フォントはフォーマッターによりサポートされる基本的なフォントですが、フォーマッター固有の方法に従うかたちで、これらのフォント以外にもフォーマッターが扱うことのできるフォントの数を増やすことができます。以下ではFOP にてフォントの数を増やす方法について記述されます。

FOP に対して新たなフォントサポートを追加するためのセットアップ手順

以下のステップにより、FOP が使用することのできるフォントを追加することができます：

1. PostScript または TrueType フォントファイルから、フォントのメトリックファイルを生成します。FOP に搭載されている PFM Reader ならびに TTF Reader ユーティリティにより、PostScript ならびに TrueType フォントを XML フォントメトリックファイルへ変換することができます。詳細については、FOP : Fonts ページを参照ください。
2. FOP 構成ファイルをセットアップすることで、目的のフォントメトリックファイルを使用することができるようになります。FOP 構成ファイルにてフォントファイルに関する情報を入力することになります。FOP: Fonts ページを参照ください。
3. `fop.bat` ファイルにて、以下にある最後の行を：

   ```
   "%JAVACMD%" […] org.apache.fop.cli.Main %FOP_CMD_LINE_ARGS%
   ```

   構成ファイルの場所を含むように変更します：

   ```
   "%JAVACMD%" […] org.apache.fop.cli.Main %FOP_CMD_LINE_ARGS% -c conf
   \fop.xconf
   ```

メトリックファイルが FOP（FOP 構成ファイル）にて登録され、FOP 実行ファイルが構成ファイルを読み取るようにセットされることで、指定されたフォントを PDF の作成に使用することができるようになります。

FOP 構成ファイルのセットアップ

FOP 構成ファイルは `fop.xconf` という名前で、FOP インストールフォルダー内にある `conf` フォルダー以下に配置されています。XML 形式により記述されるこのファイルを編集することで、FOP によりフォントメトリックファイルが適切に読み取られるように指定する必要があります。ドキュメント内部にある `font` 要素プレースホルダにて示される場所に、レンダリングを行う各要素に対応した `font` 要素を追加する必要があります：

```
<font metrics-url="arial.xml" kerning="yes" embed-url="arial.ttf">
          <font-triplet name="Arial" style="normal" weight="normal"/>
          <font-triplet name="ArialMT" style="normal" weight="normal"/>
</font>
```

上の例は、以下のように理解することができます：

`arial.xml`	メトリックファイルの URL です。絶対パスの使用が推奨されます。
`arial.ttf`	TTF ファイルの名前です（通常 `%WINDIR%\Fonts` 以下に収められています）。
`Arial`	フォントファミリーに `Arial` が指定されている場合、上記のメトリックと TTF ファイルが使用されるように指定します。
`style="normal"`	フォントスタイルが（`talic` などではなく）`normal` として定義されている場合、上記のメトリックと TTF ファイルが使用されるように指定します。
`weight="normal"`	フォントウェイトが（`old` などではなく）`normal` として定義されている場合、上記のメトリックと TTF ファイルが使用されるように指定します。

フォントの版権に関するメモ： フォントの使用は著作権法の対象となり、使用に関する条件は一定ではありません。フォントの埋め込みを行う場合、特にフォント全体の埋め込みを行う場合、購入したフォントをそのように使用しても問題が無いか確認する必要があります。

文字セット

フォントの文字セットも考慮する必要がある点に注意してください。base 14 フォントでは、ISO-8859-1 により規定される文字の他に、シンボルと Zapf Dingbats フォントにある文字がカバーされます。base 14 フォントによりカバーされない文字がドキュメントに含まれる場合、その文字が含まれる文字セットを含んでいるフォントを使用する必要があります。Arial Unicode のようなフォントには、Unicode によりカバーされる文字が含まれます。

3.9 チャート

XML ドキュメントがテキストビューまたはグリッドビューで開かれている場合、XML ドキュメント内のデータを、出力ウィンドウの1つであるチャートウィンドウに (円グラフや棒グラフのような) チャートという形で出力することができます。チャートはイメージファイルや XSLT、XQuery フラグメントという形でクリップボードへエクスポートすることができます。チャート機能を使うことで、XML ドキュメント内で選択した数値データを、素早く視覚的に表示することができます。

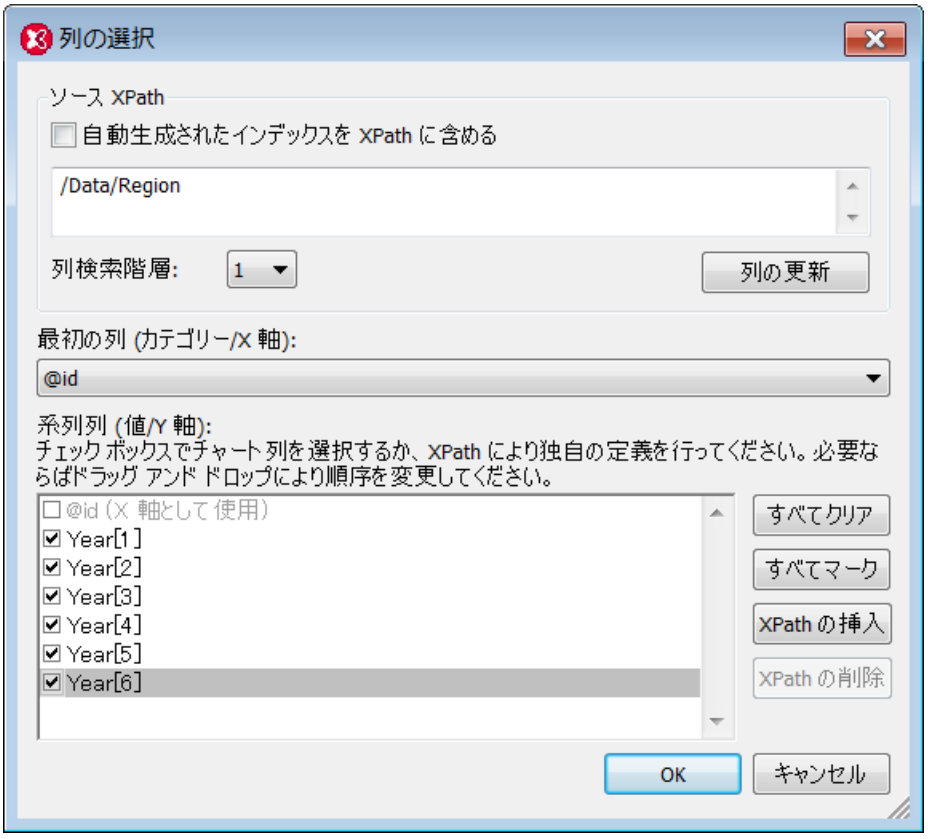
以下の種類のグラフが利用できます：

- 円グラフ (2D、3D)
- 棒グラフ (2D、3D)
- 棒グラス グループバー (2D、3D)
- スタックされた棒グラフ
- 折れ線グラフ
- 値付き折れ線グラフ
- エリアチャートならびにスタックされたエリアチャート
- ローソク足チャート
- ゲージチャート (円ならびに棒)
- オーバーレイチャート

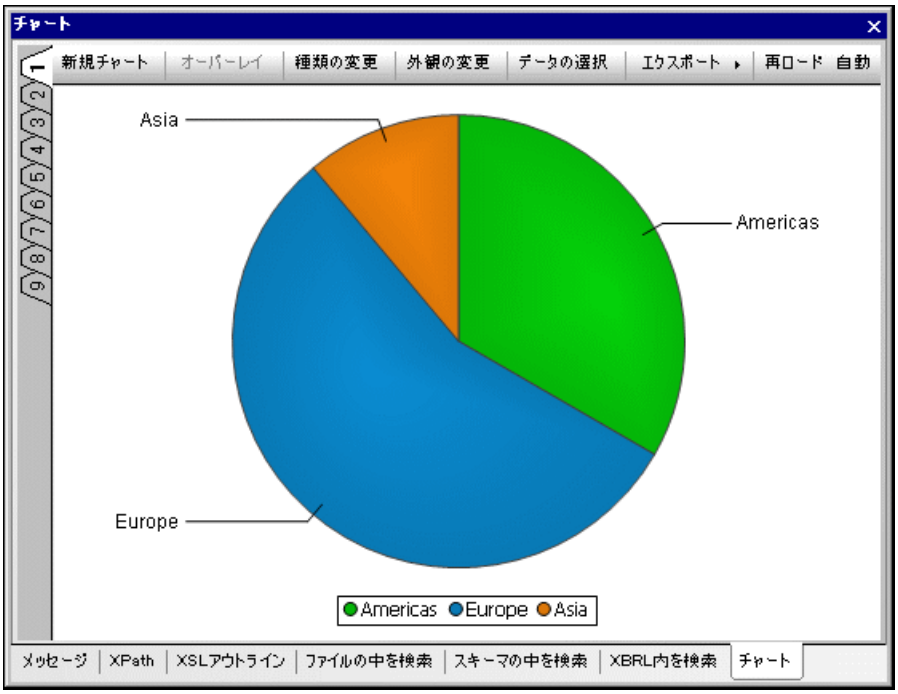
概要：作成からエクスポートまで

チャート作成を行うステップの概要を以下に記します。より詳細な情報については、このセクションのサブセクションを参照ください。

1. テキストビューまたはグリッドビューにて、データ選択のコンテキストノードとして使用したいノードを選択します。ノードの範囲を選択することも可能です。選択の方法についてはソース XPath のセクションを参照ください。
2. 右クリックして表示されるコンテキストメニューから、**新規チャート**」を選択します。その他にも、チャート出力ウインドウにて **新規チャート**」ボタンをクリックすることもできます。この操作により列の選択ダイアログが表示され、X-軸、Y-軸データを選択し、ソース XPath を修正することができます。



3. **「OK」**をクリックすることで、チャートウィンドウにチャートが作成されます（以下のスクリーンショットを参照）。



4. チャートのデータ選択などの設定は、チャートの作成後に編集することができます。ソースXPath 列選択の編集だけではなく、チャートの種類や外観などを指定することもできます。チャートの軸に対するデータの選択は **「データを選択」**ボタンをクリックすることで行うことができます。チャートの種類や外観は、それぞれ **「種類の変更」**ボタンと

外観の変更」ボタンにより修正することができます。

5. チャートはイメージファイル、またはXSLT、XQuery フラグメントとしてクリップボードへエクスポートすることができます。

その他の機能

以下の機能により、グラフの使用をより簡単に行うことができます：

- ***複数タブ***：現在のチャートを削除すること無く新規チャートを作成したい場合、（上のスクリーンショットにあるように）9つあるタブの内、使用されていないタブを選択して新規チャートの作成を行うことができます。XML ドキュメントが閉じられた場合でも、そのドキュメントから生成されたチャートは、チャートウィンドウのタブ内に存在し続けることに注意してください。
- ***自動再ロード***：（上のスクリーンショットで示されている）**自動**」ボタンを有効にすることで、XML ドキュメントのデータが変更されたときにチャートも自動的に再ロードされます。このボタンが有効になっていない場合、**再ロード**」ボタンをクリックして、手動でチャートの更新を行う必要があります。

サンプルファイル

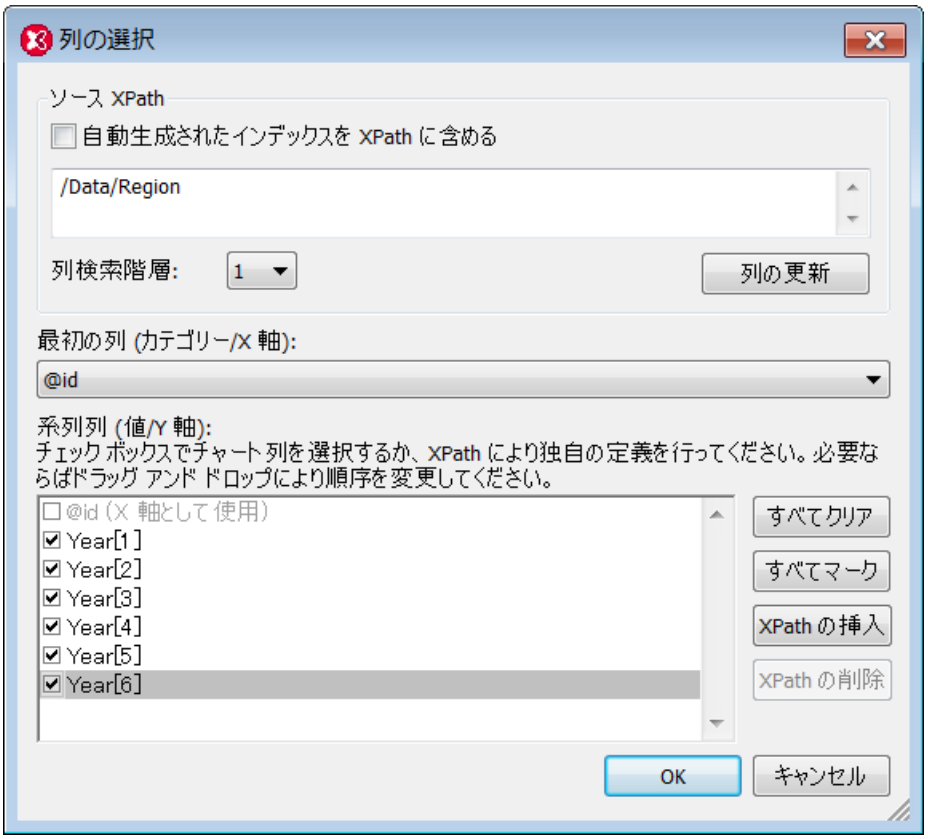
このセクションならびにサブセクションでは、チャートの動作説明を行うために`YearlySales.xml` XML ファイルが使用されます。このファイルは`C:\Documents and Settings\<username>\My Documents\Altova \XMLSpy2017\Examples\Tutorial` フォルダーからご利用になれます。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
        <Region id="Americas">
                <Year id="2005">30000</Year>
                <Year id="2006">90000</Year>
                <Year id="2007">120000</Year>
                <Year id="2008">180000</Year>
                <Year id="2009">140000</Year>
                <Year id="2010">100000</Year>
        </Region>
        <Region id="Europe">
                <Year id="2005">50000</Year>
                <Year id="2006">60000</Year>
                <Year id="2007">80000</Year>
                <Year id="2008">100000</Year>
                <Year id="2009">95000</Year>
                <Year id="2010">80000</Year>
        </Region>
        <Region id="Asia">
                <Year id="2005">10000</Year>
                <Year id="2006">25000</Year>
                <Year id="2007">70000</Year>
                <Year id="2008">110000</Year>
                <Year id="2009">125000</Year>
                <Year id="2010">150000</Year>
        </Region>
</Data>
```

3.9.1 チャートの作成

新規チャート」ボタンをクリックすると、（以下のスクリーンショットにある）列の選択ダイアログが表示され、チャート作成のための3つのデータ選択パラメーターを指定します。これらパラメーターはチャートデータテーブルの作成のために使用されます：

- *ソースXPath* : XPath 条件式は、ダイアログが開かれる際に自動的に入力されます。列の選択ダイアログが開かれたときに選択されていたXML ドキュメント内のノードがここでは入力され、キーボードを使って編集することができます。*インデックスをXPath に自動追加チェックボックス*により、XPath 内のかっこがフィルタリングに使用されるかが決定されます (詳細は*ソースXPath* を参照ください)。ソースXPath にて選択されたノードの子孫は、X-軸、Y-軸データ列として使用することができるようになります。*列検索階層コンボボックス*では、X-軸、Y-軸のデータ選択に使用するためのノードを検索する階層を指定することができます。ソースXPath を編集するか、新たな列検索階層を選択した後は、**「列の更新」**ボタンをクリックして、ダイアログにあるX-軸、Y-軸リストの更新を行う必要があります。
- *X-軸* : このコンボボックスでの選択によりX-軸として使用されるノードを指定することができます。この選択に取得されるシーケンスによりX-軸に表示されるラベルが決定されます。*コンボボックスの自動列挙*オプションによりX-軸には番号が与えられます。X-軸コンボボックスではY-軸のために作成されたXPath 条件式も利用できることに注目してください。
- *Y-軸* : このペインでチェックされたエントリが、Y-軸で表示される数値のノードになります。**「全てマーク」**と**「全てクリア」**ボタンにより、Y-軸ペインにある全てのアイテムの選択とその解除を行うことができます。**「XPath の挿入」**ボタンにより、ソースXPath から得られたノードの子孫では無い系列を生成することができます。X-軸で選択されたノードやXPath 条件式は、Y-軸に対して使用することができず、グレーで表示されます。



チャートデータテーブルの作成方法

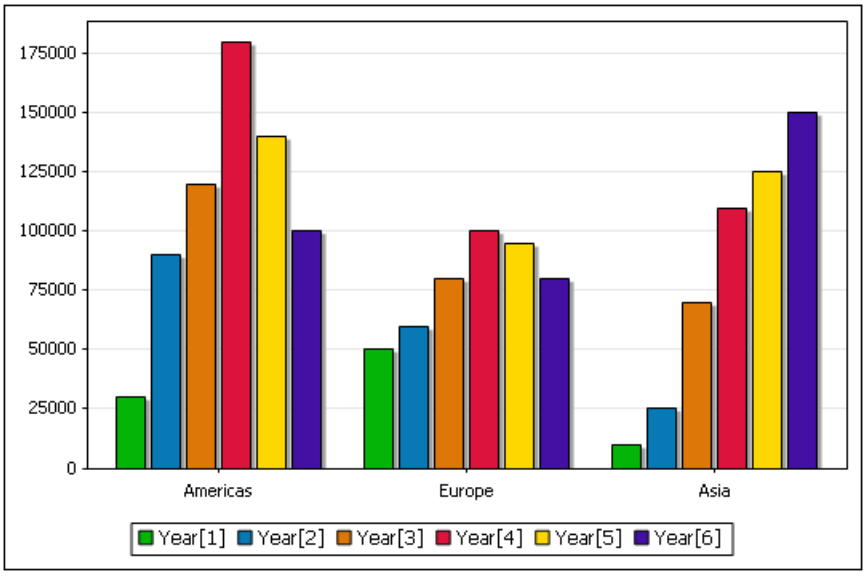
チャートにて使用されるデータは列のセルなくダイアログにおける選択により決定されます。ここでは例を使用しながらチャートのデータを選択する方法を説明します。この例で使用されるXML ドキュメントには3つの`Region` 要素が含まれるため、ソースXPath の`/Data/Region` により、それら3つの全てが返されます。各 `Region` 要素をコンテキストノードとして、データの列が生成されます。ソースXPath により選択された各 `Region` 要素に対して:

1. X-軸表現により最初の列が生成されます (デフォルトでは、この列がX-軸のラベルとして使用される列になります)。
2. 各系列 (Y-軸選択) に対して、列が生成されます。

上のスクリーンショットにある列の選択ダイアログから生成されたチャートデータは、以下のテーブルのようになります。

ソース XPath	X-軸	Y-軸 系列列)					
Region[1]	@id	Year[1]	Year[2]	Year[3]	Year[4]	Year[5]	Year[6]
Region[2]	@id	Year[1]	Year[2]	Year[3]	Year[4]	Year[5]	Year[6]
Region[3]	@id	Year[1]	Year[2]	Year[3]	Year[4]	Year[5]	Year[6]

このデータから生成された棒グラフは以下のようになります：



以下の重要なポイントにご注意してください：

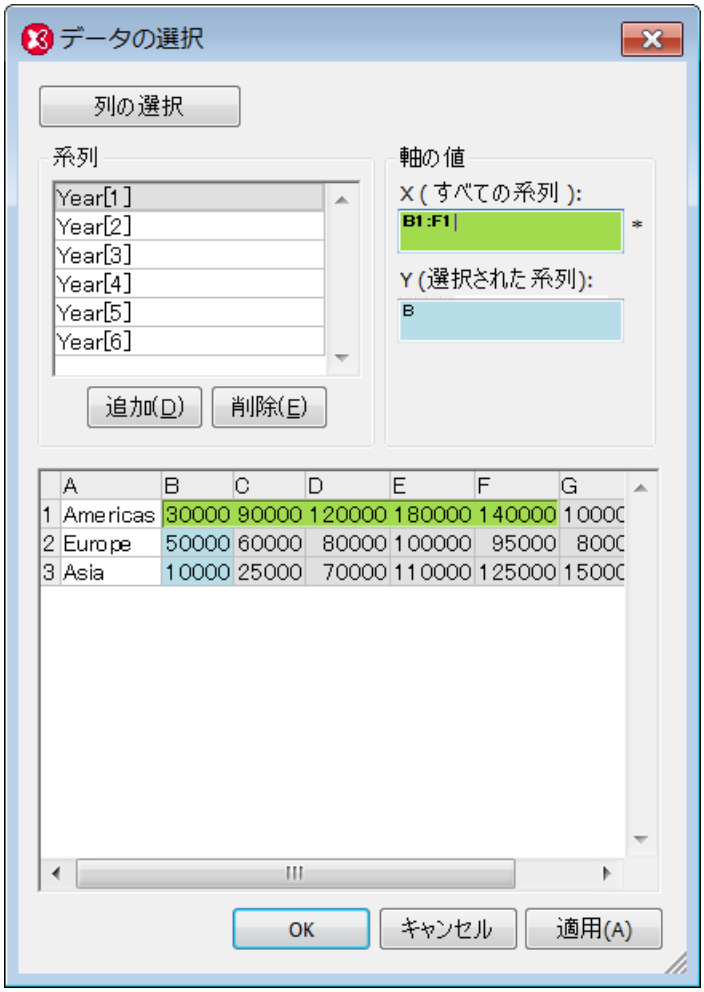
- X-軸の要素の数は、ソースXPath 条件式により得られたシーケンスのサイズにより決定されます（この例の場合では3つになります）。
- ソースXPath から得られたノードはコンテキストノードとなり、X-軸の各要素で使用される2種類のデータ生成に使用されます：(i) X-軸の要素ラベル（X-軸選択により作成）と、(ii) 要素に対する系列のプロット（これらの系列はY-軸選択とともに選択されます）。X-軸ならびにY-軸に対して入力されたXPath 条件式は、これら（ソースXPath）ノードのコンテキスト内で評価されます。
- X-軸選択により得られたシーケンスが、それぞれ各要素のラベルになります。実際にある要素よりもラベルの数が少ない場合、いくつかの要素はラベルなしで表示されます。
- 各系列（例えば `Year[1]`）は各コンテキストノードに対して1回評価されます。円グラフのようなチャートによっては、使用される系列を1つにすることができます。
- 凡例は系列アイテムの名前から取得されます。

ここで使用したサンプルのXML ドキュメントを示します。このファイルは `YearlySales.xml` というファイル名で `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\Tutorial` に収められています。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
        <Region id="Americas">
                <Year id="2005">30000</Year>
```

```
                <Year id="2006">90000</Year>
                <Year id="2007">120000</Year>
                <Year id="2008">180000</Year>
                <Year id="2009">140000</Year>
                <Year id="2010">100000</Year>
        </Region>
        <Region id="Europe">
                <Year id="2005">50000</Year>
                <Year id="2006">60000</Year>
                <Year id="2007">80000</Year>
                <Year id="2008">100000</Year>
                <Year id="2009">95000</Year>
                <Year id="2010">80000</Year>
        </Region>
        <Region id="Asia">
                <Year id="2005">10000</Year>
                <Year id="2006">25000</Year>
                <Year id="2007">70000</Year>
                <Year id="2008">110000</Year>
                <Year id="2009">125000</Year>
                <Year id="2010">150000</Year>
        </Region>
</Data>
```

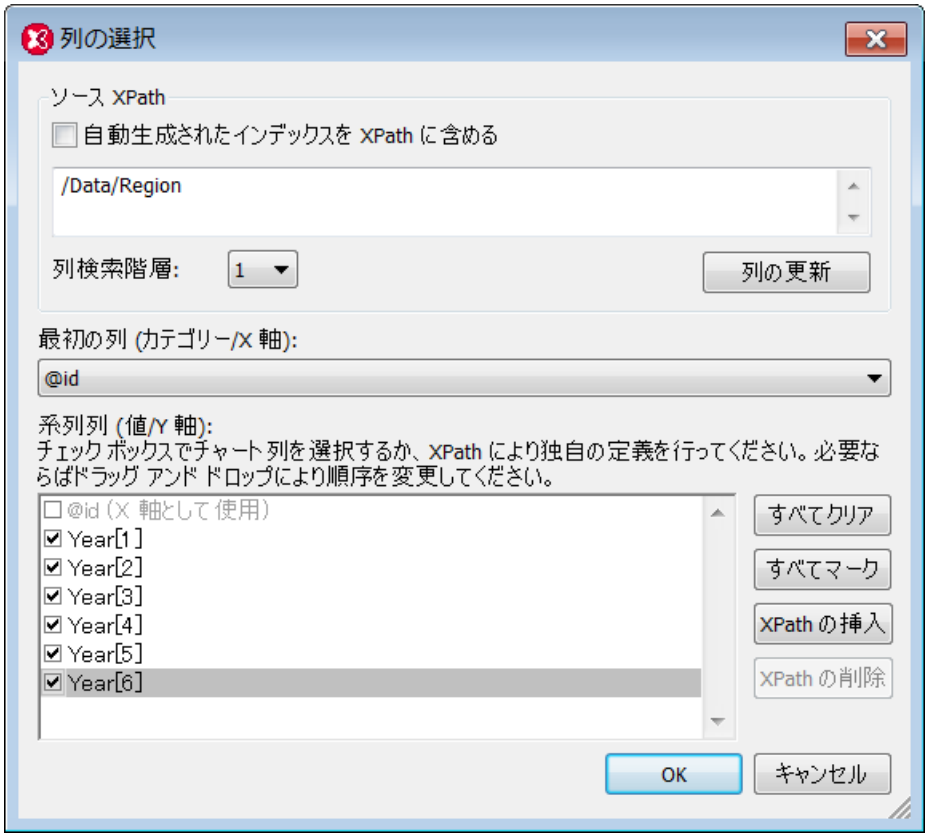
データの選択ダイアログに表示される上のデータの選択を (以下のスクリーンショットにある) テーブルの中で見ることができます。データの選択ダイアログへはチャート出力ウィンドウの **[データの選択]** ボタンをクリックすることでアクセスすることができます。



列の選択ダイアログにある個別のパラメーターの詳細については、各セクションを参照ください:ソースXPath、X-軸選択、Y-軸選択、チャートデータ。

3.9.2 ソース XPath

ソースXPath は列の選択ダイアログにて指定することができます。ソースXPath により、ドキュメント内のどのノードが、X-軸、Y-軸のデータとして使用できるかが決定されます。列検索階層コンボボックスでは、X-軸、Y-軸のデータとして使用されるノードの検索の深さが指定されます。



ソースXPath または列検索階層を選択した後、**列の更新**」ボタンにより、ダイアログ内でX-軸とY-軸に使用可能な選択肢を更新します。かっこがXPath 条件式に含まれている場合 (例：`/data/Region[1]` では `[1]` というかっこ表現により最初の`Region` 要素が選択されます)、自動生成されたインデックスを含めるチェックボックスを有効にする必要があります。ソースXPath 条件式を修正した場合、**列の更新**」ボタンをクリックするようにしてください。

カーソル位置からのソースXPath

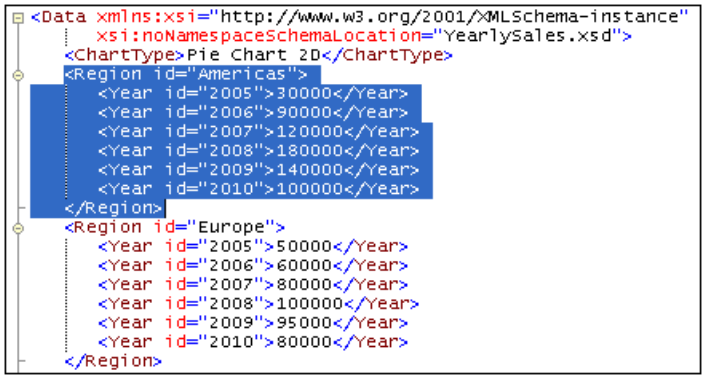
列の選択ダイアログが開かれたときに選択されるソースXPath について説明するために、以下に示されるドキュメントを使用します。このファイルは`YearlySales.xml` というファイル名で`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\Tutorial` に収められています。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
        <Region id="Americas">
                <Year id="2005">30000</Year>
                <Year id="2006">90000</Year>
                <Year id="2007">120000</Year>
                <Year id="2008">180000</Year>
                <Year id="2009">140000</Year>
                <Year id="2010">100000</Year>
        </Region>
        <Region id="Europe">
                <Year id="2005">50000</Year>
                <Year id="2006">60000</Year>
                <Year id="2007">80000</Year>
                <Year id="2008">100000</Year>
```

```
            <Year id="2009">95000</Year>
            <Year id="2010">80000</Year>
        </Region>
        <Region id="Asia">
            <Year id="2005">10000</Year>
            <Year id="2006">25000</Year>
            <Year id="2007">70000</Year>
            <Year id="2008">110000</Year>
            <Year id="2009">125000</Year>
            <Year id="2010">150000</Year>
        </Region>
</Data>
```

以下のケースが考えられます：

- カーソルが`Data`要素の開始タグ（属性やその値を含む）や終了タグ内にある場合、または`Data`要素内部（但し、`Data`の子要素内部ではない）にカーソルがある場合：ソースXPathは`/Data`となります。
- カーソルが`Region`要素の開始タグ（属性やその値を含む）や終了タグ内にある場合、または`Region`要素内部（但し、`Region`の子要素内部ではない）にカーソルがある場合：ソースXPathは`/Data/Region`となります。
- カーソルが`Year`要素の開始タグ（属性やその値を含む）や終了タグ内にある場合、または`Year`要素内部にカーソルがある場合：ソースXPathは`/Data/Region[N]/Year`となります。かっこによるフィルター`[N]`により、選択された`Year`要素を含んでいる`Region`要素が指定され、X-軸、Y-軸のデータ選択は、この`Region`要素内にある`Year`要素に限定されます。
- 個別の`Region`要素（例：`/Data/Region[1]`）を選択したい場合、以下のスクリーンショットにあるように、最初の`Region`要素をハイライトします。



- 同様に、2つの`Region`要素をハイライトすることで、これら2つの`Region`要素だけを選択するXPath条件式が生成されます。

ソースXPath条件式は、後からでも列の選択ダイアログにて編集することができます。

グリッドビューでは、ノードをクリックした範囲をマークすることで、選択を行うことができます。グリッドビューの選択から生成されるソースXPathは、上のテキストビューのものと同様のものになります。

XPath のインデックスを追加

インデックスを追加するチェックボックスにより、XPath条件式内にあるかっこを使ったフィルタリングが行われるか、列の選択ダイアログ呼び出された際に、これらかっこフィルタリングを自動的に入力するか、または手動で入力するかが決定されます。例えば、カーソルが`Region`要素の最初の子孫要素内にあり、インデックスを追加するチェックボックスが有効になっている場合、自動的に入力されるXPath条件式は、例えば`/Data/Region[1]/Year`となります。インデックスを追加するチェックボックスが有効になっていない場合、条件式は`/Data/Region/Year`のようになります。

インデックスを追加するチェックボックスでは、手動で入力されたかっこを保持するかどうかも決定されます。ソースXPath条

件式のかっこを使用したい場合、インデックスを追加するチェックボックスを有効にする必要があります。

ソースXPath 選択の意味

ソースXPath 選択に関して以下の点に注意してください。

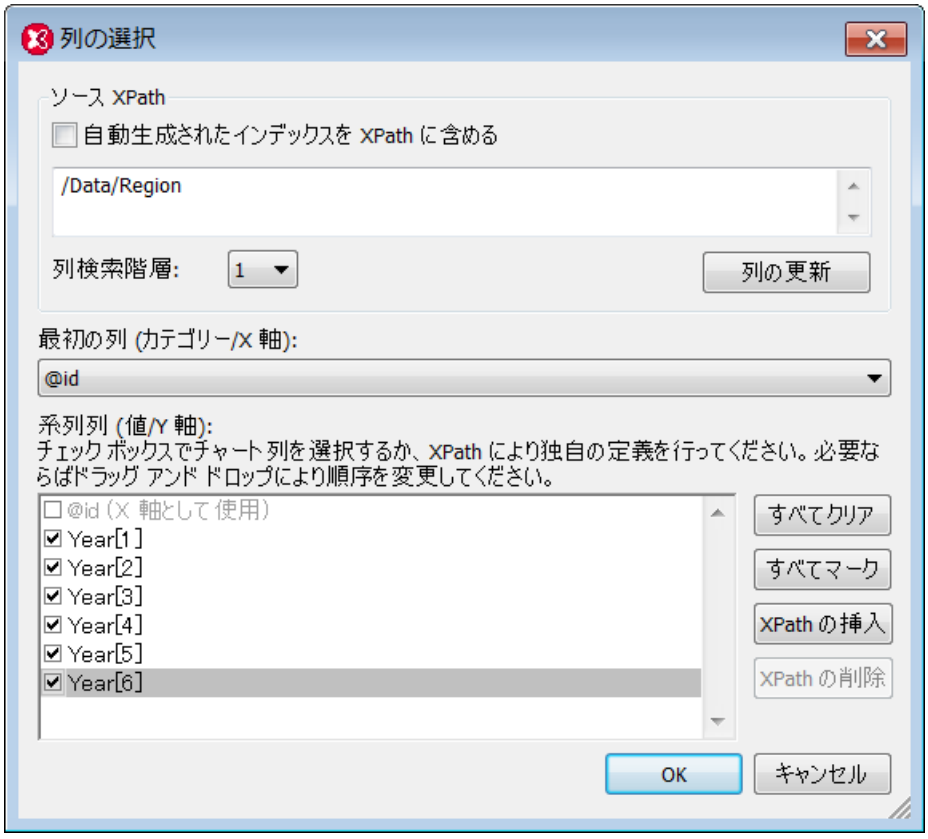
- ソースXPath により得られたシーケンスのアイテム数により、X-軸の要素の数が決定されます。ソースXPath を修正する以外に、X-軸の要素数を変更する方法は1つしかありません。要素の数よりも大きな系列の番号を選択するというものです。この方法の詳細に関してはX-軸選択を参照ください。
- ソースXPath ノードは、X-軸、Y-軸のデータ選択で利用可能な全てのノード、ならびにXPath 条件式の祖先ノードとなります。
- 上の2つの結果、ソースXPath 条件式に対する変更は、X-軸の要素数だけではなく、チャートに関連したXPath 条件式のコンテキストにも影響を与えてしまうことにご注意してください。

以下に、上の例で示されたXML ドキュメントに対して適用できるXPath 条件式とその意味を記します：

- **/Data/Region** ：3つのRegion 要素を返し、X-軸には3つの要素が与えられます。各 Region 要素がXPath 条件式のコンテキストノードになります。
- **/Data/Region/Year** ：18のYear 要素を返し、X-軸の要素数は18になります。各 Year 要素がXPath 条件式のコンテキストノードになります。
- **/Data/Region[1]/Year** ： 最初のRegion 要素の子要素である6つのYear 要素が返され、X-軸の要素数は6になります。最初のRegion 要素の子要素である各 Year 要素がXPath 条件式のコンテキストノードとなります。
- **distinct-values(//Year/@id)** ：6つのアイテムが返されます（Year/@id 属性のdistinct 値：2005、2006、2007、2008、2009、2010）。しかしこのXPath 条件式はノードアイテムを返さないため、XPath 条件式のコンテキストノードとなることはできません。例えばcurrent() 関数を使用するなどして、このシーケンス内のアイテムがXML ドキュメント内のターゲットノードに使用されるのであれば、このXPath 条件式をドキュメントのルートからスタートするように書きなおして、コンテキストノードを確立する必要があります（例：/Data/Region[1]/Year[@id eq .]）。

3.9.3 X-軸の選択

X-軸の選択は、列の選択ダイアログにて指定されます（以下のスクリーンショットを参照）。この選択によりX-軸に表示されるラベルが決定されます。ラベルはデータの選択ダイアログにて編集することができます（以下を参照）。



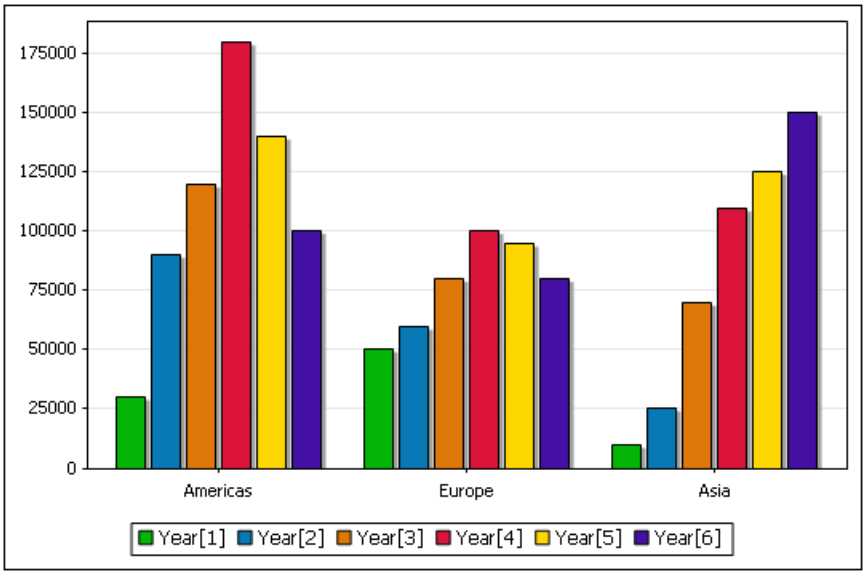
以下のXML ドキュメントを例にして考えてみましょう (このファイルは`YearlySales.xml` というファイル名で`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\Tutorial` に収められています)。カーソルが最初の`Region` 要素の開始タグ内に位置している状態で、チャート出力ウィンドウの **新規チャート** ボタンをクリックします。列の選択ダイアログがソースXPath と共に表示されます：`/Data/Region`。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
        <Region id="Americas">
                <Year id="2005">30000</Year>
                <Year id="2006">90000</Year>
                <Year id="2007">120000</Year>
                <Year id="2008">180000</Year>
                <Year id="2009">140000</Year>
                <Year id="2010">100000</Year>
        </Region>
        <Region id="Europe">
                <Year id="2005">50000</Year>
                <Year id="2006">60000</Year>
                <Year id="2007">80000</Year>
                <Year id="2008">100000</Year>
                <Year id="2009">95000</Year>
                <Year id="2010">80000</Year>
        </Region>
        <Region id="Asia">
                <Year id="2005">10000</Year>
                <Year id="2006">25000</Year>
                <Year id="2007">70000</Year>
```

```
            <Year id="2008">110000</Year>
            <Year id="2009">125000</Year>
            <Year id="2010">150000</Year>
        </Region>
</Data>
```

ソースXPath のセクションで説明したように、このソースXPath により、X-軸に3つの要素があるチャートが作成されます（ソースXPath は Region 要素の数である3つのアイテムを返すため）。これら3つの要素のラベルを3つの地域の名前で表したいので、X-軸選択のコンボボックスから@id 属性を選択します（上の列の選択ダイアログのスクリーンショットを参照）。

各要素に対してチャートデータを生成するには、各 Region 要素の評価を行わなければなりません。各 Region 要素に対して、id 属性によりX-軸属性のラベルが適切に生成されます。X-軸は以下のスクリーンショットのように表示されます。

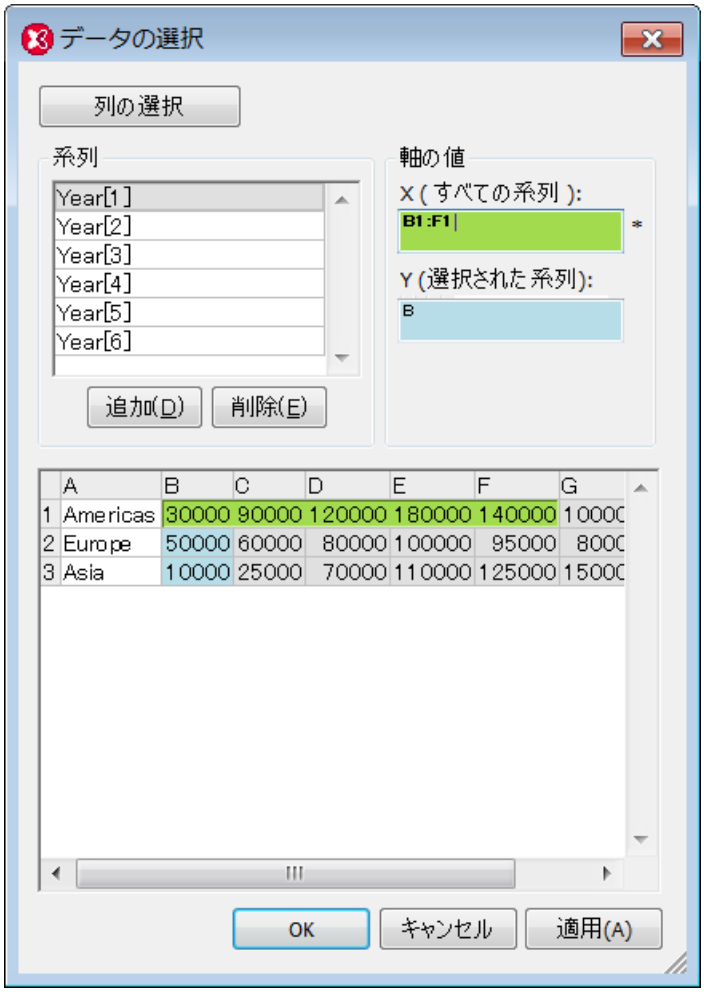


他のXPath 条件式がX-軸コンボボックスで選択された場合、その条件式が Region 要素コンテキスト内で評価され、評価の結果が対応する要素のラベルとなります。*自動列挙* オプションでは要素の番号に対応した番号のシーケンスを生成します（最初の要素が1となり、2番目の要素が2となります。

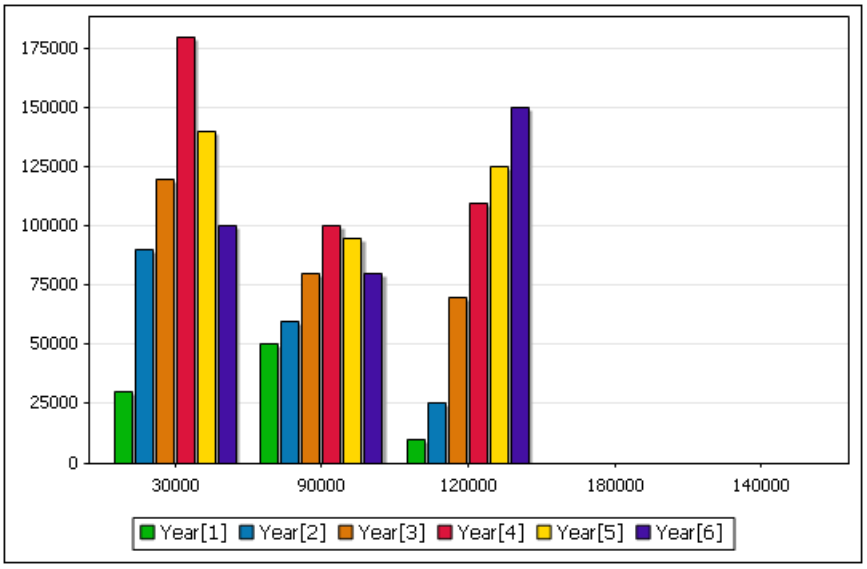
X-軸のラベルとX-軸要素数の修正

X-軸に対するラベルの選択はデータの選択ダイアログから行うことができます（チャート出力ウィンドウの **「データの選択」** ボタンをクリックすることでアクセスできます）。

以下のスクリーンショットで示されるデータの選択ウィンドウにおいて、軸の値ペイン内のX-軸テキストボックスをクリックします。これでX-軸の選択を修正することができます。そこで B1 フィールドをクリックして、F1 までマウスをドラッグし、選択を B1:F1 とします。 **「OK」** をクリックして新たなチャートを確認します。



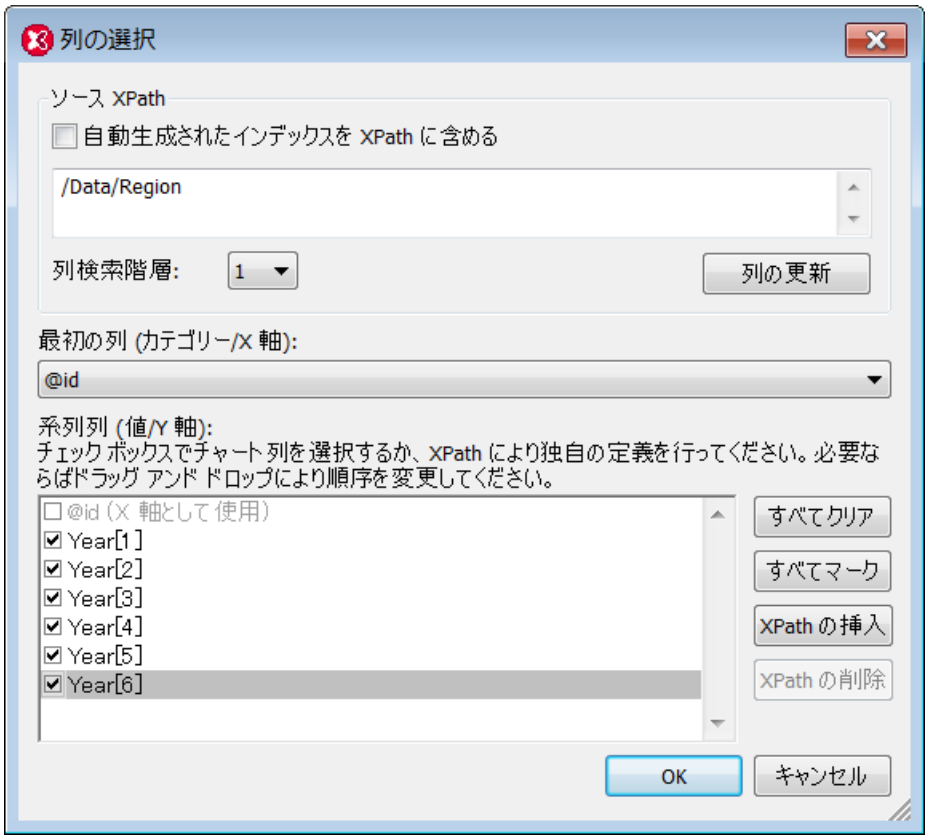
この選択により、以下のスクリーンショットにあるような要素に対するラベルが作成されます。新しい選択では5つのアイテムが含まれるため、5つの要素が生成されたことにも注目してください。実際に表示されているのは最初の3つだけですが、これはソースXPath が返したノード数は3つで、これら3つのノードがチャートに表示されるためです。これら3つのノードは、データの選択ダイアログにて表示されているテーブルの行に対応します。テーブル内の行はソースXPath を変更することでしか修正できないということに注意してください。



ソースXPath ならびにX-軸の選択に関するより詳細については、チャートデータの作成方法を参照ください。

3.9.4 Y-軸の選択

以下のスクリーンショットにあるY-軸は系列軸とも呼ばれます。

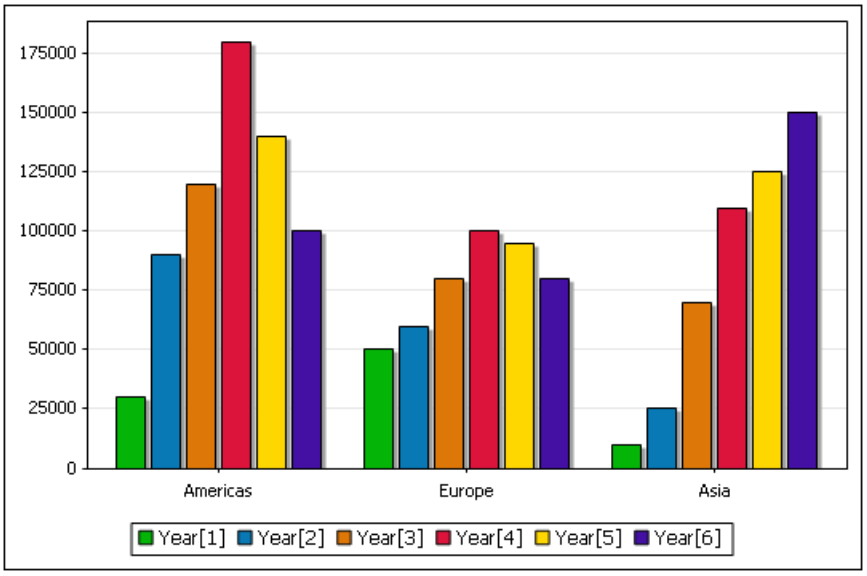


この軸で行われる選択により、各 X-軸要素に対してどれだけの系列がプロットされるのか決定されます。1つの系列だけが

選択された場合、各 X-軸が要素となり、その系列の XPath 条件式により得られた値がプロットされます。2つ以上の系列が選択された場合、以下のスクリーンショットにあるように複数の系列が表示されます。チャートデータは以下のテーブルに示されるようになります。Y-軸のデータ選択に対するコンテキストノードは各 `Region` 要素となります。

ソース XPath	X-軸	Y-軸 (系列列)					
`Region[1]`	`@id`	`Year[1]`	`Year[2]`	`Year[3]`	`Year[4]`	`Year[5]`	`Year[6]`
`Region[2]`	`@id`	`Year[1]`	`Year[2]`	`Year[3]`	`Year[4]`	`Year[5]`	`Year[6]`
`Region[3]`	`@id`	`Year[1]`	`Year[2]`	`Year[3]`	`Year[4]`	`Year[5]`	`Year[6]`

結果として表示されるチャートは以下のようなものになります：



3つの X-軸要素があり、各 `Region/@id` 属性の値によりラベルが表示されます。X-軸の各要素にて、各系列の XPath 条件式が評価されます。この例の場合、X-軸の各要素に対して、6つある各系列が評価、プロットされています。例えば、最初の系列（`Year[1]`）は、3つある地域の全てに対してプロットされており、同じことが `Year[6]` まで当てはまります。

メモ 円グラフや、各要素に対して1つの系列しかない様なグラフでは、1つの軸しか使用されません。例えば、各要素に対して1つの系列しか存在しないグラフでは、X-軸の各要素には1つの棒しか表示されず、各棒が1つある系列を表しています。円グラフの場合、単一の系列の値の合計が100%になるように、X-軸の各要素が割り当てられます。

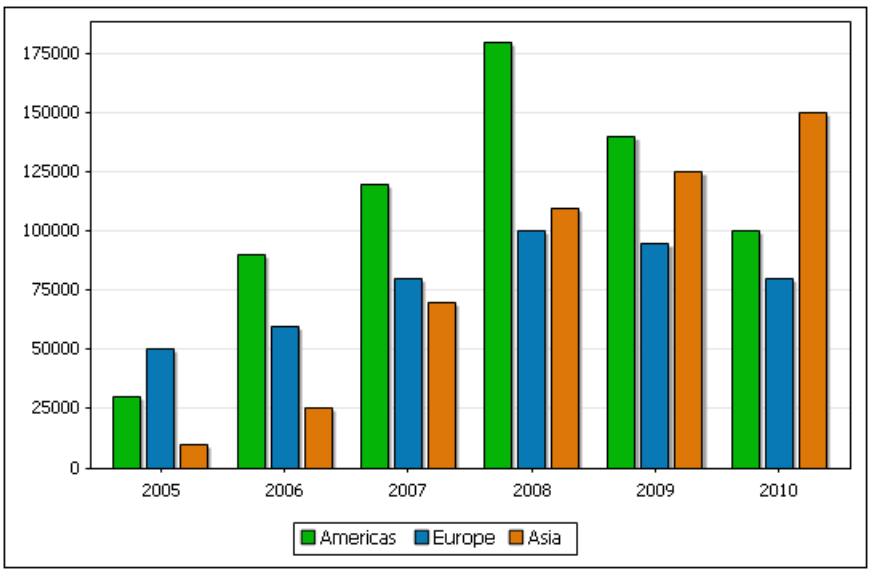
Y-軸の凡例

チャート下部に表示される凡例は、各系列の名前になります。これら名前はデータの選択ダイアログにて修正することができます。

X-軸とY-軸の選択を切り替える

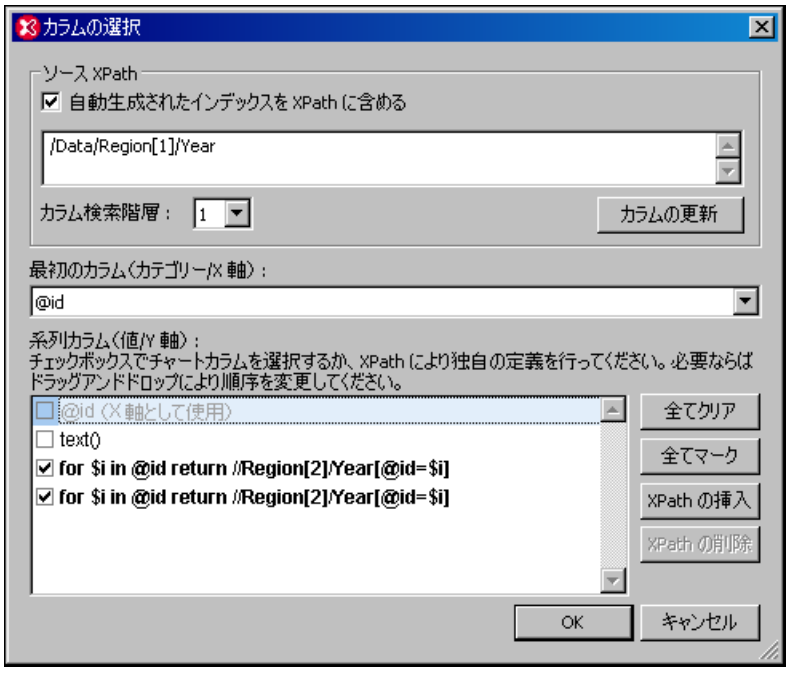
上の例では地域が X-軸となり、各地域の年間売上が Y-軸にプロットされていました（つまり `Year` 要素が系列になっていました）。しかし年を X-軸にプロットし、棒グラフを使って毎年の年間売上を地域ごとに比較したい場合はどうすればよいのでしょうか？（ソース XPath の選択から得られる）6つの X-軸系列を入手し、X-軸の系列を対応する年によりラベル付け

し、各 X-軸系列で描画される3つの (地域) 系列を選択する必要があります。以下のスクリーンショットにある列の選択ダイアログでは、このようなデータの選択を行う方法が示されています。



系列の選択

ノードを系列として選択するためには、ノードのチェックボックスにチェックを入れます。ソースXPath と列検索階層を修正して、目的のノードが系列ペインに表示されるようにします。他にも、以下のスクリーンショットにあるように XPath 条件式を追加してノードを選択するという方法もあります。このような使用に関する詳細については[チャートの例 詳細](#)を参照ください。



メモ

このセクションで使用されたXML ドキュメントのサンプルは以下のようなものになります。このファイルは

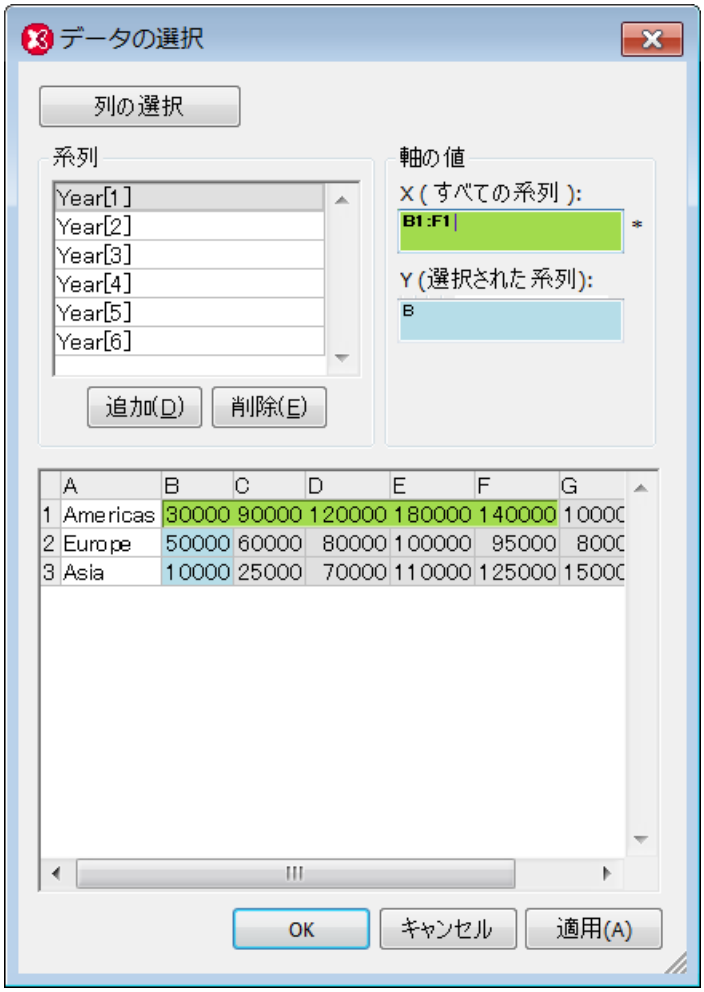
YearlySales.xml というファイル名で C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\Tutorial に収められています。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
       <Region id="Americas">
              <Year id="2005">30000</Year>
              <Year id="2006">90000</Year>
              <Year id="2007">120000</Year>
              <Year id="2008">180000</Year>
              <Year id="2009">140000</Year>
              <Year id="2010">100000</Year>
       </Region>
       <Region id="Europe">
              <Year id="2005">50000</Year>
              <Year id="2006">60000</Year>
              <Year id="2007">80000</Year>
              <Year id="2008">100000</Year>
              <Year id="2009">95000</Year>
              <Year id="2010">80000</Year>
       </Region>
       <Region id="Asia">
              <Year id="2005">10000</Year>
              <Year id="2006">25000</Year>
              <Year id="2007">70000</Year>
              <Year id="2008">110000</Year>
              <Year id="2009">125000</Year>
              <Year id="2010">150000</Year>
       </Region>
</Data>
```

3.9.5 チャートのデータ

「データの選択」ボタンをクリックすることで、(以下のスクリーンショットにある) データの選択ダイアログが表示されます。ダイアログは、(i) 系列ペイン、(ii) 軸の値ペイン、そして (iii) チャートデータテーブルの3つのペインから構成されます。

「列の選択」ボタンをクリックすることで、列の選択ダイアログが表示され、ソースXPath や X-軸、Y-軸のデータ選択を変更することができます。



系列ペイン

系列ペイン内で最初から選択されている系列は、列の選択ダイアログにて選択したものになります。**「OK」**ボタンをクリックしたときに、このペインで選択されているものがチャート内に表示される系列になります。系列ペインでは3つの操作を行うことができます：

- **系列の追加と削除**：チャート内に表示される系列の数を変更することができます。
- **系列の名前変更**：系列の名前はチャート内の凡例として表示されます。
- **各系列のデータを選択**：系列ペインにて系列が選択された状態だと、X-軸とY-軸のデータを、軸の値ペインで指定することができます。その方法については以下を参照ください。

軸の値ペイン

X-軸ならびにY-軸のデータは、軸の値ペインの対応するテキストボックスから入力することができます。どちらかのテキストボックスをクリックして、その中の値を編集することができ、編集状態であることがテキストボックスの右側に、アスタリスクとして表示されます。データの選択は、チャートデータテーブルからデータの範囲を選択することでも行うことができます。選択するデータの範囲は、列や行全体、またはそれらの一部として行うことができます。データ範囲の指定は、キーボードによる入力（例えば、`A` や `3`、`B1:F1` など）もしくはチャートデータテーブルの範囲をマークすることで行うことができます。データの範囲を選択するには、範囲の最初のセルを選択し、カーソルを最後のセルまでドラッグします。列や行全体をマークするには、列または行のヘッダーをそれぞれマークしてください。

X-軸の選択により、X-軸ノードのラベルが決定され、全ての系列に対して適用されます。X-軸における要素の数は変更さ

れません。

Y-軸の選択により、選択された系列に対して使用されるセルの範囲が決定されます。選択されたセルの数がX-軸の要素より少ない場合、対応する要素を持たないX-軸の系列は表示されません。選択されたセルの数がX-軸の要素より多い場合、余った分だけ新たに要素が対応する値とともに作成されます。

チャートデータテーブル

(データの選択ダイアログ下部に表示される) チャートデータテーブルの構造は、列の選択ダイアログの選択から取得されます。

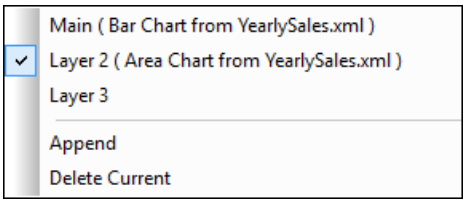
- テーブル内にある行の数は、ソースXPath から得られたシーケンス内にあるアイテム数に対応します。
- 列はアルファベットのA から順に名前がふられています。この列名を使って、軸の値ペインから選択を行うことができます (例：`B1:F1`)。
- 最初の列は、ソースXPath 条件式から得られたノードをコンテキストに、列の選択ダイアログにて選択されたX-軸を評価することで取得しています。
- 最初の列を除く、その他全ての列は、列の選択ダイアログにて選択されたY-軸を評価することで得られています。列の選択ダイアログのY-軸の選択から得られた各系列は、チャートデータテーブルの列に対応します。

チャートデータテーブルは、列の選択ダイアログにて選択されたデータのスーパーセット (上位集合) として見ることもできます。このスーパーセットから、(軸の値ペインにより) 目的のデータの範囲、または個別の系列を選択することができます。

3.9.6 オーバーレイ

オーバーレイとは、ベースチャートの上に置かれるチャートのことです。オーバーレイを以下の方法で追加することができます：

1. **オーバーレイ**ボタンをクリックして、オーバーレイメニューを開きます (*下のスクリーンショット参照*)。



2. **追加** をクリックします。新規のレイヤーが追加されます。
3. 新規のレイヤーが選択されると、**新規チャート**ボタンをクリックして、チャート作成内の選択で指定されているとおりデータを選択します。
4. チャートの種類と外観を変更するには、**種類の変更** と**外観の変更** をそれぞれクリックします。

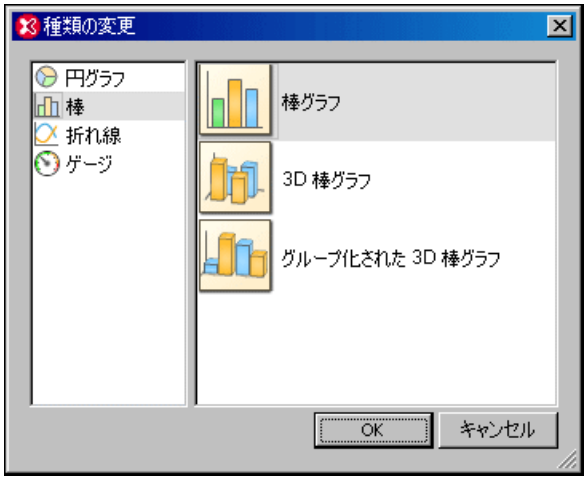
追加できるオーバーレイの数量に制限はありません。各新規のレイヤーはダイアグラム内で既存のレイヤーに追加され重ねて表示されます。レイヤーの順序を変更する場合は、順序の変更を使用して変更します。現在選択されているレイヤーを、**現在選択されている項目**をクリックして削除することができます。

メモ： 各オーバーレイは、下のレイヤーの上に追加されます。エリアチャートのみが透明に設定することができ、他のレイヤーの設定は最適では内可能性があります。たとえば、棒グラフが折れ線グラフの上に置かれている場合、折れ線グラフの外観を損なう可能性があります。上記の点を考慮して、レイヤーの順序を計画してください。

3.9.7 チャートの設定：クイックリファレンス

チャートの種類

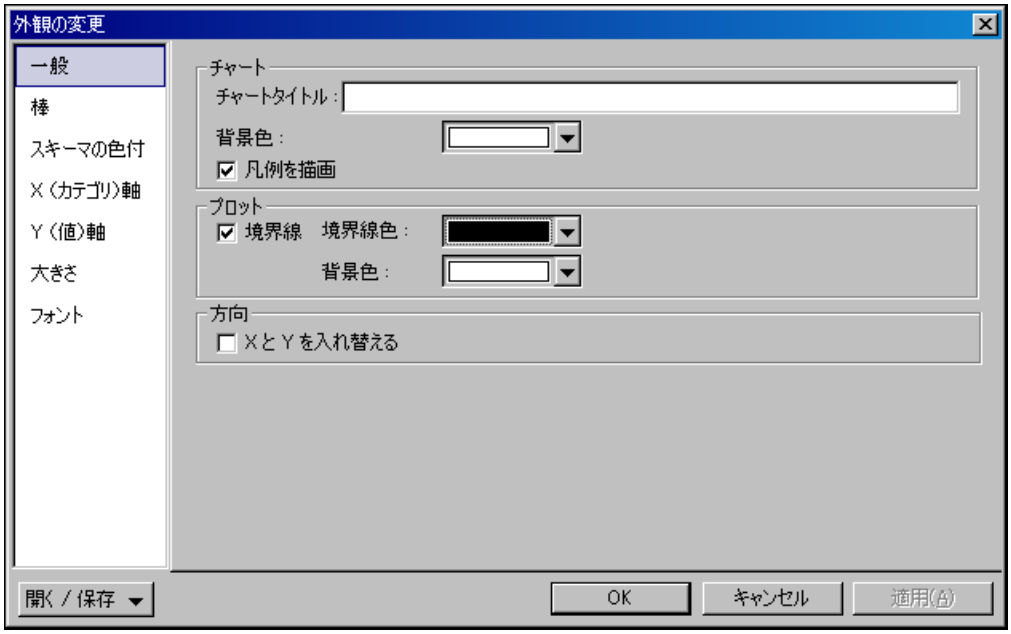
(以下のスクリーンショットにある) 種類の変更ダイアログにて、使用したい種類のグラフを選択して、「**OK**」により確定します。種類の変更ダイアログへは、「**種類の変更**」ボタンからアクセスすることができます。



チャートの種類が変更された後は、チャートの設定（タイトルや高さ、幅など）を、以下のスクリーンショットで示される外観の変更ダイアログから行う必要があり、チャートデータも指定する必要があります（以下のスクリーンショットを参照）。各グラフの種類に対してデータの選択を行う方法については、チャートの作成ならびにチャートデータのセクションを参照ください。

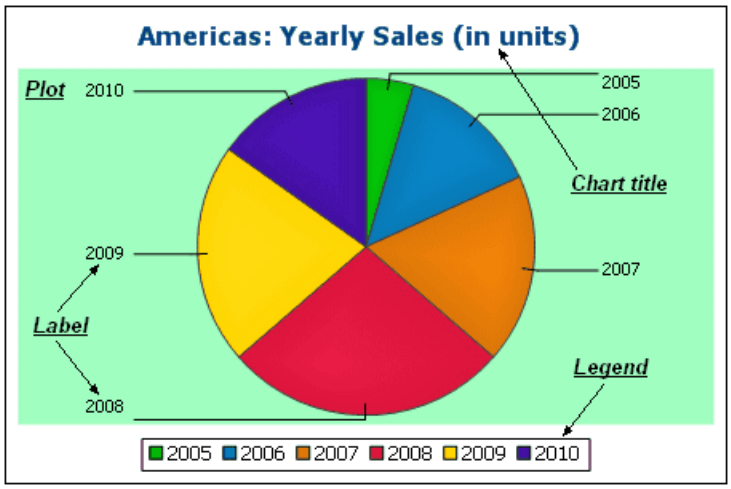
チャートの外観

以下のスクリーンショットは棒グラフの設定ダイアログのものです。チャートのタイトルやスキーマの色付、フォントサイズといったチャートの設定は、外観の変更ダイアログにて行ないます。このダイアログへは **「外観の変更」**ボタンからアクセスすることができ、そこで設定することのできる内容もチャートの種類によって変わってきます。



以下の設定は全ての種類のチャートに対して適用されます：

- **一般**：チャートのタイトルや、タブ、チャートの背景色、プロットの境界線やその背景色などを編集することができます（以下のスクリーンショットを参照）。以下のスクリーンショットでは、淡い緑色の背景色が与えられています。チャートの下部に表示され、チャートに表示されている色の説明を行っている凡例を表示するかどうかも、この設定画面から決定することができます。



- ***塗りつぶし*** :ユーザー定義の塗りつぶし定義に加えて、4つの定義済み塗りつぶし定義が利用可能です。色の追加や削除を行うことで、お好きな塗りつぶしの定義を修正することができます。塗りつぶしタブにて選択された色の定義がチャートで使用されます。
- ***大きさ*** :チャートが持つ様々な特徴の大きさを、ピクセルまたはパーセンテージによる比率で指定することができます。
- ***フォント*** :このタブではチャートの凡例、タイトルそしてラベルのフォントプロパティを指定することができます。大きさはピクセル、またはチャートに対するパーセンテージにより指定します。

更に、各種類のチャートは以下のような独自の設定項目を持っています :

- ***円グラフ*** : (i) 最初の項目が開始される角度を指定します。(ii) 項目が並べられる向きを指定します。(iii) 輪郭の色を指定します。(iv) 色をハイライトするか指定します (3D 円グラフでは、影と透明度を指定することができます)。(v) ラベルを表示するかどうか指定します。(vi) 値やパーセンテージをラベルに追加するか、そしてパーセンテージに表示される桁数を指定します。
- ***棒グラフ*** : 全般にて X 軸 とY-軸を入れ替えることで、水平の棒グラフを作成することができます (2次元棒グラフのみ)。 棒にて輪郭の描画と影付けを行ないます (輪郭の描画は2次元棒グラフのみ)。 X-軸にて X-軸のラベルと色、そして垂直の補助線をセットします。 Y-軸にて Y-軸のラベルと色、水平の補助線、表示する値の範囲、そしてY-軸の目盛り間隔を指定します。 Z-軸にて Z-軸のラベルと色を指定します (3次元棒グラフのみ)。 (3D にて) 垂直の傾き、水平の回転、そして視界の広さを指定します。
- ***折れ線グラフ*** : 全般にて X-軸とY-軸を入れ替えます。 ラインにて プロットの点をどのように表示するか指定します。 X-軸にて X-軸のラベルと色、そして垂直の補助線をセットします。 Y-軸にて Y-軸のラベルと色、水泳の補助線、表示する値の範囲、そしてY-軸の目盛り間隔を指定します。
- ***ゲージ*** : (i) ゲージが開始される角度 (アングル) と ゲージの値が表示される角度を指定します (円ゲージのみ)。(ii) 表示される値の範囲を指定します (円ゲージならびに棒ゲージ)。(iii) 大きな目盛りの間隔と小さな目盛りの間隔を指定します (円ゲージならびに棒ゲージ)。(iv) ダイヤルや針、境界線などの色を指定します。

3.9.8 チャートの設定と外観

チャートの設定は次の通り構成されています :

- 基本チャート設定: 最も基本的なチャートの種類の設定です。チャートの種類を選択するには、チャートウィンドウのツールバー内の **[種類の変更]** をクリックします。種類の変更ダイアログが表示されます。
- チャートの詳細設定によりチャートの外観 (タイトル、凡例、色、フォントなど) を変更することができます。詳細設定は外観の変更ダイアログで定義することができます。このダイアログにアクセスするには、チャートウィンドウのツールバー内の **[外観の変更]** をクリックします。

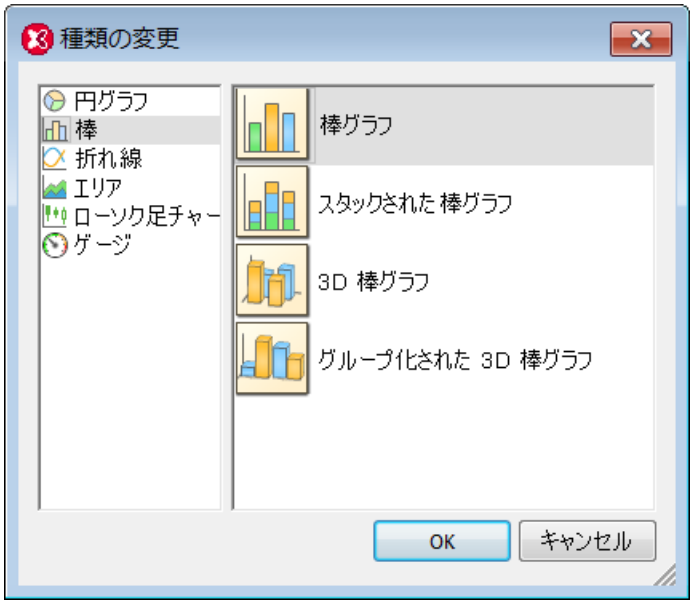
チャートの基本設定

このセクション：

- チャートの種類の設定
- チャートの種類のリスト
- 他の基本設定

チャートの種類の設定

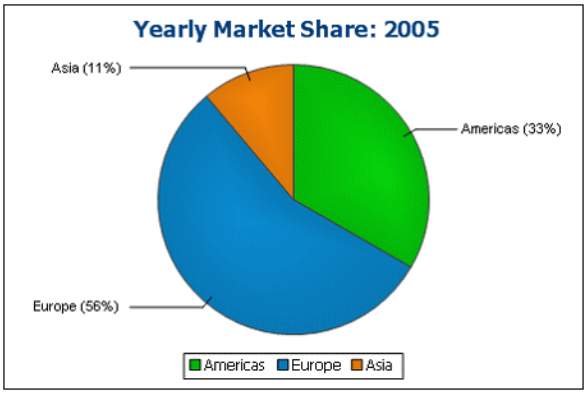
最も基本的な設定はチャートの種類です。チャートの種類を選択するには、チャートウィンドウのツールバーの**[種類の変更]** 種類の変更をクリックします。



チャートの種類は、チャートウィンドウにある**[種類の変更]** ボタンをクリックすることで表示される、種類の変更ダイアログボックスにて定義されます。(上のスクリーンショット)。チャートの種類を選択して、**[OK]** をクリックします。

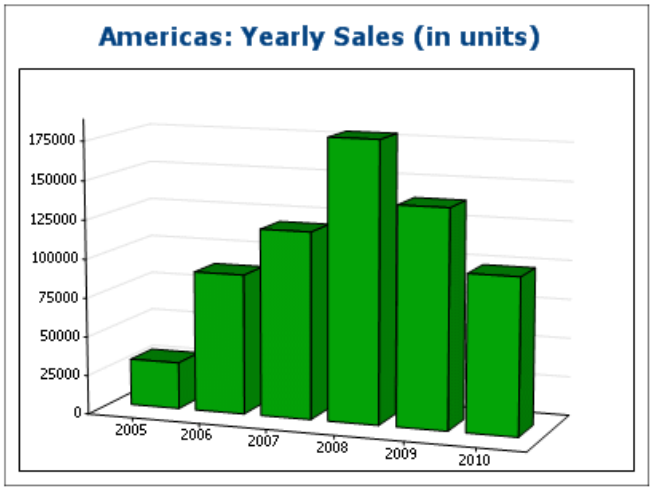
▾ *円チャート*

円グラフでは、単一軸の値を表示し、別の軸により値のラベル付けを行います。

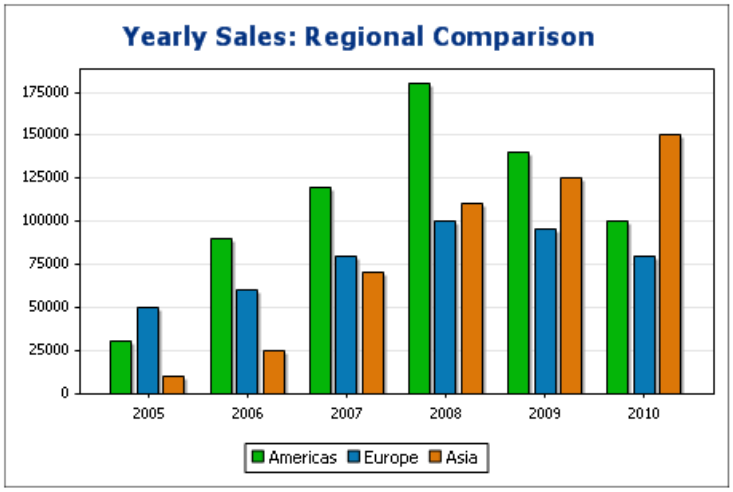


- *棒グラフ*

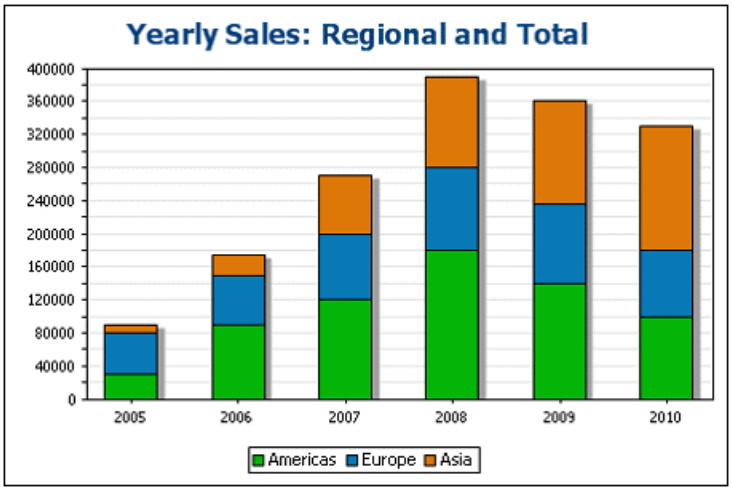
  棒グラフでは、X-軸に数値以外の値を使用することができます。



上のサンプルで表示されるように次のセットの3つの値を使用することもできます : (i) 地域、 (ii) 年度、 (iii) 売り上げ。 棒グラフは 2D ならびに 3D にて表示することができます。

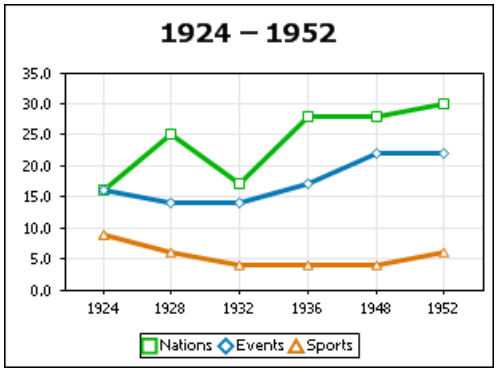


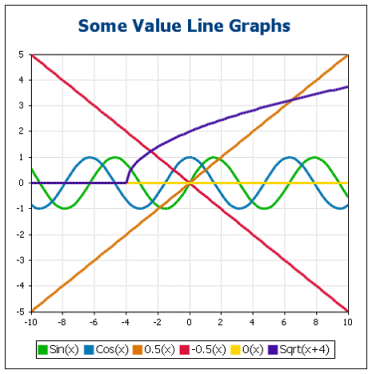
更に、スタックされた棒グラフを使用することで、チャートに含まれている系列の合計を表示することができます。



- *折れ線グラフ*

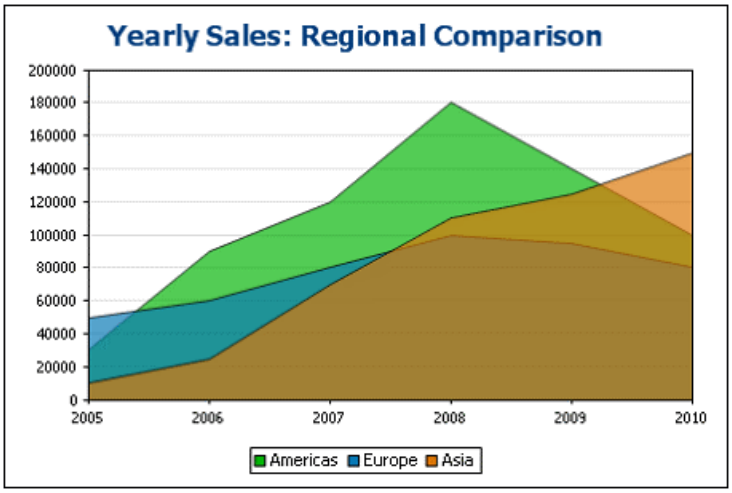
通常の折れ線グラフ（下左のスクリーンショット）と値付き折れ線グラフ（下右のスクリーンショット）の違いは、値付き折れ線グラフでは、X-軸に数値カラムしか使用することができない点にあります。折れ線グラフのX-軸に文字列を使用したい場合には、**[種類の変更]** ダイアログボックスの右ペインから折れ線グラフを選択してください。





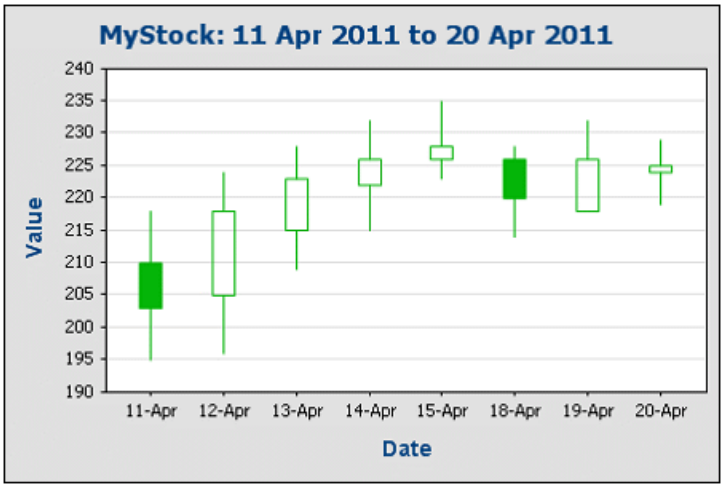
- *エリアチャート*

エリアチャートとは、線の下の面が色付けされる折れ線グラフの1種類です。エリアチャートはスタックすることができます。



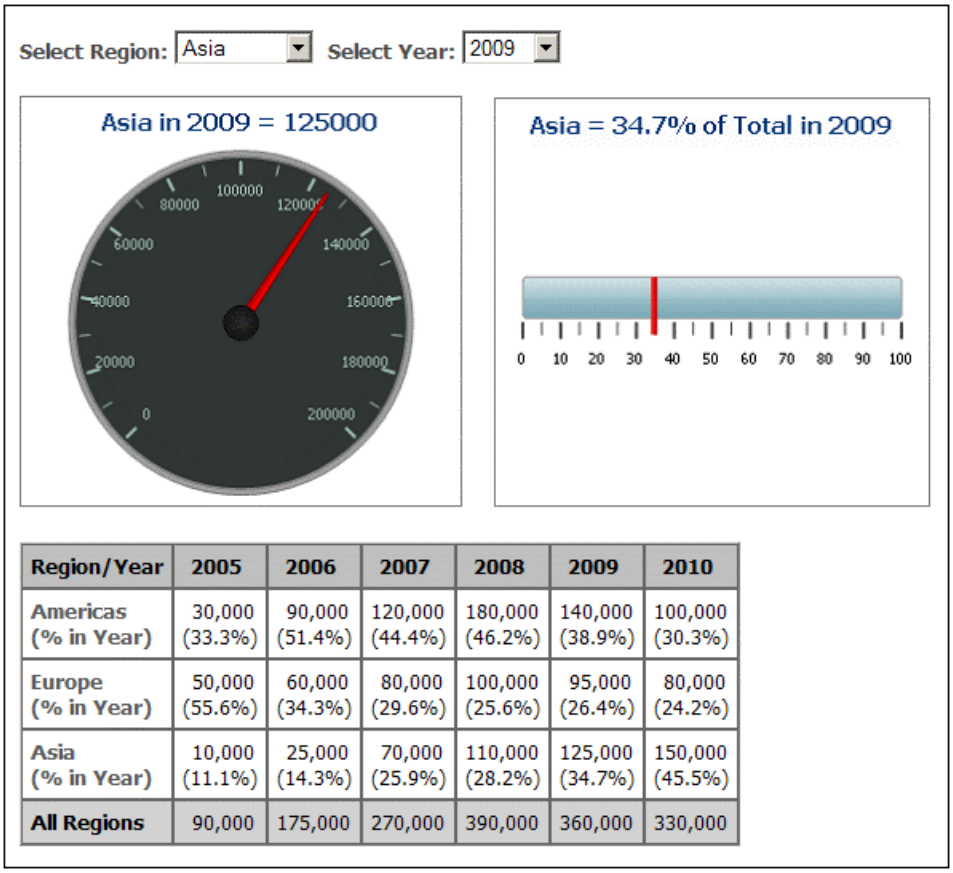
- *ローソク足チャート*

  ローソク足チャートは、ある期間内における証券やデリバティブ、または通貨といった値の動きを表示するために使用されます。チャートには資産の動きが表示され、毎日の終値だけでなく、高値、安値、そしてオプションとして始値を表示することができます。4つの系列（始値、高値、低値、終値）を定義すると、チャートは以下のように表示されます。



- *ゲージチャート*

  ゲージチャートには1つの値が最低ならびに最高値とともに表示されます。



Region/Year	2005	2006	2007	2008	2009	2010
Americas (% in Year)	30,000 (33.3%)	90,000 (51.4%)	120,000 (44.4%)	180,000 (46.2%)	140,000 (38.9%)	100,000 (30.3%)
Europe (% in Year)	50,000 (55.6%)	60,000 (34.3%)	80,000 (29.6%)	100,000 (25.6%)	95,000 (26.4%)	80,000 (24.2%)
Asia (% in Year)	10,000 (11.1%)	25,000 (14.3%)	70,000 (25.9%)	110,000 (28.2%)	125,000 (34.7%)	150,000 (45.5%)
All Regions	90,000	175,000	270,000	390,000	360,000	330,000

他の基本設定

チャート設定ペインでは、チャートのタイトルを設定することができます (*下のスクリーンショット参照*)。

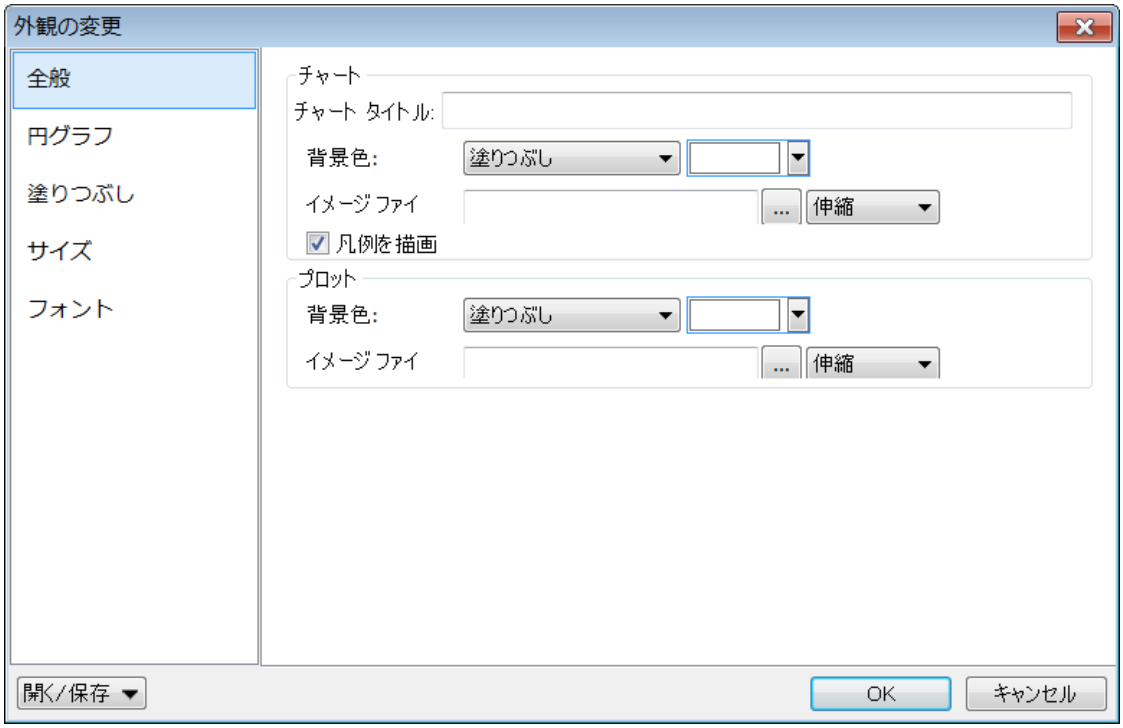
チャートの詳細設定

このセクション:

- 詳細設定へのアクセス
- 詳細設定の概要
- チャート設定のロード、保存、リセット

詳細設定へのアクセス

チャートの詳細設定にアクセスするには以下を行います :チャートウィンドウのツールバー内の **[外観の変更]** をクリックします。これにより特定のチャート種類のための外観の変更ダイアログが表示されます。 (*下のスクリーンショットは円グラフの外観の変更ダイアログを表示しています*)。



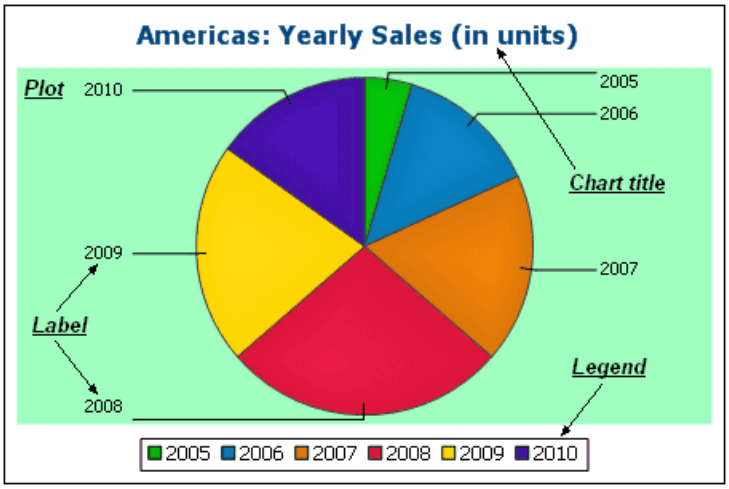
詳細設定の概要

詳細設定はすべてのチャートの種類に共通するタブとチャートの種類に特有のタブにより整理されています。

チャート設定全般

- *全般*

  チャートのタイトル (*下のスクリーンショット参照*) は基本設定と同じですが (*上を参照*) 詳細設定でも編集することができます。このダイアログの他の設定は、チャートの背景色とプロットです。下のスクリーンショットでは、プロットには薄い緑色の背景色が与えられています。チャートおよび/またはプロットの背景のイメージとしてイメージファイルを設定することもできます。このイメージはチャートまたはプロットのエリア全体をカバーするように拡大、(チャートまたはプロットの) 2つのディメンションにズームがマッチするようにズーム、中央揃え、並べて表示することができます。凡例はチャートの配色コードのキーとなりオン/オフすることができます。



- *配色パターン*

  4つの設定済み配色パターンとユーザーにより定義される配色パターンを使用することができます。色を配色パレットに追加または削除することにより配色パターンを変更することができます。このタブにより選択される配色パターンはチャートで使用されます。

- *サイズ*

  ピクセルまたはパーセンテージ率としてチャートのさまざまな部分のサイズを設定することができます。

- *フォント*

  チャートのタイトル、凡例、ラベルのプロパティをこのタブで指定することができます。サイズはチャートのサイズのパーセンテージまたは絶対ポイントとして設定できます。

- *ロード/保存ボタン*

  設定はXML ファイルに保存され、正しい構成を持つXML ファイルからロードされることができます。構成を確認するには、チャートの設定を保存しXML 、ファイルを開きます。このボタンをクリックして、チャートの設定をディフォルトに再設定することもできます。

特定の種類のチャートの設定

- *円グラフ*

  設定 : (i) 最初の扇形が描かれる角度 ; (ii) 扇形が描かれる方向 ; (iii) アウトラインの色 ; (v) 色に対するハイライト有無 (3D 円グラフ: ドロップシャドウ 透明度の使用); (v) ラベルの有無 ;および (vi) 値とパーセンテージがラベルに追加されるか、およびパーセンテージへの小数点の数。

- *棒グラフ*

  設定 :*(全般)*X とY 軸を交換して横向きの棒グラフを生成する;*(棒)*棒のアウトラインおよびドロップシャドウ (ドロップシャドウは 2D 棒グラフのみ);*(X-軸)* X-軸の ラベルと配色と垂直方向の枠線 ;*(Y-軸)*Y-軸のラベル、配色と水平方向の枠線、値の範囲の表示、Y-軸の目盛 ;*(Z-軸、 3D のみ)*Z-軸のラベルと配色 ;*(3D)* 垂直方向へ

の傾斜、水平方向への回転、ビューの幅。

- *折れ線グラフ*

  設定：*(全般)* X とY 軸を交換してグラフの生成；*(ライン)* プロットポイントの有無；*(X-軸)* X-軸のラベルと配色と垂直方向の枠線；*(Y-軸)* Y-軸のラベル、配色と水平方向の枠線、値の範囲の表示、Y-軸の目盛。

- *ゲージチャート*

  設定：(i) ゲージが開始する角度とスケールの掃引角度；(i) 表示される値の範囲；(ii) メジャーおよびマイナーなメモリの間隔と配色；(v) ダイアル、針、境界線の配色。

- *エリアチャート*

  エリアの透明度は 0 (透明度無し) から 255 (最大透明度) の値で設定することができます。この場合、スタックされていないエリアチャートでの透明度の使用は他のエリアの下にあるエリアをビューアが可視することができます。エリアのアウトラインを指定することができます。

- *ローソク足チャート*

  塗りつぶしの色は2つの場面で使用することができます：(i) 終値が始値よりも高い場合、(i) 始値が終値よりも高い場合。後者の場合、系列の配色もオプションとして使用することができます。系列の色は、外観の変更ダイアログの配色パレットタブで指定することができます。

チャート設定のロード、保存、リセット

チャート設定は、XML ファイルに保存することのできるデフォルトの設定と異なります。これらの設定は、後にチャートの設定としてロードすることができ、時間と動力をセーブすることができます。**[ロード保存]** ボタン (このセクションの最初のスクリーンショット参照) をクリックすると以下のオプションを使用することができます：

- **デフォルトとして設定**: 設定に加えられた変更を拒否し、**すべての** 設定セクションのデフォルトの設定を復元します。
- **ファイルからロード**: XML ファイルに以前保存された設定をインポートすることができます (*次のコマンドを参照*)。コマンドは、必要なファイルの場所を入力することのできる **[参照]** ダイアログを表示します。
- **ファイルに保存**: **[名前を付けて保存]** ダイアログボックスを開きます。保存するXML ファイルを指定することができます。このファイルはデフォルト設定と異なる設定のファイルをリストします。

一般

[外観の変更] ダイアログボックスの一般セクションでは、チャートのタイトル、凡例の追加や削除、背景イメージ、更に棒グラス 折れ線グラス エリアチャート、ローソク足チャートに対してはチャートの方向を指定することができます。

チャート

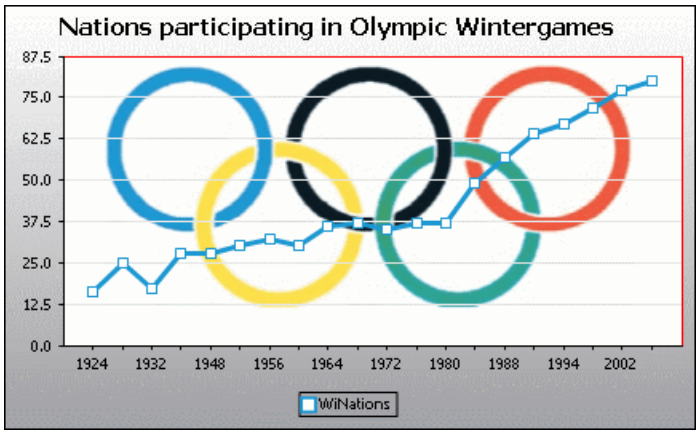
チャートタイトルフィールドにチャートを説明する様なタイトルを入力し、チャート全体に適用される背景色をドロップダウンリストから選択します。塗りつぶしや、垂直または水平グラデーション、そしてグラデーションでは2つの色を選択します。更に、背景色の代わりに背景イメージを指定し、ドロップダウンリストから表示オプションを選択することもできます：

- 伸縮：イメージがチャートの高さならびに幅に伸ばされて表示されます
- 合わせる：イメージが、アスペクト比を保ったままチャートのフレームに合うように表示されます
- 中央に表示：イメージがオリジナルのサイズで中央に表示されます
- 並べて表示：イメージがチャートより小さい場合、イメージの複製が背景エリアに表示されます

凡例を描画チェックボックスはデフォルトでオンになっており、チャート内に凡例を表示したくない場合はチェックを外してください。

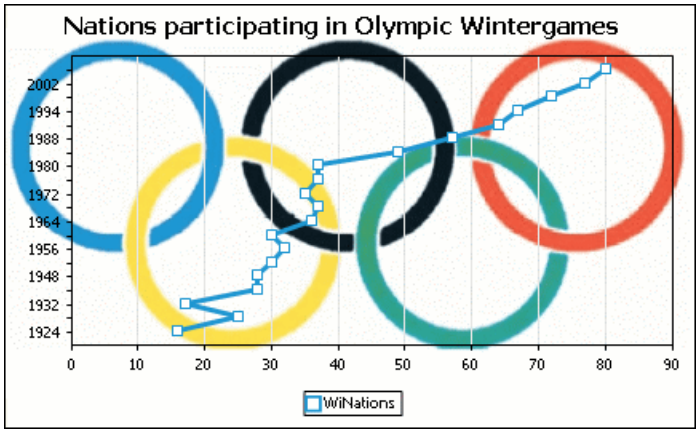
プロット

プロットとは実際のチャートが表示される領域です。プロットの周りに線をひいたり、プロット領域に対して背景色やイメージを指定することができます。以下のスクリーンショットでは、プロット領域の色は白のままでチャートの背景色がグレー（垂直グラデーション）に、境界線の色が赤に変更され、背景イメージが加えられています。



方向

少数の系列と、多くの値がデータに存在する場合、X-軸とY-軸を入れ替えた方が、より見やすい表示となることもあります。以下のスクリーンショットでは、プロットの背景色が「透明」にセットされ、イメージがチャートの背景に適応されていることにも注目してください。



このオプションは円グラフならびにゲージチャートには適用できないことに注意してください。

各チャートの機能

[外観の変更] ダイアログボックスでは、チャートの種類に関連した機能の定義を行うことができます。

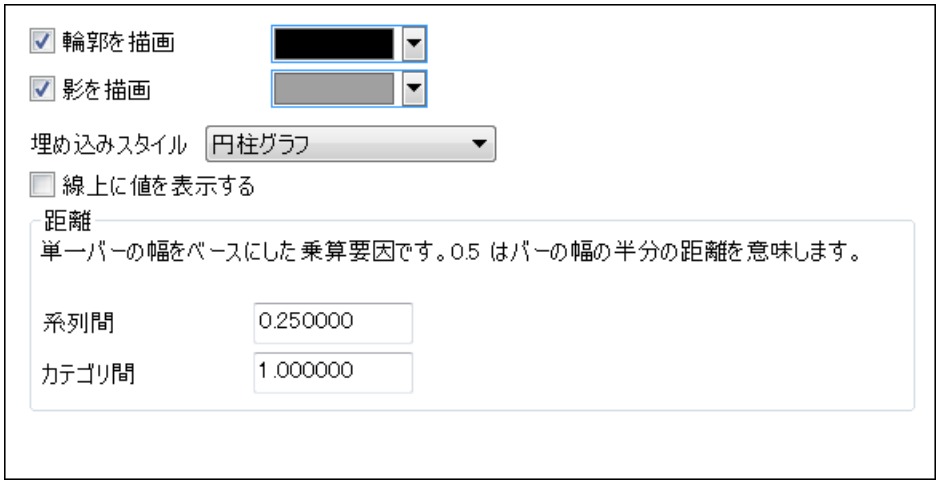
円グラフ

2D ならびに 3D バージョンで行うことのできる設定にそこまで違いはありません。2D の円グラフではハイライトを付与することができます。

3D の円グラフでは、影の描画や透明度、3D の傾きを指定することができます。

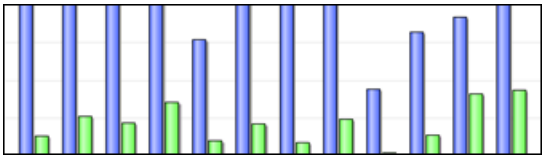
[開始角度] の値により、選択されたカラム内にある最初の行が表示されるチャートの位置を指定します。0° の角度が時計の12時方向に対応します。
凡例やラベルを表示したり、値やパーセントをラベルに付与したり、パーセントに使用される桁数を指定することができます。
[輪郭を描画] チェックボックスの隣から選択できる色により、チャートや個々の円グラフセグメントの周りに描かれる境界線の色を指定することができます。時計回りのチェックボックスにより、行の表示を時計回りの、または反時計回りに行うかを指定できます。
3D の円グラフでは、影の描画やその色、チャートの透明度、そして 3D の傾きを指定することができます。2D のチャートでは、ハイライトを描画オプションにより、更なる側面をチャートに付与することができます。

棒グラフ

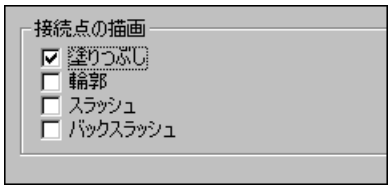


棒グラフのために以下を設定することができます :

- 棒に輪郭を追加し、色を定義することができます
- 2D の棒グラフでは、影を付けることができ、色を定義することもできます (3D 棒グラフではこのオプションを使用することはできません)。
- デフォルトで、棒の形式は、円柱ですが、『グラデーション (縦方向)』または『塗りつぶし』をドロップダウンリストから選択することができます。
- (Y-軸上のバーの高さに対応する)棒の値を棒上に表示することができます。
- 棒グループの系列と棒グループの距離は、単一の棒の幅の小数として指定することができます。例えば、下のスクリーンショットでは、青い系列と緑の系列から構成される棒グループが表示されており、系列間の距離は、棒の幅の 25% (=0.25) に設定されています。棒グループの距離は棒の幅の 100% (=1.0) に設定されています。



折れ線グラフ

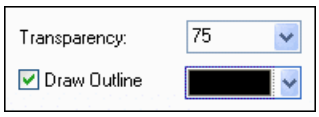


チャート内にある値を結びつける接続点を描画するには、接続点の描画グループボックスにあるチェックボックスのうち少なくとも1つにチェックを入れる必要があります。系列をマークするために、最大5つの形が使用されます：正方形、ひし形、三角形、逆三角形、円。6つ以上の系列がチャート内に存在する場合、接続点のオプションを2つ以上選択することで、接続点の形を組み合わせて使用することができます。以下のスクリーンショットでは、**塗りつぶし**と**スラッシュ**の2つが選択されており、6つ目以降の系列にはスラッシュが使用されます。

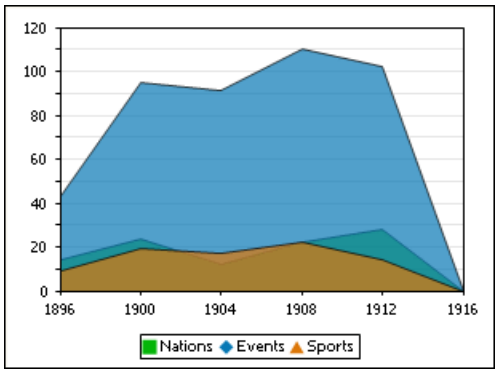
線を描くオプション (i) 接続点のみ (ii) 接続点と線により線をを使用して描くことができます。

接続点は棒グラフと値付き棒グラフの両方で使用することができます。

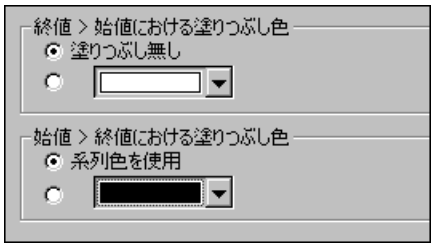
エリアチャート



エリアチャートでは透明度を指定することができ、1つの系列がチャート内にある他の系列により隠れてしまうのを防ぐことができます。更に、データエリアに対して輪郭を描画、ならびに色を指定することができます。(*下のスクリーンショット参照*)。



ローソク足チャート



始値と終値の値が系列として既に定義されている場合、使用される色と、終値が始値よりも大きい場合、ローソクの塗りつぶしを行うかを指定することができます。

ゲージチャート

角グループボックスにある**開始**値により 0 のある位置と、**[可動角度]** によりチャートの表示に使用される角度が指定されます。値の範囲グループボックスでは、表示する最小ならびに最大値を指定することができます。大きな目盛り間隔では目盛りが値とともに、小さな目盛り間隔では値なしで表示され、それぞれに異なる色を指定することができます。色のグループボックスでは、ダイアルや針、チャートを囲む境界線や針のベース (中央部にある部分の針) の色を指定することができます。

色

選択されたチャートの種類によりXMLSpy ではチャート内で使用される色の指定を行う2つのセクションが用意されます：

- 円グラフ、折れ線グラフ、エリアチャート、ローソク足チャートに対して使用される塗りつぶし
- ゲージチャートに対して使用される色の範囲

塗りつぶし

[外観の変更] ダイアログボックスにある塗りつぶしセクションでは、(デフォルト、グレイスケール、カラー、パステルから成る) 4つ

の定義済み塗りつぶしセットが用意されており、カスタマイズすることができます。更に、独自の色を最初から作成することもできます。

最初の色が最初の系列に対して使用され、2つ目の色が2つ目の形式に対して使用されます。色を選択してマウスのドラッグ操作により別の位置に移動することで色の順番を変更することができます。新たな色を追加したり既にある色を削除するには、それぞれ対応するボタンをクリックしてください。ローソク足チャートでは、最初の色だけが使用されます。

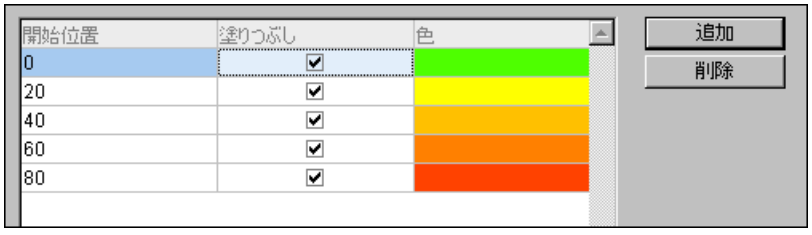
1つ以上のオーバーレイチャートレイヤーをチャートウィンドウに追加した場合、外観の変更ダイアログボックスに、デフォルトで選択されている**[前のチャートレイヤーから継続して色を使用する]** ラジオボタンが表示されます。

このラジオボタンが選択されていると、前のレイヤーから色が継承され、そのオーバーレイに対して独自の色を選択できなくなります。アクティブなレイヤーにおける系列は、その前のレイヤーで使用された色の集合が継続して使用されます。これにより、チャートウィンドウ内に表示される系列全てに異なる色を持たせ、より簡単に系列が識別できるようになります。

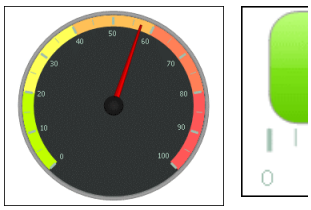
このオプションは任意のレイヤーで無効にすることができ、異なる色の集合を指定して、更に続くレイヤーにてそれらの色を継承することもできます。

色の範囲

ゲージチャートでは、値の範囲に対して色を指定することで、ゲージの外観をカスタマイズすることができます。

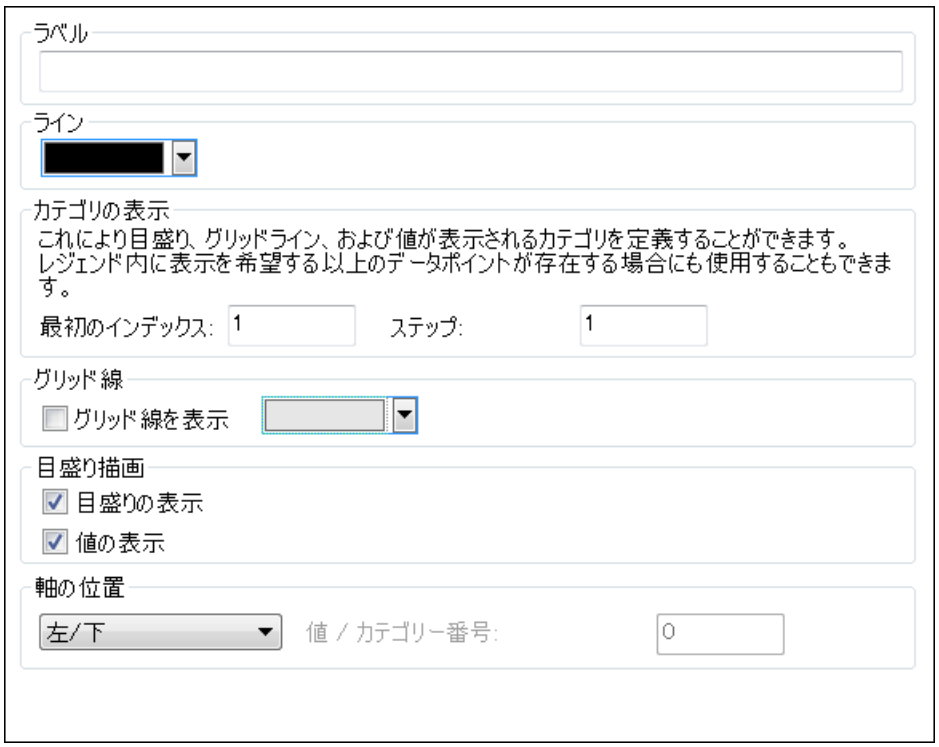


上のスクリーンショットで定義された設定により、ゲージチャートは以下のような外観になります：

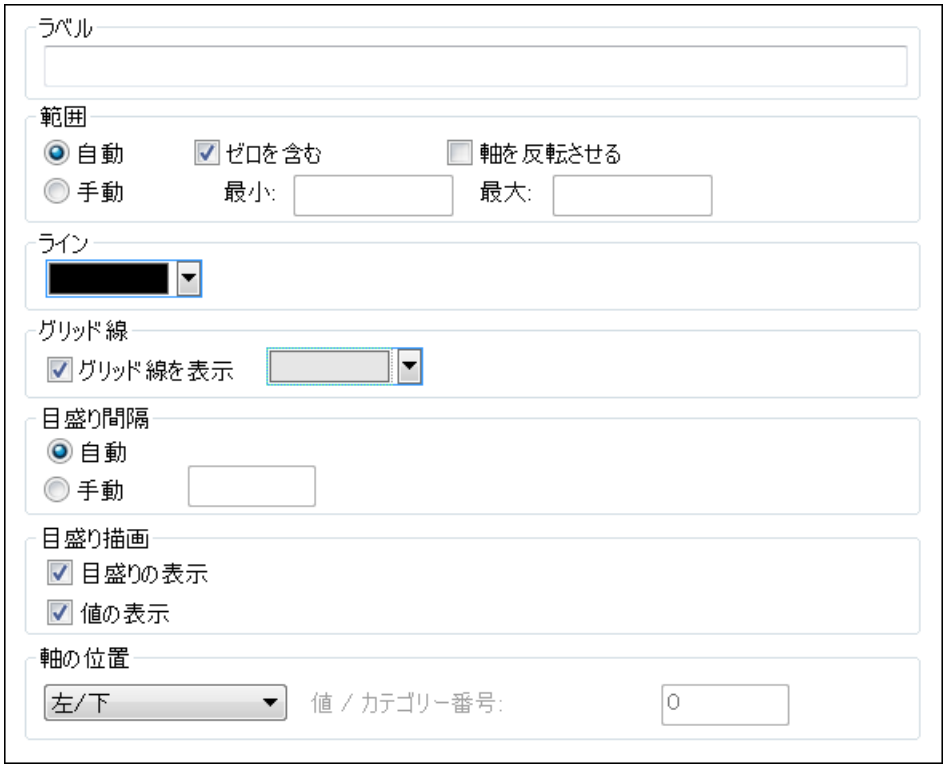


X-軸

[外観の変更] ダイアログボックスのX-軸セクションでは、軸のラベルの入力や、軸ならびにグリッド（表示されている場合）の色を定義、目盛りや軸の値を表示するか指定することができます。このセクションにおける記述は全ての棒グラフ、折れ線グラフ、エリアチャート、ローソク足チャートに対して共通するものです。



折れ線グラフの場合、値の範囲ならびに表示される目盛りの間隔を定義することもできます。



ラベル
ラベルフィールド内に入力されたテキストは、X-軸の説明文として軸の下に表示されます。

範囲
デフォルトでは範囲グループボックスの自動ラジオボタンが選択されています。チャートの一部分における詳細を表示したい場合、手動ラジオボタンを選択して、**最小**ならびに**最大値**を対応するフィールドに入力します。
X-軸に使用されるカラムが 0 を含まない場合、**[ゼロを含む]** チェックボックスを無効にして、X-軸の表示を系列内にある最小の値から始めることもできます。

ライン
チャート内に表示される軸の色は、**ライン**ドロップダウンリストから指定することができます。既に表示されている色を選択するか、**[他の色 ..]** ボタンをクリックして標準的な色を選択するか、カスタマイズされた色を作成することができます。カスタムタブにある**[選択 ..]** ボタンをクリックすると、スポイトが表示され、画面上に表示されている色を選択することができます。

グリッド線
グリッド線を表示チェックボックスが有効になっている場合、隣にあるドロップダウンボックスから色を選択することができます。

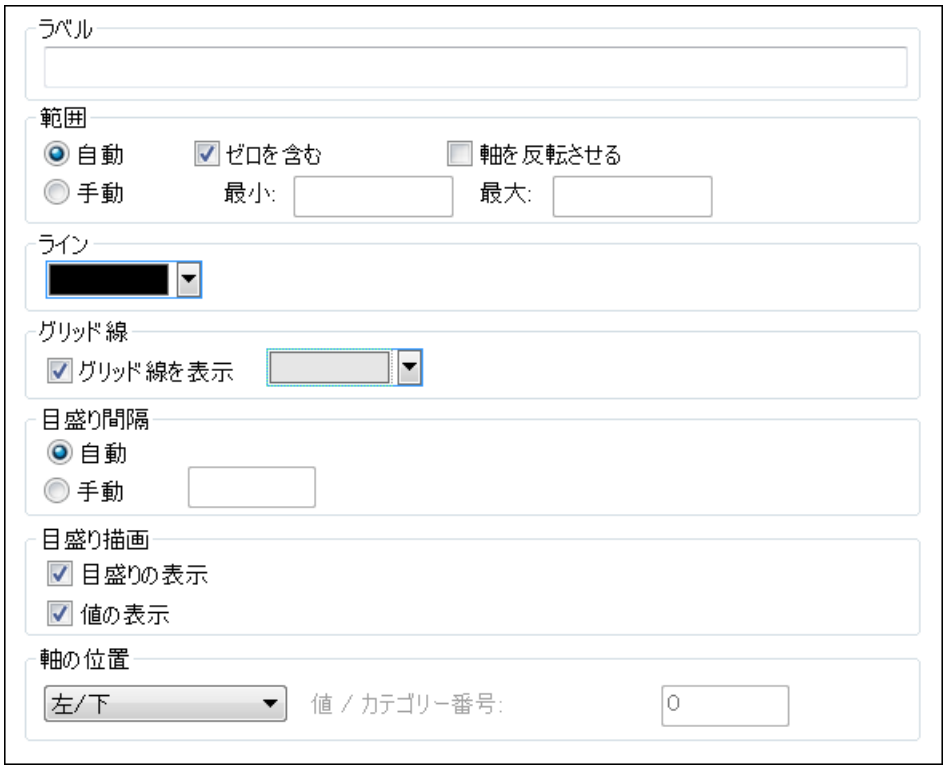
目盛り間隔
デフォルトで表示される目盛り間隔で都合が悪い場合は、目盛り間隔グループボックスにある手動ラジオボタンを有効にして、目盛り間の値をフィールドに入力してください。

目盛り描画
軸にある目盛りや、軸上にある値の表示を切り替えることができます。

軸の位置
軸が表示される位置をドロップダウンリストから選択することができます。「値 /カテゴリー番号」を選択すると、プロット内の自由な位置に軸を配置することができます。

Y-軸

[外観の変更] *ダイアログボックスのY-軸セクション*では、軸のラベルの入力、軸の色や 表示される場合は) グリッド線の定義、値の範囲指定、目盛りの表示やその間隔、そして軸の値を表示するかなどを設定することができます。このセクションにおける記述は全ての棒グラフならびに折れ線グラフに対して共通するものです。



ラベル
ラベルフィールドに入力された値は、Y-軸の説明として軸の左側に表示されます。

範囲
デフォルトでは範囲グループボックスの自動ラジオボタンが選択されています。チャートの一部分における詳細を表示したい場合、手動ラジオボタンを選択して、**最小**ならびに**最大値**を対応するフィールドに入力します。
X-軸に使用されるカラムが 0 を含まない場合、**[ゼロを含む]** チェックボックスを無効にして、X-軸の表示を系列内にある最小の値から始めることもできます。

ライン
チャート内に表示される軸の色は、**ライン**ドロップダウンリストから指定することができます。既に表示されている色を選択するか、**[他の色 ..]** ボタンをクリックして標準的な色を選択するか、カスタマイズされた色を作成することができます。カスタムタブにある**[選択 ..]** ボタンをクリックすると、スポイトが表示され、画面上に表示されている色を選択することができます。

グリッド線
グリッド線を表示チェックボックスが有効になっている場合、隣にあるドロップダウンボックスから色を選択することができます。

目盛り間隔
デフォルトで表示される目盛り間隔で都合が悪い場合は、目盛り間隔グループボックスにある手動ラジオボタンを有効にして、目盛り間の値をフィールドに入力してください。

目盛り描画
軸にある目盛りや、軸上にある値の表示を切り替えることができます。

軸の位置
軸が表示される位置をドロップダウンリストから選択することができます。「値 /カテゴリー番号」を選択すると、プロット内の自由な位置に軸を配置することができます。

Z- 軸

[外観の変更] ダイアログボックスのZ-軸セクションでは、軸のラベルの入力や、軸ならびに色を定義、軸上の目盛りを表示するか指定することができます。このセクションにおける記述は全ての 3D 棒グラフ (3D 棒グラフならびにグループ化された 3D 棒グラフ) 対して共通するものです。

ラベル
ラベルフィールドに入力された値は、Z-軸の説明として軸の右側に表示されます。

ライン
チャート内に表示される軸の色は、**ライン**ドロップダウンリストから指定することができます。既に表示されている色を選択するか、**[他の色 ..]** ボタンをクリックして標準的な色を選択するか、カスタマイズされた色を作成することができます。カスタムタブにある**[選択 ..]** ボタンをクリックすると、スポイトが表示され、画面上に表示されている色を選択することができます。

目盛りの描画
目盛りの表示を切り替えることができます。

3D アングル

[外観の変更] ダイアログボックスの 3D アングルセクションでは、チャートの 3D 的な外観を 3D 棒グラフに対して設定することができます。

傾きの値によりX-軸に対する角度が決定され、回転の値によりY-軸に対する角度が決定されます。**[自動チャート軸サイズ]** チェックボックスにチェックを入れることで、チャート軸の大きさをチャートのウィンドウに自動的に合わせることができます。

[自動チャート軸サイズ] チェックボックスが有効になると、現在表示されているチャートの大きさに対して最適なX-軸ならびにY-軸の大きさがXMLSpy により計算されます。チャートの幅と高さは、チャートウィンドウの大きさを変更することで動的に変更されます。

サイズ

[外観の変更] ダイアログボックスの大きさセクションでは、様々な余白や軸、またはゲージ目盛りの間隔を定義することができます。以下に示す全てのプロパティが全てのチャートに対して使用できるわけでは無いことに注意してください。

一般

外部の余白	プロットとチャートウィンドウ端の空間
タイトルからプロット	チャートタイトルとプロット上部の端との空間
凡例からプロット	プロット下部の端と凡例との空間

円グラフ

プロットからラベル	円グラフにて、円グラフの左端ならびに右端とラベル間の空間
グラフの高さ	3D 円グラフの高さ
グラフの影	3D 円グラフにて (有効化されていた場合) 表示される影の長さ

X-軸

X-軸から軸ラベル	棒グラフと折れ線グラフにて表示されるX-軸とラベル間の空間
X-軸からプロット	2D 棒グラフと折れ線グラフにて表示されるX-軸とプロット間の空間
X-軸目盛りサイズ	棒グラフと折れ線グラフにて表示されるX-軸上の目盛り長さ

Y-軸

Y-軸から軸ラベル	棒グラフと折れ線グラフにて表示されるY-軸とラベル間の空間
Y-軸からプロット	2D 棒グラフと折れ線グラフにて表示されるY-軸とプロット間の空間
Y-軸目盛りサイズ	棒グラフと折れ線グラフにて表示されるY-軸上の目盛り長さ

Z-軸

Z-軸から軸ラベル	3D 棒グラフにて表示されるZ-軸とラベル間の空間
Z-軸目盛りサイズ	3D 棒グラフにて表示されるZ-軸上の目盛り長さ

ライン描画

接続点のサイズ	折れ線グラフにてチャート内の値をマークする接続点の大きさ

3D 軸サイズ

ベースの手動X-軸サイズ	3D 棒グラフにおいてX-軸の長さとチャートウィンドウの大きさの関係を定義します。3D アングルセクションにある**自動チャート軸サイズ**のチェックを外さなければサイズが自動的に計算されることに注意してください
ベースの手動Y-軸サイズ	3D 棒グラフにおいてY-軸の長さとチャートウィンドウの大きさの関係を定義します。3D アングルセクションにある**自動チャート軸サイズ**のチェックを外さなければサイズが自動的に計算されることに注意してください
Z-軸系列余白	3D 棒グラフのZ-軸における各系列間の距離

ゲージ

境界線の幅	円ゲージチャートのゲージ周りに表示される境界線の幅

ゲージ間隔

領域線から目盛りまでの距離	円ゲージチャートにおいて、領域内側の端と値を表示する目盛りの空間
大きな目盛りの長さ	円ゲージチャートに表示される (値を持った) 大きな目盛りの長さ
大きな目盛りの幅	円ゲージチャートに表示される (値を持った) 大きな目盛りの幅
小さな目盛りの長さ	円ゲージチャートに表示される値を持たない目盛りの長さ
小さな目盛りの幅	円ゲージチャートに表示される値を持たない目盛りの幅

ゲージ針

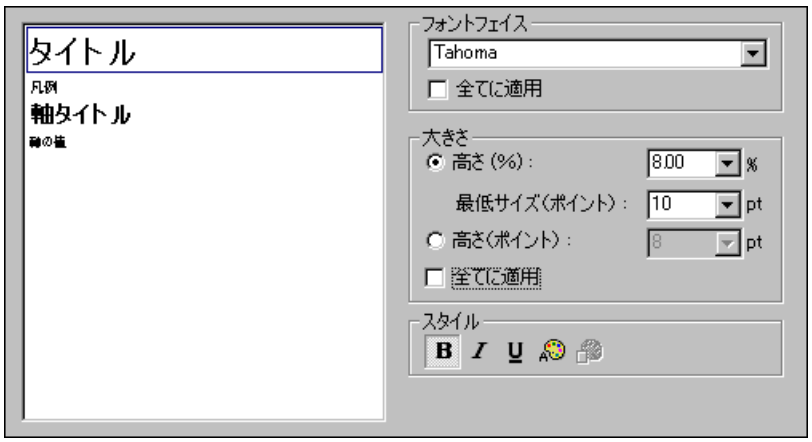
針の長さ	円ゲージチャートにおける針の長さ (パーセントはゲージの直径から計算されるため、50% を超えた値を選択すると、針の先端がゲージの外側に出てしまうことに注意してください!)
ベースにおける針の幅	円ゲージチャートの中心における針の幅 (太さ)
針ベースの半径	円ゲージチャート中央部に表示されるベースの半径

ゲージ色の範囲

境界線から色の範囲までの距離	円ゲージチャートにおける境界線内側にある端と、色の範囲との間にある空間
色の範囲幅	円ゲージチャートにおけるカスタマイズ可能な色が表示される範囲の幅 (パーセントはゲージの直径から計算されることに注意してください!)

フォント

[外観の変更] ダイアログボックスのフォントセクションではチャートウィンドウ内のオブジェクトに使用されるフォント設定を行うことができます。



フォント設定
フォントセクションではチャートウィンドウ内に表示される各要素のフォントフェイス、大きさ、そしてスタイルを変更することができます。大きさはチャートサイズに対するパーセントとして定義することができるとともに、最低サイズをポイントで指定、または絶対サイズをポイントで指定することができます。同一のフォントや大きさを全てのテキスト要素に適用したい場合、**[全てに適用]** チェックボックスにチェックを入れます。
リストボックス内にある要素は以下のように定義されます：

- **タイトル** :チャートの名前
- **凡例** :チャート内で使用されている色の説明
- **ラベル** :円グラフにおける各パイの説明
- **軸タイトル** :棒グラフおよび折れ線グラフにおけるX-、Y-、Z-軸の名前
- **軸の値** :棒グラフおよび折れ線グラフの軸上に表示される単位
- **目盛りの値** :ゲージチャートにて表示される単位

3.9.9 エクスポート

「エクスポート」 ボタンをクリックすることで、以下のオプションを利用することができます：

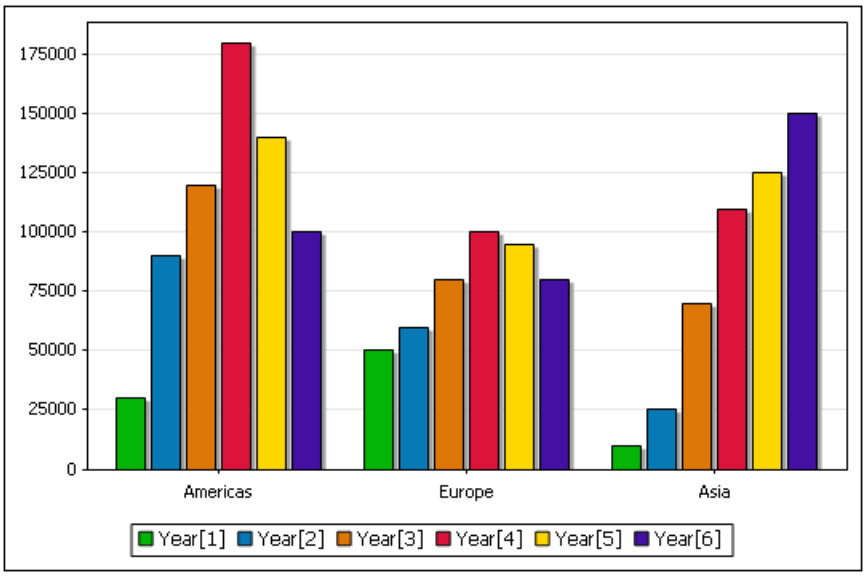
- **チャートをファイルに保存** :イメージの形式をPNG、GIF、BMP、JPG から選択することができます。
- 現在のイメージをクリップボードにコピー または サイズ変更されたイメージをクリップボードにコピー :チャートのイメージをクリップボードにコピーして、他のアプリケーションで使用することができます。
- **このチャートを印刷** :イメージをネットワーク上のプリンターに送信します。イメージの高さ幅は、それぞれページサイズとの比率（パーセンテージ）で指定することができます。
- **XSLT または XQuery コードをクリップボードにコピー** :XSLT または XQuery フラグメントを作成します。両者とも、実質はAltova 拡張関数の `CreateChart` となります。この関数は、他のAltova 拡張関数にて使用することができ、XMLSpy または RaptorXML エンジンにて処理することでチャートの生成を行うことができます。各フラグメントには、`CreateChart` 拡張関数の使用を助けるための例がコメントアウトされた形で作成されます。

3.9.10 チャートの例: 基本

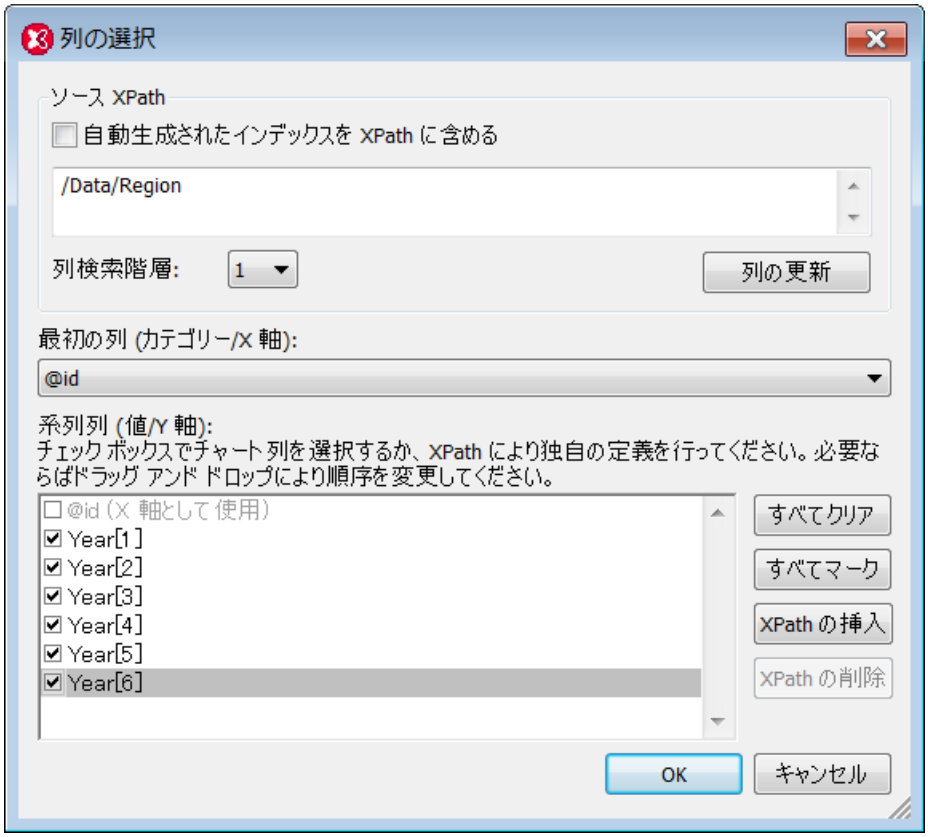
以下のXMLドキュメントを考えてみましょう。このファイルは`YearlySales.xml`というファイル名で`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\Tutorial`に収められています。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
        <Region id="Americas">
                <Year id="2005">30000</Year>
                <Year id="2006">90000</Year>
                <Year id="2007">120000</Year>
                <Year id="2008">180000</Year>
                <Year id="2009">140000</Year>
                <Year id="2010">100000</Year>
        </Region>
        <Region id="Europe">
                <Year id="2005">50000</Year>
                <Year id="2006">60000</Year>
                <Year id="2007">80000</Year>
                <Year id="2008">100000</Year>
                <Year id="2009">95000</Year>
                <Year id="2010">80000</Year>
        </Region>
        <Region id="Asia">
                <Year id="2005">10000</Year>
                <Year id="2006">25000</Year>
                <Year id="2007">70000</Year>
                <Year id="2008">110000</Year>
                <Year id="2009">125000</Year>
                <Year id="2010">150000</Year>
        </Region>
</Data>
```

ここでX-軸に3つの地域が表示され、各地域の年間売上がプロットされるようなグラフの作成を行ないます。グラフは以下の棒グラフにあるようなものになります。



Region 要素をソースXPath として選択するため、このチャートの作成は簡単に行うことができます。ソースXPath 条件式からは 3 つのアイテム（3 つのRegion 要素）を含むシーケンスが取得されます。各 Region 要素はX-軸ならびにY-軸のデータ選択におけるコンテキストノードとなります。



系列に関しては、各地域にあるYear 要素を使用するので、列の検索階層は1で十分となります。Region 要素のid 属性をX-軸に使用します。従って、id 属性の値がX-軸の要素のラベルとして使用されます。チャートには存在する全てのYear 要素を表示したいので、Year 系列は全てチェックします。

「OK」ボタンをクリックすることで、目的のチャートが表示されます。より高度なチャートに関する情報は、チャートの例 詳細を参照ください。

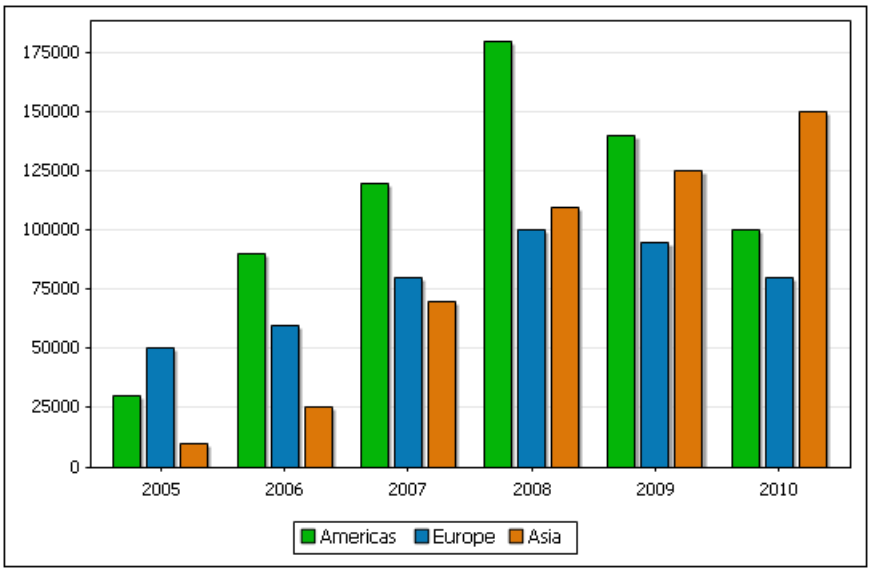
3.9.11 チャートの例 :詳細

以下のXML ドキュメントを考えてみましょう。このファイルはYearlySales.xml というファイル名でC:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\Tutorial に収められています。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
        <Region id="Americas">
                <Year id="2005">30000</Year>
                <Year id="2006">90000</Year>
                <Year id="2007">120000</Year>
                <Year id="2008">180000</Year>
                <Year id="2009">140000</Year>
                <Year id="2010">100000</Year>
```

```
        </Region>
        <Region id="Europe">
                <Year id="2005">50000</Year>
                <Year id="2006">60000</Year>
                <Year id="2007">80000</Year>
                <Year id="2008">100000</Year>
                <Year id="2009">95000</Year>
                <Year id="2010">80000</Year>
        </Region>
        <Region id="Asia">
                <Year id="2005">10000</Year>
                <Year id="2006">25000</Year>
                <Year id="2007">70000</Year>
                <Year id="2008">110000</Year>
                <Year id="2009">125000</Year>
                <Year id="2010">150000</Year>
        </Region>
</Data>
```

ここで、X-軸に年が表示され、毎年ごとに地域間の年間売上比較を行えるようなグラフを作成します。グラフは以下の棒グラフにあるようなものになります。

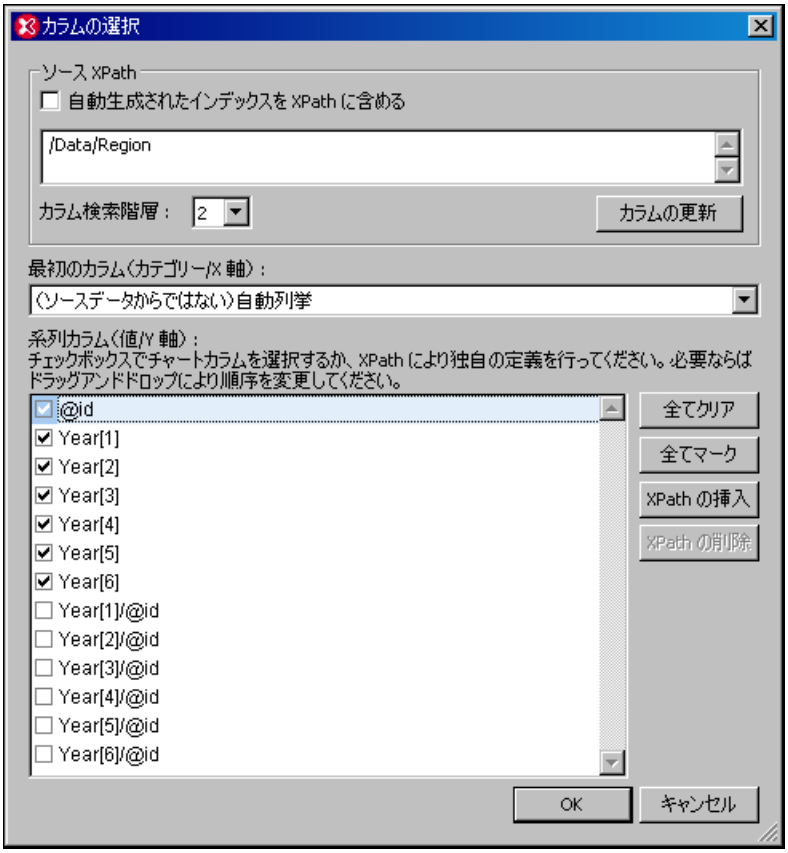


このグラフを作成する2つの方法を説明します。これら2つの説明を通して、異なるデータ選択パラメーターを組み合わせることで目的のグラフを作成する方法を学習します。

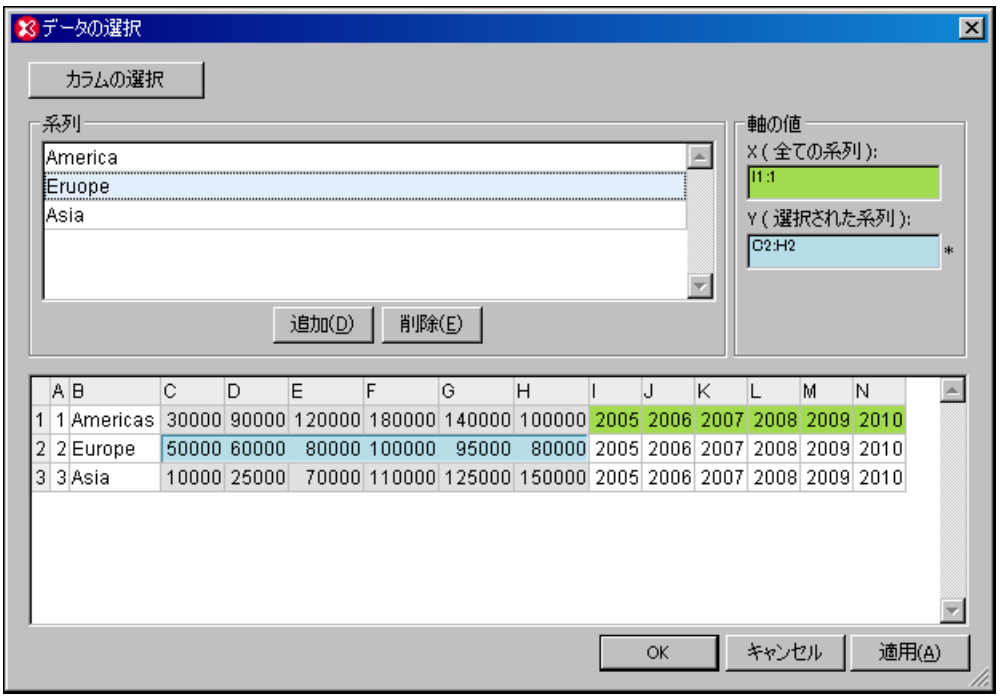
方法その1 軸の値を変更

最初の方法では、列の選択ダイアログにて選択された軸をデータの選択ダイアログにて修正します。ここで注意しなければならない唯一のことは、必要なデータは全てデータの選択ダイアログのチャートデータテーブルにて使用可能になっているということです。

1. 列の選択ダイアログにて、必要な全てのノードがX-軸、Y-軸として選択できる状態であることを確認してください。以下のスクリーンショットにて、列検索階層が2にセットされていることに注目してください。これで`Year/@id`属性を選択することができるようになります。



2. 以下のスクリーンショットにある）データの選択ダイアログにて、チャートデータテーブルには以下の列が含まれています：最初の列がX-軸の選択（自動列挙した選択）で、その他の列が、`Region/@id`、`Year` 要素のコンテンツ、そして `Year/@id` 属性からなる、系列（Y-軸）の列。以下の 2 つの点に注意してください：(i) テーブルには 3 つの行しか存在せず、X-軸の要素数は 3 つになります（しかし我々がX-軸にほしいのは 6 年分の（つまり 6 つの）要素です）。(ii) 系列には 13 の列があります。



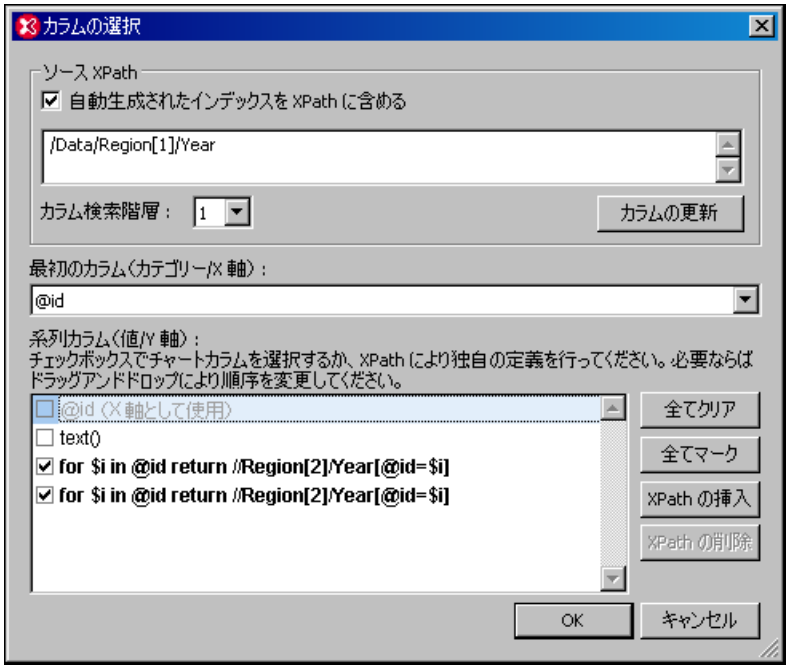
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	1	Americas	30000	90000	120000	180000	140000	100000	2005	2006	2007	2008	2009	2010
2	2	Europe	50000	60000	80000	100000	95000	80000	2005	2006	2007	2008	2009	2010
3	3	Asia	10000	25000	70000	110000	125000	150000	2005	2006	2007	2008	2009	2010

3. 系列ペインでは、13 あるうちの系列行のうち、任意の 10 個を削除し、残った 3 つの名前を、スクリーンショットにあるように `America`、`Europe`、`Asia` に変更します。ここで選択された順序が X-軸の要素にて表示されるラベルの順序になります。
4. 系列ペインにて、`America` の系列を選択します。軸の値ペインにて X-軸のテキストボックスをクリックして、修正を開始します。チャートデータテーブルの `I1` をクリックして、そのまま `N1` までドラッグします。Y-軸テキストボックスにて `C1:H1` と入力するか、ドラッグすることでチャートデータテーブルの `C1` から `H1` までを選択します。
5. Europe と Asia の系列に対しては、`C2:H2` ならびに `C3:H3` をそれぞれ Y-軸に対して選択します。X-軸の選択は `America` の場合と同様になります。
6. 「**OK**」をクリックすると、目的のグラフが生成されます。

メモ X-軸のラベル数が 6 になったため、、チャートデータテーブルにある行の数からデフォルトで定義された X-軸の要素も 3 から 6 に増加します。

方法その2：XPath 条件式により系列を生成する

2 つめの方法では、系列の生成のために XPath 条件式を挿入します。これは、目的の系列に対応するノードを、以下のスクリーンショットにあるように、ソース XPath が子孫として持たないために必要な操作となります。それでもソース XPath は 6 つの X-軸要素を、最初の `Region` 要素の `Year` 要素を選択することで、生成します。かっこの `[1]` により、最初の `Region` 要素を選択するには、*インデックスに含める* チェックボックスを有効にして、**列の更新**ボタンをクリックしてください。

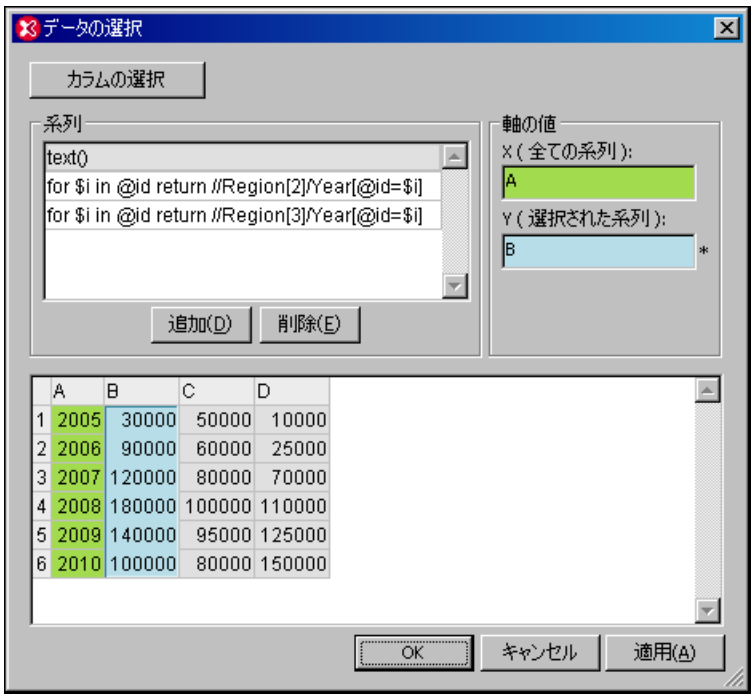


列の選択ダイアログにて、Region[1]/Year 要素は、`@id` と`text()` という2つの子孫しか持っていないことが確認できます。@id 属性をX-軸に選択し、6つのX-軸要素に対して、正しいX-軸のラベルを生成するようにします。チャートデータテーブルは以下のように評価されます：

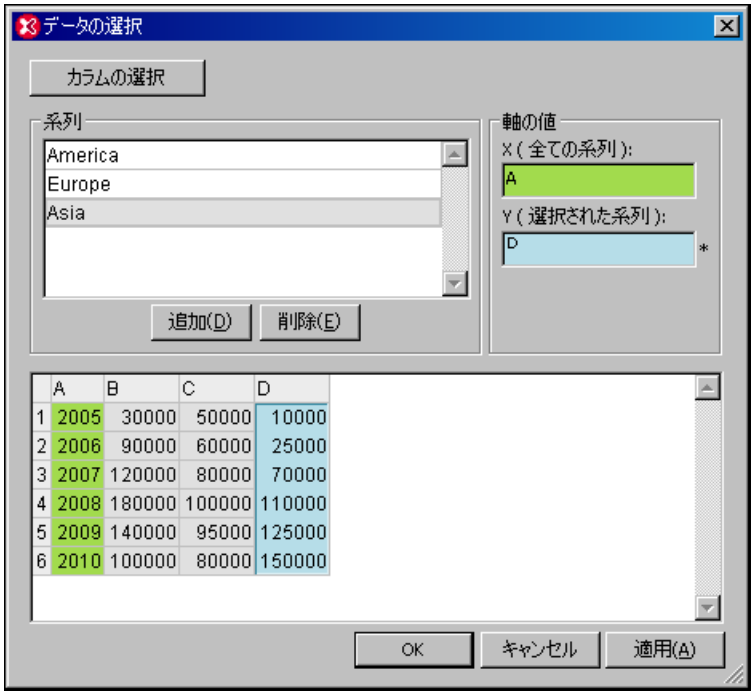
ソースXPath	X-軸	Y-軸 系列列)		
`Region[1]/Year[1]`	`@id`	`text()`	`XPath-1`	`XPath-2`
`Region[1]/Year[2]`	`@id`	`text()`	`XPath-1`	`XPath-2`
`Region[1]/Year[3]`	`@id`	`text()`	`XPath-1`	`XPath-2`
`Region[1]/Year[4]`	`@id`	`text()`	`XPath-1`	`XPath-2`
`Region[1]/Year[5]`	`@id`	`text()`	`XPath-1`	`XPath-2`
`Region[1]/Year[6]`	`@id`	`text()`	`XPath-1`	`XPath-2`

コンテキストノードが、6つある`Region[1]/Year` 要素の各ノードになっていることに注目してください。(スクリーンショットに表示されている)最初のXPath 条件式は、現在の`Year/@id` 属性値を探し、現在の`Region[1]/Year` の`Year@id` と同じ`Year/@id` 値を持つ`Region[2]/Year` 要素を返します。2つめのXPath 条件式は同じことを`Region[3]/Year` 要素に対して行ないます。この方法を使えば、6つある各年に対して、Y-軸の系列は、各`Region` 要素の`Year` 子要素となります。text() ノードを使うことで、`Region[1]/Year` 要素のコンテンツが返されます。

データの選択ダイアログのチャートデータテーブルは以下のようになります。



データの選択ダイアログの系列の表示をXPath 条件式から、(以下のスクリーンショットにあるような) 意味の分かる凡例に変更します。軸の値ペインを使うことで、正しいデータ列を各系列に対して割り当てることができます (Y-軸のテキストボックスをクリックして、チャートデータテーブル内にある目的の列を選択します)。



上で紹介した2つの方法により、全く同じチャートが作成されます。ここでは2つの異なるアプローチを紹介することで、データ選択パラメーターの使用方法について説明しました。

3.9.12 チャートの例 :ローソク足チャート

ローソク足チャートは、通常株式市場における株価の動きを表すために使用されます。ローソク足チャートには 2 つの種類があります：

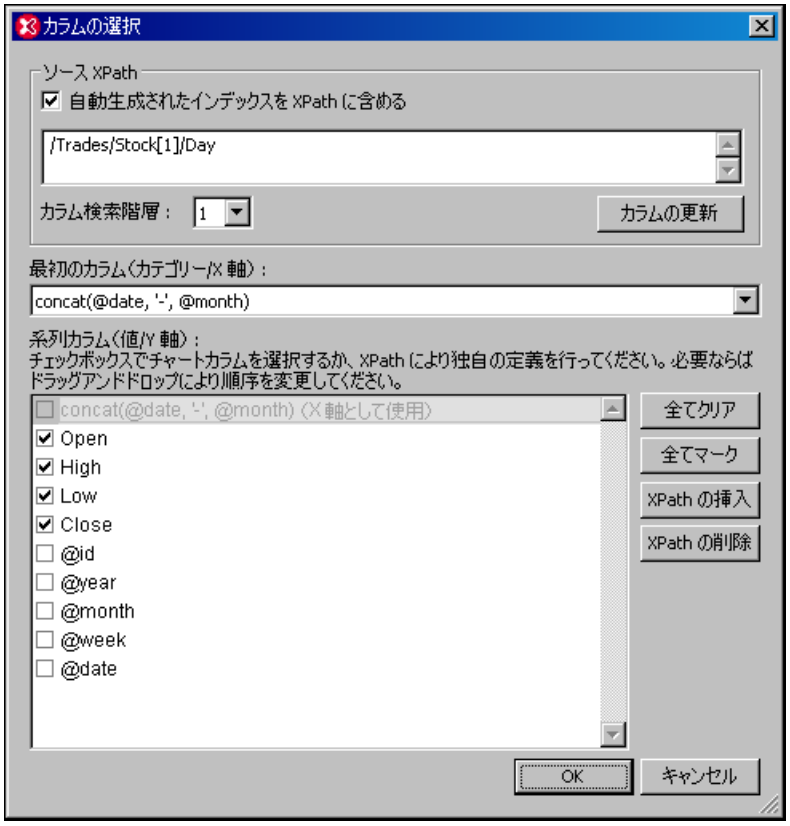
- 4 つの系列を持つローソク足チャート：1 日の始値、高値、低値、終値が表示されます。
- 3 つの系列を持つローソク足チャート：1 日の高値、低値、そして終値が表示されます。

フォルダー `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\Tutorial` 内にある Candlestick.xml という名前のサンプル XML ドキュメントを使用して、ローソク足チャートを実際に作成してみましょう。以下がファイルの基本的な構造になります。

```
<Trades>
 <Stock name="MyStock">
  <Day id="20110103" year="2011" month="Jan" week="01" date="03">
   <Open>90</Open>
   <High>110</High>
   <Low>88</Low>
   <Close>105</Close>
  </Day>

  .....

 </Stock>
</Trades>
```

以下の操作を行うことで、このファイルからローソク足チャートを作成することができます：

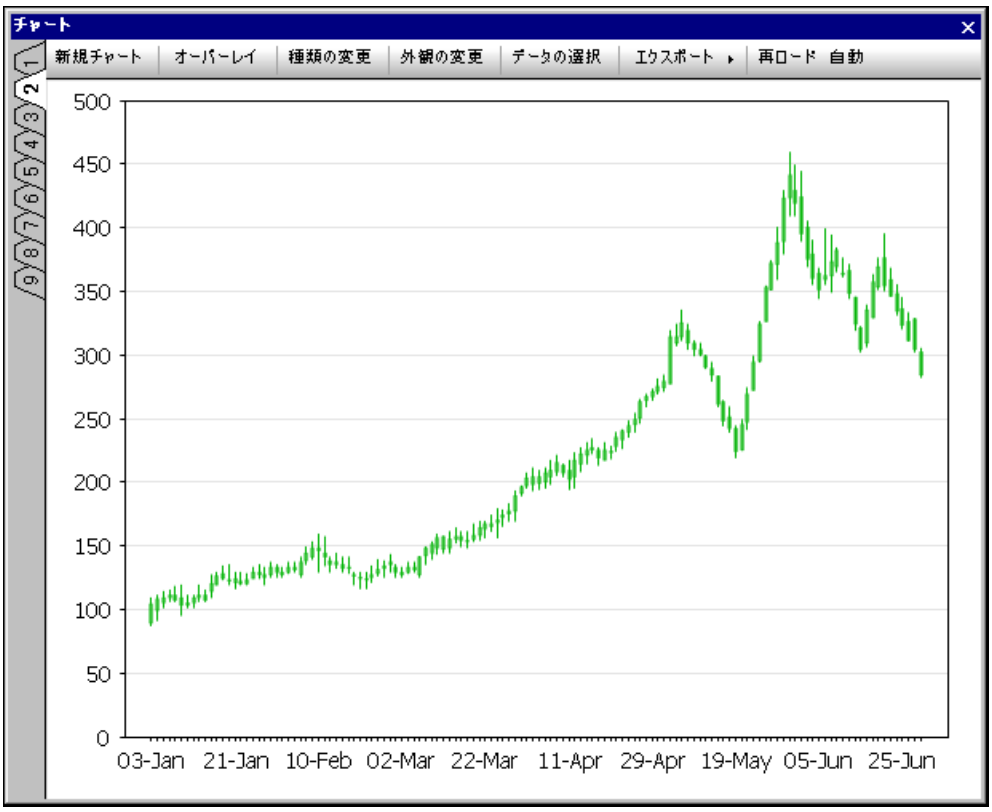
1. カーソルを Day 要素内部に移動し、チャートウィンドウの **新規チャート** ボタンを押下します。列の選択ダイアログが表示されます（以下のスクリーンショットを参照）。インデックスを含めるというチェックボックスが無効になっている場合、そのチェックボックスを有効にして、**列の更新** をクリックします。



2. XPath の挿入ボタンをクリックして、以下の XPath 条件式を入力します：`concat(@daet, '-',`

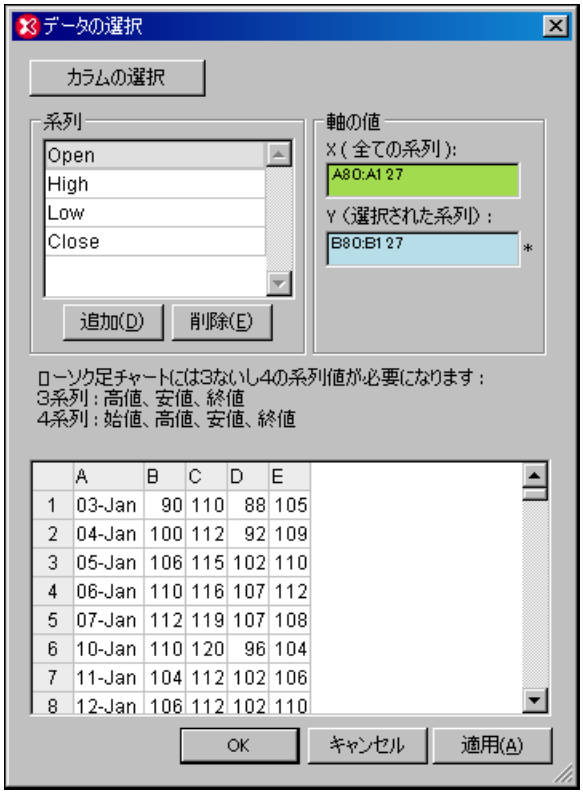
@month)

3. 最初の列トコンボボックスのドロップダウンリストから、先ほど入力したXPath 条件式を選択します。これで各 Day 要素の日付と月がX-軸要素のラベルとして使用されます。
4. Y-軸の系列に対して、系列列ペインからOpen、High、Low、Close を選択します。
5. **「OK」**をクリックします。
6. チャートウィンドウにある **「種類の変更」**ボタンをクリックし、ローソク足チャートを選択します。
7. **「OK」**をクリックします。これで以下のスクリーンショットにあるようなローソク足チャートが作成されます。



データサブセットの選択

列の選択ダイアログにて選択したデータのサブセットのみを使用したい場合（例えば、選択したチャートデータの中から数日分のデータだけをチャートに表示したい場合）、チャートウィンドウにある **「データの選択」**ボタンをクリックして、データの選択ダイアログを開きます（以下のスクリーンショットを参照）。



軸の値ペインにて、X-軸の範囲を入力します。例えば、上のスクリーンショットにあるように、`A80:A127` と入力します。その他の系列では、Y-軸のテキストボックスをクリックした後、系列ペインから目的の系列を選択し、その系列の範囲を入力します。4つある系列に対して同様の操作を行います。例えば、上のスクリーンショットでは、Open 系列における Y-軸の範囲が選択されています。データの選択ダイアログに関するより詳細な情報については、チャートのデータのセクションを参照ください。

3.10 XML 署名

XML ファイルにはデジタル署名を施して、署名の検証を行うことができます。署名が行われた後にファイルが変更された場合、検証に失敗することになります。XMLSpy ではXML 署名の作成ならびに検証がサポートされます。

XMLSpy ビューにおけるXML 署名

XML 署名はXML スキーマやWSDL、XBRL を含む全種類のXML ファイルに対して作成することができます。**「XML | XML 署名の作成」**ならびに**「XML | XML 署名の検証」**コマンドはXMLSpy のテキストビュー、グリッドビュー、スキーマビュー、WSDL ビュー、そしてXBRL ビューにて利用することができます。

XML 署名の動作原理

署名の作成から検証まで、以下のような流れで行われます：

1. 証明書の秘密鍵またはパスワードによりXML ファイルが署名されます。XMLSpy では、**「XML | XML 署名の作成」**コマンドにより署名を作成することができます。署名は、(i) XML ドキュメントを処理して、(ii) 証明書の秘密鍵またはパスワードを処理することで作成されます。
2. 署名はXML ファイル内部に保管できるほか、独立したファイルとしても保管することができます。
3. XML ファイルの署名は、証明書の公開鍵またはパスワードにより行われます。検証は、(i) XML ドキュメントを処理、(ii) 署名に使用された証明書の公開鍵またはパスワードを処理して、その結果を署名と比較することで行われます。署名が行われた後にXML ファイルが変更されると、検証に失敗することになります。XMLSpy では**「XML | XML 署名の検証」**コマンドにより署名の検証を行うことができます。

XMLSpy にて署名の作成ならびに検証を行う方法については、このセクション以下にあるサブセクションを参照ください：

- XML 署名の作成
- XML 署名の検証

XML 署名にて証明書を使用する方法

証明書をXML 署名に使用するには、証明書に対応した秘密鍵ならびに公開鍵が必要になります。秘密鍵はXML 署名の作成時に使用され、XML 署名の検証を行う際には公開鍵が必要になります。

通常であれば、XML ドキュメントの送信者が証明書の秘密鍵に対するアクセス権限を持っており、XML 署名が作成されます。ドキュメントの受信者は証明書の公開鍵へアクセスすることになります。公開鍵へアクセスするには2つの方法があります：(i) 送信された署名内にある公開鍵情報へアクセス、(ii) 送信者が使用した証明書の公開鍵へアクセスする。

証明書に関する詳細については、証明書を使用するセクションを参照ください。

メモ XMLSpy のXML 署名機能では、RSA-SHA1 ならびにDSA-SHA1 の証明書のみがサポートされます。

XML ドキュメントの確認

XML ドキュメント内にXML 署名が埋めこまれている場合、`http://www.w3.org/2000/09/xmlsig#` 名前空間にある`Signature` 要素がXML ドキュメントに追加されます。スキーマに従う形でドキュメントの妥当性を保つためには、適切な要素宣言をスキーマに加える必要があります。XMLSpy では 2 つの方法で署名を埋め込むことができます：

- *Enveloped* ：ルート要素（またはドキュメント要素）最後の子要素として、`Signature` 要素が作成されます。
- *Enveloping* ：`Signature` 要素がルート要素（またはドキュメント要素）として作成され、オリジナルのXML ドキュメント要素が、`Object` という名前でSignature 要素の子要素として埋めこまれます。

XML ドキュメントのスキーマを修正したく無い場合、XML 署名を外部ファイルとして作成することができます。詳細については、XML 署名の作成セクションにある配置に関する記述を参照ください。

Enveloped された署名の Signature 要素が許されるような XML スキーマの抜粋を以下に示します。以下にあるサンプルをガイドとして使用しながら、スキーマの修正を行ってください。

以下にあるうちの最初の例では、XML 署名スキーマがユーザーのスキーマにインポートされます。XML 署名スキーマは http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd というWeb アドレスに配置されています。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xsig="http://www.w3.org/2000/09/xmldsig#"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">
  <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
             schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
  <xs:element name="Root">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="FirstChildOfRoot"/>
        <xs:element ref="SecondChildOfRoot" minOccurs="0"/>
        <xs:element ref="ThirdChildOfRoot" minOccurs="0"/>
        <xs:element ref="xsig:Signature" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  ...
</xs:schema>
```

以下にある2つ目のサンプルでは、他の名前空間にある全ての要素にマッチする一般的なワイルドカード要素を追加します。processContents 要素を lax にセットすることで、バリデーターの検証処理でその要素の検証がスキップされます（マッチする要素宣言が見つからないため）。結果として、ユーザーは XML 署名スキーマを参照する必要がなくなります。しかしこの方法を使用すると、（Signature 要素に限らない）全ての要素を XML ドキュメントの特定位置に加えることが可能になってしまいます。

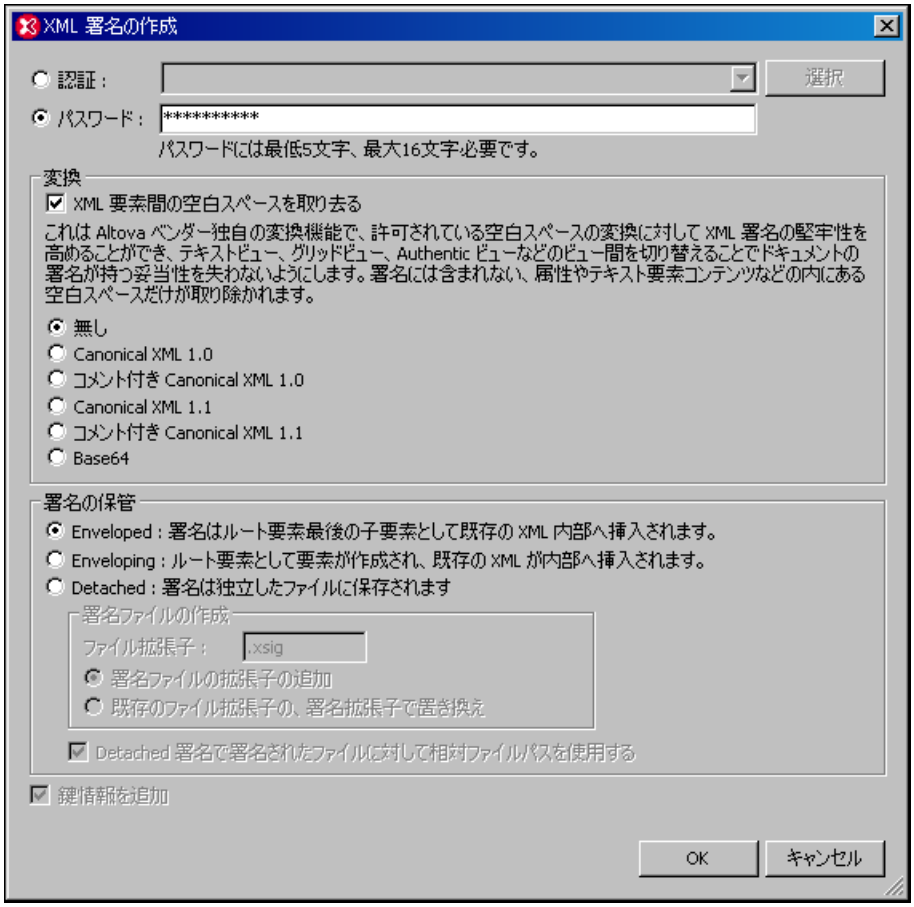
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">
  <xs:element name="Root">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="selection"/>
        <xs:element ref="newsitems" minOccurs="0"/>
        <xs:element ref="team" minOccurs="0"/>
        <xs:any namespace="##other" minOccurs="0" processContents="lax"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  ...
</xs:schema>
```

W3C 仕様

XML 署名に関する詳細については、W3C の XML 署名に関する仕様を http://www.w3.org/TR/xmldsig-core/ から参照ください。

3.10.1 XML 署名の作成

XML ドキュメントに対して XML 署名を作成するには、署名を行いたい XML ドキュメントを開きます。その後、**「XML | XML 署名の作成」**コマンドをクリックします。この操作により XML 署名の作成ダイアログが表示されます（以下のスクリーンショットを参照）。



確認方法 認証またはパスワード

署名は証明書またはパスワードを使うことで行うことができます。使用するラジオボタンを選択してください。

- *認証*：選択ボタンをクリックして、目的の証明書を選択してください。選択した証明書には秘密鍵が含まれている必要があります。署名はこの証明書の秘密鍵により生成されます。署名の検証を行うには、その証明書（または、その公開鍵）へアクセスする必要があります。証明書の公開鍵が署名の検証に使用されます。証明書に関する詳細は、証明書を使用するセクションを参照ください。
- *パスワード*：5文字から16文字の長さのパスワードを入力してください。このパスワードは、署名の検証を行う際に必要になります。

変換

XML データ変換ならびに変換の結果が署名の作成に使用されます。署名の計算を行う前に、正規化アルゴリズムを XML データ（`SignedInfo` コンテンツ）のファイルへ適用することができます。以下にアルゴリズムの重要な特徴を記します：

- *（コメント付き）Canonical XML*：署名の計算にコメントが含まれる場合、XML 内にあるコメントに対して変更が加えられることで検証が失敗するようになります。署名の計算にコメントを含めない場合、XML ドキュメント内にコメントを加えたり、既にあるコメントを修正しても、署名の検証結果に影響は与えません。

- *Base64* : XML ドキュメントのルート要素 (またはドキュメント要素) がBase64 にエンコードされているものとして、バイナリからの読み込みを試みます。ルート要素がBase64 でエンコードされていない場合、エラーが返されるか、要素が空として認識されます。
- *無し* : 変換を行うことはなく、ディスクに保存されているバイナリファイルから得られたXML データが署名の作成に直接使用されます。署名作成後にデータの編集が行われると、署名の検証が失敗するようになります。*空白スペースを取り去る* チェックボックスが有効になっている場合、全ての空白スペースが取り去られるため、空白スペースに対する変更は無視されます。*無し* オプションと *Canonical* オプションの違いは、正規化によりXML データストリームが生成され、属性の順序などが正規化されるというものです。結果として正規化による変換を行った際には属性の順序といった変更が正規化され (つまり検証には成功し)、変換を伴わない場合、それらの変更が反映されることになります (つまり検証は失敗します)。署名が (Enveloped または Enveloping により) 埋め込まれている場合、デフォルトで正規化が行われます。署名が Detached で作成された場合、*無し* が選択された場合、または *空白スペースを取り去る* チェックボックスにチェックが入っていない場合、XML データはそのままのかたちで (つまり変換無しで) 使用されます。

証明の保管

署名はXML ファイル内部に配置することができるほか、独立したファイルとしても作成することができます。以下のオプションが利用可能です：

- *Enveloped* : 署名要素がルート要素 (ドキュメント要素) 最後の子要素として作成されます。
- *Enveloping* : 署名要素がルート要素 (ドキュメント要素) として作成され、XML ドキュメントが子要素として挿入されます。
- *Detached* : XML 署名が独立したファイルとして作成されます。この場合、署名ファイルのファイル拡張子を指定することができるほか、(i) XML ファイルに拡張子を追加するかたちで (例：`test.xml.xsig`)、または (ii) XML ファイルのXML 拡張子を置き換えることで (例：`test.xsig`) ファイル拡張子の形式を指定することができます。署名ファイルにおけるXML ファイルへのリファレンスを相対または絶対パスで行うか指定することもできます。

メモ XML スキーマ (`.xsd`) ファイルに対しては、外部署名ファイルとしてしかXML 署名を作成することはできません (埋め込むことはできません)。XBRL ファイルに対するXML 署名は、外部署名としてXBRL ビューから作成することができます。WSDL ファイルの場合、署名ファイルを外部ファイルとして作成するか、Enveloped として WSDL ファイル内に含めることができます。

メモ XML 署名がDetached (独立したファイル) として作成された場合、署名ファイルにあるリファレンスによりXML ファイルと署名ファイルの関連付けが行われます。従って署名の検証を行う際には、署名ファイルがアクティブになっている状態で行う必要があります (XML ファイルがアクティブになっている状態で行うことはできません)。

鍵情報の追加

鍵情報を追加 オプションは、証明書をベースに署名を作成する際に有効になるオプションで、パスワードによる署名では無効になっています。

このオプションが有効になっていると、公開鍵情報が署名内部に配置されます。オプションが選択されていない場合、鍵情報が署名に加えられることはありません。鍵情報を追加することで、検証を行う際に、証明書そのもの (証明書内部にある公開鍵情報) が必要なくなります (公開鍵情報が既に署名に含まれているため)。

3.10.2 XML 署名の検証

XML 署名の作成後にXML ファイルが変更されなかった場合に限りXML 署名の検証は成功し、そうでない場合、検証は失敗します。XMLSpy では以下に記される条件が満たされる場合、XML 署名の検証を行うことができます：

- XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれている
- XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれていない
- 証明書をベースにした署名は外部ファイルとして保存されており、鍵情報が署名に含まれている
- 証明書をベースにした署名は外部ファイルとして保存されており、鍵情報が署名に含まれていない

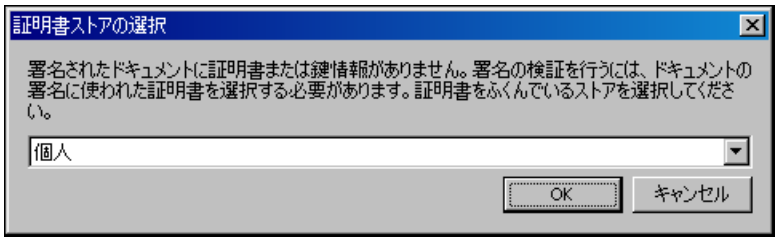
- パスワードをベースにした署名がXML ファイルに含まれている
- パスワードをベースにした署名が独立したファイルに存在する

XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれている

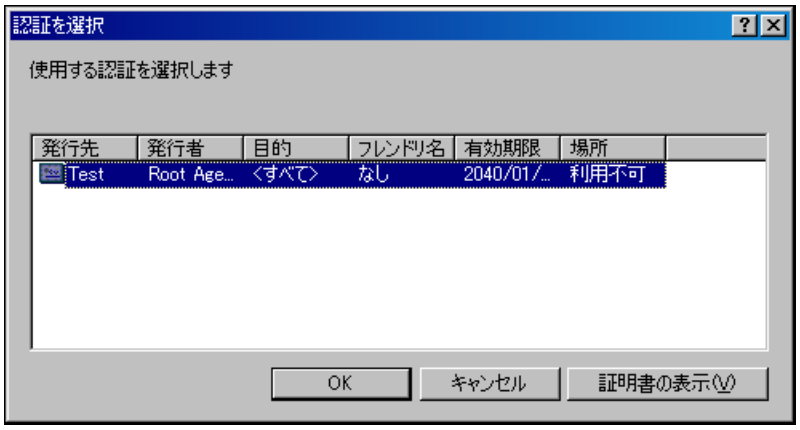
XML 署名の検証を行うには、XMLSpy にてXML ファイルをアクティブにします。**「XML | XML 署名の検証」**コマンドをクリックすると、検証処理が実行され、検証が成功または失敗したという結果がメッセージウィンドウに表示されます。

XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれていない

証明書をベースにした署名に鍵情報が含まれていない場合、XMLSpy により検証に使用する公開鍵が収められている証明書を指定するよう促されます。XMLSpy にてXML ファイルがアクティブになっている状態で検証が実行されます。**「XML | XML 署名の検証」**コマンドをクリックすると、証明書が収められている証明書ストアを選択するよう促されます（以下のスクリーンショットを参照）



証明書ストアを選択し、**「OK」**ボタンをクリックすると、そのストア内にある証明書を表示するダイアログが表示されます（以下のスクリーンショットを参照）。検証に必要な証明書を選択し、**「OK」**をクリックします。



検証処理が実行され、結果がメッセージウィンドウに表示されます。

証明書をベースにした署名は外部ファイルとして保存されており、鍵情報が署名に含まれている

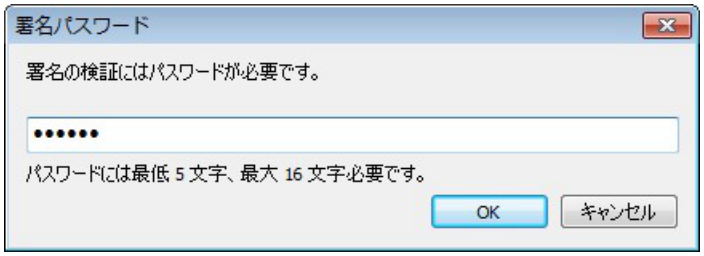
証明書をベースにしたXML 署名が独立したファイルとして存在する場合、署名ファイルがXMLSpy にてアクティブになっている状態で署名の検証が行われます。**「XML | XML 署名の検証」**コマンドをクリックすると、検証処理が実行され、その結果（検証が成功したか失敗したか）がメッセージウィンドウにて表示されます。

証明書をベースにした署名は外部ファイルとして保存されており、鍵情報が署名に含まれていない

証明書をベースにしたXML 署名が独立したファイルとして存在する場合、署名ファイルがXMLSpy にてアクティブになっている状態で署名の検証が行われます。**「XML | XML 署名の検証」**コマンドをクリックすると、検証に必要な公開鍵情報が得られる証明書を指定するようXMLSpy により促されます。XML ファイルに証明書をベースにした署名が含まれており 鍵情報が署名に含まれていないの記述に従い、証明書を選択してください。

パスワードをベースにした署名が XML ファイルに含まれている

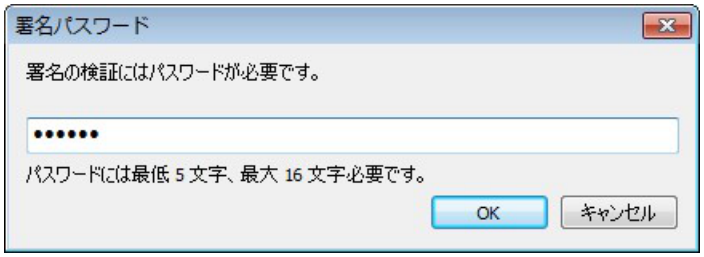
XML ファイル内にパスワードをベースにしたXML 署名が含まれている場合、XMLSpy ではそのXML ファイルがアクティブになっている状態で検証を行うことができます。**「XML | XML 署名の検証」**コマンドをクリックすると、パスワードを入力するためのダイアログが表示されます（以下のスクリーンショットを参照）。



5文字から16文字のパスワードを入力し、**「OK」**をクリックします。検証処理が実行され、その結果（検証が成功したか失敗したか）がメッセージウィンドウにて表示されます。

パスワードをベースにした署名が独立したファイルに存在する

独立したファイルにパスワードをベースにした署名が収められている場合、XMLSpy では署名ファイルがアクティブになっている状態で検証を行うことができます。**「XML | XML 署名の検証」**コマンドをクリックすると、パスワードを入力するためのダイアログが表示されます（以下のスクリーンショットを参照）。



5文字から16文字のパスワードを入力し、**「OK」**をクリックします。検証処理が実行され、その結果（検証が成功したか失敗したか）がメッセージウィンドウにて表示されます。

3.10.3 証明書を使用する

証明書はXML 署名の作成や検証に広く使用されています。このセクションでは、証明書の入手、インポート、エクスポート方法について記述します。以下のような構成となります：

- 秘密鍵 公開鍵のペアとともに証明書を入手する
- 秘密鍵 公開鍵証明書をインポートする
- Windows マシン上の証明書ストア
- 公開鍵証明書をエクスポートする

秘密鍵 公開鍵のペアとともに証明書を入手する

以下の方法により証明書を入手することができます：

- **認証局（CA）**：認証局により証明書の所有者のアイデンティティーが確認されます。この方法により入手した証明書は、誰にでも作成することができる独自の証明書と異なるものです。
- **独自の証明書を作成**：認証局により確認されていない証明書ですが、それなりの安全性も提供されます。Microsoft Visual Studio といったツールを使用することで、証明書を作成することができます。

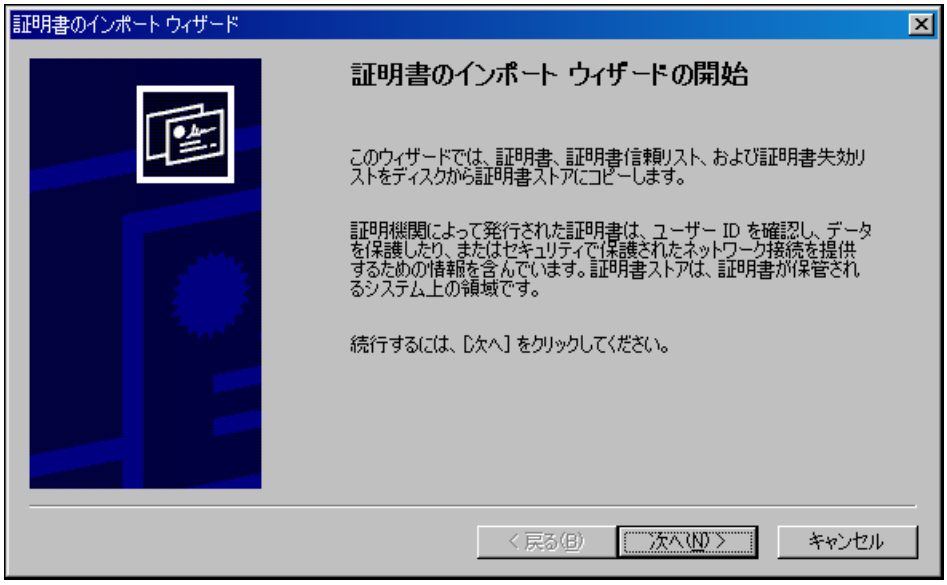
XML 署名を行うには、秘密鍵と公開鍵ペアを含む証明書を作成する必要があります。

メモ XMLSpy の XML 署名機能では、RSA-SHA1 ならびに DSA-SHA1 の証明書のみがサポートされます。

秘密鍵 公開鍵証明書をインポートする

秘密鍵と公開鍵を含む証明書を入手できたら、Windows 証明書ストアにインポートする必要があります。以下の操作を行ってください：

1. 証明書ファイルをダブルクリックして、証明書のインポートウィザードを起動します（以下のスクリーンショットを参照）。**「次へ」**をクリックします。

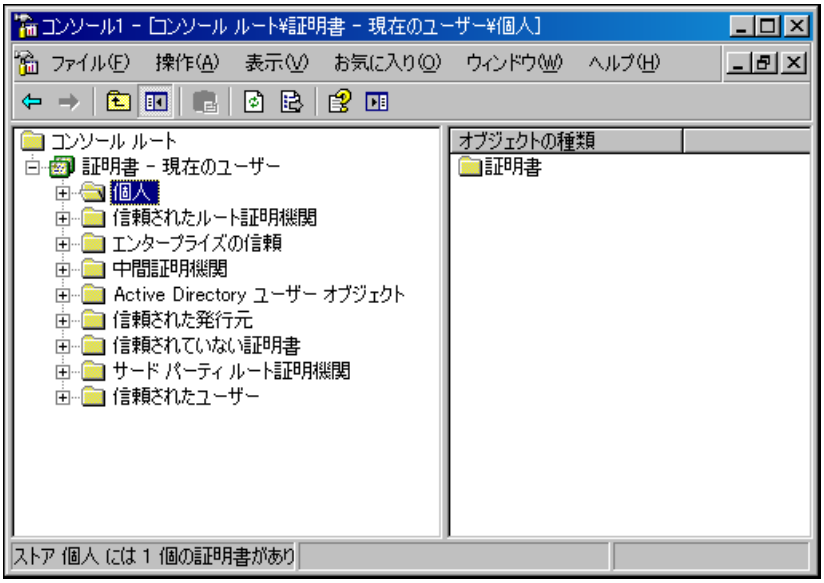


2. インポートする証明書ファイルにて目的の証明書ファイルが選択されていることを確認して、**「次へ」**をクリックします。
3. 秘密鍵のパスワードを入力します。XML 署名の作成に秘密鍵を使用するには、パスワードを知っている必要があります。秘密鍵のパスワードは、証明書を作成した際に入力したものを使用してください。パスワードを入力した後に、**「次へ」**をクリックしてください。
4. 証明書の種類により、証明書が配置される証明書ストアを自動的に選択させることができるほか、ストアを自分で選択することもできます（証明書が配置されているストアが分かっているという点で、自分でストアを選択したほうが良いかも知れません）。完了後に**「次へ」**をクリックしてください。
5. **「完了」**をクリックして、処理を完了します。

Windows マシン上の証明書ストア

以下の操作により Windows XP マシン上にある証明書ストアへアクセスすることができます：

1. スタートメニューから**「ファイル名を指定して実行」**を選択します。
2. `mmc` と入力して、**「OK」**をクリックします。コンソールウィンドウが表示されます（以下のスクリーンショットを参照）



3. コンソールウィンドウにて、**「ファイル | スナップインの追加と削除」**コマンドを選択します。
4. スナップインの追加と削除ダイアログのスタンドアロンタブにて、**「追加」**ボタンをクリックします。
5. スタンドアロンスナップインの追加ダイアログが表示されるので、証明書を選択して**「追加」**をクリックします。
6. スタンドアロンスナップインの追加ダイアログを閉じます。
7. スナップインの追加と削除ダイアログにて**「OK」**をクリックします。
8. コンソールウィンドウのコンソールルートに証明書アイテムが追加されます（上のスクリーンショットを参照）。証明書アイテムの中にはお使いのコンピューター内にある証明書ストアが含まれます。
9. コンソールウィンドウの**「ファイル | 保存」**コマンドにより、Microsoft Management Console ファイル（msc ファイル）が保存されます。このMSC ファイルを使用することで、（コンソールウィンドウの**「ファイル | 開く」**コマンドから）お使いのコンピューターにある証明書ストアへアクセスすることができます。

公開鍵証明書をエクスポートする

秘密鍵付きの証明書を使用している場合、公開鍵だけを持った証明書をエクスポートしたほうが良い場合があります。作成された公開鍵付きの証明書をXML 署名受信者へ渡すことで、秘密鍵の証明書により作成された署名の検証に使用することができます。

以下の操作を行うことにより秘密鍵と公開鍵の証明書から公開鍵の証明書をエクスポートすることができます。

1. コンソールウィンドウから証明書ストアを開きます。(i) **スタート**メニューの**「ファイル名を指定して実行」**コマンドにて mmc と入力し、(ii) 表示されるコンソールウィンドウにて**「ファイル | 開く」**を選択し、証明書ストアが保存された MSC ファイルを選択します（このセクション直前のセクションを参照ください）。
2. 公開鍵証明書としてエクスポートする証明書を選択し、右クリックします。
3. コンテキストメニューから、**「すべてのタスク | エクスポート」**を選択します。これで証明書のエクスポートウィザードが表示されます（以下のスクリーンショットを参照）。

4. **「次へ」**を選択します。
5. 秘密鍵のエクスポートウィンドウにて、*秘密鍵をエクスポートしない*を選択し、**「次へ」**をクリックします。
6. エクスポートファイルの形式ウィンドウにて、目的のフォーマットを選択し、**「次へ」**をクリックします（分からない場合は、デフォルトのDER フォーマットのままにしてください）。
7. エクスポートするファイルウィンドウにて、ファイルを保存する場所とファイルの名前を指定します（ファイルの拡張子は、自動的に追加されます）。**「次へ」**をクリックしてください。
8. **「完了」**をクリックしてエクスポートを完了します。

公開鍵を含む証明書が指定された場所にて作成されます。対応する秘密鍵の証明書により署名されたXML ファイルを受け取った相手に、公開鍵の証明書を送ることができます。証明書を受信した相手は、公開鍵の証明書を証明書ストアにインポートすることで、証明書の公開鍵を検証に使用することができます。

3.11 その他の機能

今までに紹介した以外にも、XML ファイルを使用するうえで以下の機能を利用することができます。

- エンコーディング
- DTD やスキーマの生成
- 検索と置換
- XPath の評価
- テキストのインポートとエクスポート

エンコーディング

XML ファイル (そしてその他のドキュメント) のエンコーディングはメニューコマンド **「ファイル | エンコード」** により設定することができます。XML ならびに非 XML ファイルのデフォルトエンコーディングは **「オプション | エンコード」** タブにより指定することができます。

DTD や XML スキーマの生成

「DTD/スキーマ | DTD/スキーマの生成」 メニューコマンドにより、XML ドキュメントの構造を記述するスキーマを作成することができます。表示される DTD スキーマの生成ダイアログでは、生成するスキーマの種類 (DTD もしくは XML スキーマ) や、XML ドキュメントにある値から列挙を生成するなどの XML スキーマオプションを指定することができます。

検索と置換

「編集」 メニューからアクセスすることができる検索ならびに置き換え機能によりパワフルな検索機能を手に入れることができます。検索の対象となる単語には、大文字や小文字の区別を行うかや、単語の完全マッチ、更には正規表現を使用することもできます。検索の対象はドキュメント内の選択や、以下のスクリーンショットに示されるように、特定のノード型に限定することができます。

上のスクリーンショットはテキストビューの検索と置き換えダイアログです。ダイアログとその機能は、アクティブになっているビューにより変化します。

XPath の評価

XPath ウィンドウで入力した XPath 条件式をアクティブな XML ドキュメントに対して評価することができます。評価結果は XPath ウィンドウ内に表示され、その中のノードをクリックすることで、メインウィンドウ内の対応するノードがハイライトされます。XPath ウィンドウは **「XML | XPath を評価」** コマンドによりアクティブにすることができます。

テキストのインポートとエクスポート

テキストデータは他のアプリケーションから（そして他のアプリケーションへ）インポート（ならびにエクスポート）することができます。これらの機能のコマンドは変換メニューから利用することができます。

4 DTD とXML スキーマ

Altova Web サイト: XML スキーマエディター

このセクションではDTD ならびにXML スキーマに関する概要を説明します。SchemaAgent ならびにパワフルなスキーマの内を検索する機能についても解説されます。編集機能の他に、XMLSpy ではDTD/スキーマに関する以下の機能が提供されます:

カタログメカニズム
OASIS カタログ機能によりURI からローカルアドレスのリダイレクトが可能になり、複数のワークステーションにおける利用が促進されます。

スキーマルール
ユーザー定義の追加制約をXML スキーマに対して割り当てることができます。XMLSpy にはスキーマルールエディターが搭載されており、XML スキーマに対して、スキーマルールセットを作成、編集することができます。

スキーマサブセット
スキーマビューでは、巨大なスキーマのコンポーネントを、別のファイルとして作成することができます。これら (より小さな) スキーマサブセットを、大きなスキーマにインクルードすることもできます。また、スキーマのフラット化という名前の機能を使うことで、逆の操作を行い、インクルードされているファイル内にあるコンポーネントを、直接巨大なスキーマに取り込むことができます。スキーマのサブセットと、スキーマのフラット化に関する詳細な情報は、スキーマサブセットのセクションを参照ください。

DTD とXML スキーマ間の変換
「DTD/スキーマ」 メニューを使用することで、DTD とXML スキーマ間の変換を行うことができます。使用方法などDTD がフラット化されている場合、含まれているインポートされたモジュールは親ファイルに直接保存され、使用されていないコンポーネントは削除されます。

サンプル XML ファイルの生成
「DTD/スキーマ」 「サンプルXML ファイルの生成」 メニューコマンドにより、アクティブなDTD やXML スキーマファイルをベースにしたスケルトンXML ドキュメントの生成を行うことができます。スキーマをベースにしたXML ファイルを作成するにはとても便利な機能です。

定義へ移動
カーソルがXML ドキュメントのノード内に位置している場合、**「DTD/スキーマ」 「定義へ移動」** メニューコマンドにより、スキーマファイルが開かれ、選択されているXML ノードの定義を行っている箇所がハイライトされます。

4.1 DTD

DTD ドキュメントはテキストビューならびにグリッドビューにて編集することができます。デフォルトで開かれるビューはオプションダイアログのファイルタイプタブから設定することができます。

テキストビュー

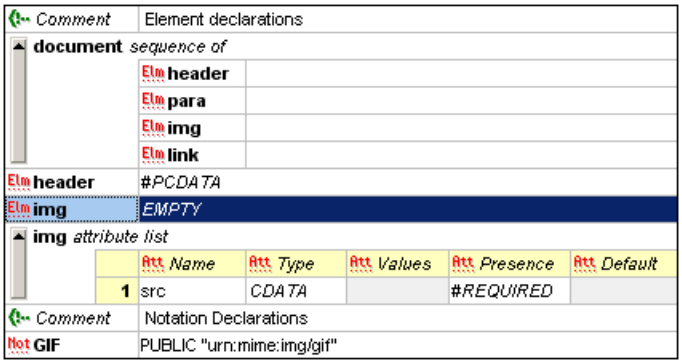
テキストビューでは、ドキュメントは構文により色分けされています。以下はDTD の断片サンプルです。

```
<!-- Element declaration -->
<!ELEMENT document (header, para, img, link)>
<!ELEMENT header (#PCDATA)>
<!ELEMENT img EMPTY>
  <!ATTLIST img
    src CDATA #REQUIRED
  >

<!-- Notation Declarations -->
<!NOTATION GIF PUBLIC "urn:mime:img/gif">
```

インデントはインデントガイドにより表され、タブキーにより入力することができます。インデントの大きさはテキストビュー設定ダイアログにてセットすることができます。

グリッドビュー

グリッドビューにおいて、DTD ドキュメントはテーブルとして表示されます。以下のスクリーンショットでは、上にあるDTD をグリッドビューで表示したものです。



カーソルがテーブルの行の中にある場合、または行が選択されている場合、**XML** メニューのDTD 編集コマンドが選択可能になります。グラフィカルに表示されるDTD にてノードの挿入や追加、または子ノードの追加を行うことができます。特定の選択箇所で利用することのできるDTD アイテムは、**XML** メニューのサブメニュー **挿入**、**追加**、**子を追加**) にて選択可能になります。選択されたDTD アイテムを別のアイテムに変換したり、アイテムを左右に移動することで、ドキュメント構造内の位置を変更することができます。ノードが選択されている場合、使用可能なDTD アイテムが入力ヘルパーにも表示されます。

XMLSpy のDTD 機能

XMLSpy には以下の非常に便利な機能が含まれています：

- *DTD をXML スキーマへ変換*：**DTD/スキーマ| DTD/スキーマの変換|** コマンドにより、DTD をXML スキーマへ変換することができます。
- *DTD からサンプルXML ファイルを生成*：**DTD/スキーマ| サンプルXML ファイルの生成|** コマンドにより、アクティブなDTD をベースにしたXML ドキュメントを生成することができます。

4.2 XML スキーマ

XML スキーマドキュメントの編集はテキストビュー、グリッドビュー、そしてスキーマビューで行うことができます。XML スキーマドキュメントが開かれるデフォルトのビューは、オプションダイアログのファイルタイプタブにて設定することができます。ドキュメントの編集中でも、その時に一番使いやすいビューに切り替えて作業を行うことができます。XML スキーマドキュメントは、大抵の場合 .xsd または .xs のファイル拡張子で保存されます。

テキストビューでの編集

テキストビューの XML スキーマ編集は XML ドキュメントと同じように行われ、XML ドキュメントで使用可能な編集機能を XML スキーマでも利用することができます。XML ドキュメントと同様に、参照しているスキーマを特定し、アクセスすることができる場合、カーソルの位置に追加することができるアイテムがテキストビュー入力ヘルパーに表示されます。

グリッドビューでの編集

グリッドビューの XML スキーマ編集は XML ドキュメントと同じように行われ、XML ドキュメントで使用可能な編集機能を XML スキーマでも利用することができます。グリッドビューにてアイテムが選択されている場合、カーソルの位置に追加することのできるアイテムがグリッドビュー入力ヘルパーに表示されます。

スキーマビューでの編集

スキーマビューはスキーマのデザインをグラフィカルに行うためのビューです。スキーマビューにてスキーマの作成、編集を行うと、対応するテキストドキュメントが XMLSpy により生成されます。スキーマビューにて XML スキーマドキュメントを作成、編集する詳細な方法については編集ビュー | スキーマビューを参照ください。

XMLSpy での XML スキーマ機能

XMLSpy には更に、以下のような便利な機能が含まれます：

- *XML スキーマを DTD へ変換*：**「DTD/スキーマ | DTD/スキーマの変換」**コマンドにより、XML スキーマを DTD へ変換することができます。
- *スキーマからサンプル XML ファイルを生成*：**「DTD/スキーマ | サンプル XML ファイルの生成」**コマンドにより、アクティブな XML スキーマをベースにした XML ドキュメントを生成することができます。生成されるサンプル XML の要素ならびに属性のサンプル値も指定することができます。
- スキーマビューにて表示される XML スキーマ (.xsd) に対する XML 署名は、独立した署名ファイルとして作成することができます。署名の使用方法については XML 署名のセクションを参照ください。

4.3 スキーマサブセット

XML スキーマのコンポーネントは、スキーマサブセットと呼ばれる、(より小さな) 外部スキーマファイルとして作成することができます。小さなスキーマサブセットの集合を (インクルードと一緒に) 使用することで、巨大なスキーマを使用した際に比べて、単一の巨大なスキーマよりもスキーマの管理を行ないやすくなります。

スキーマビューにて以下のような操作を行うことで、スキーマサブセット機能が持つ機能を理解することができるはずです：

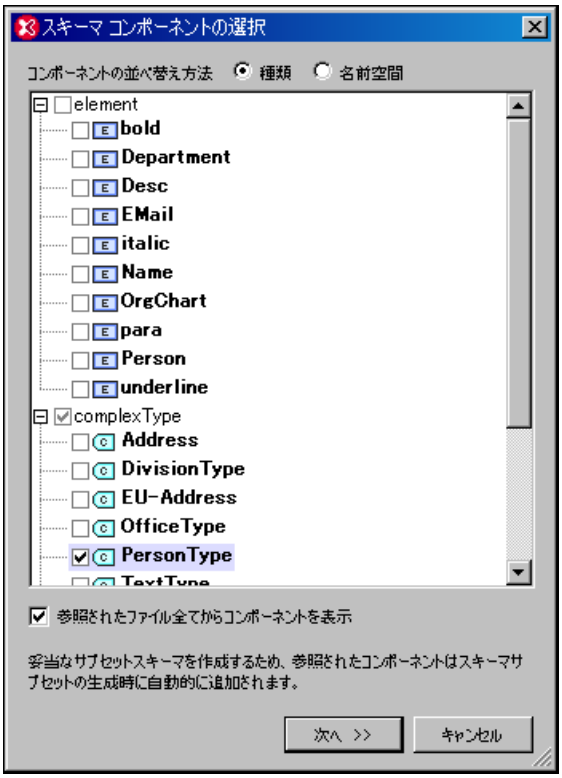
1. アクティブなスキーマのコンポーネントを含むスキーマサブセットを作成します。具体的な操作の方法については以下を参照ください。
2. 必要に応じて別のスキーマサブセットを作成します。
3. 新たに作成されたスキーマサブセットをインクルードして、大きなスキーマを形成します。スキーマ概要ウィンドウにて、インクルードコンポーネントを追加または挿入し、新たに作成したスキーマサブセットファイルを選択する操作を、各スキーマサブセットに対して行ないます。
4. オリジナルの巨大なスキーマに元々存在し、インクルードされたサブセットにも存在するコンポーネントを削除します。

スキーマビューでは、上の操作とは逆に、インクルードされたファイルをフラット化することができます。つまり、(i) スキーマサブセットに含まれているコンポーネントがメインスキーマに直接追加し、(ii) インクルードされたスキーマサブセットをメインスキーマから削除します。スキーマをフラット化する方法については以下に記述されます。

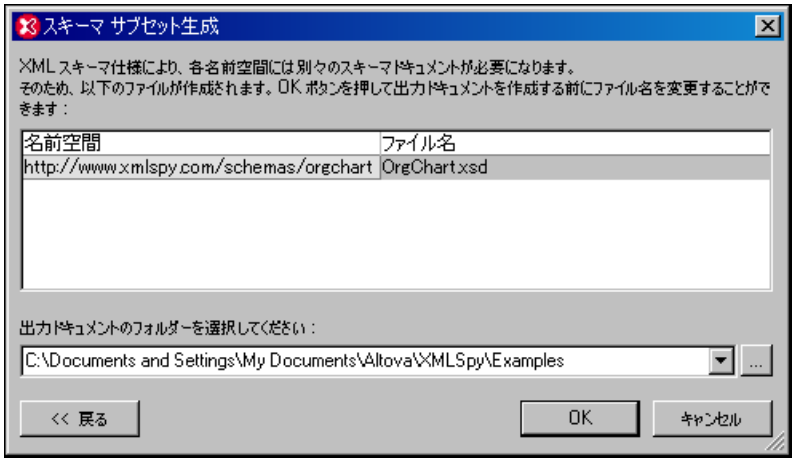
スキーマサブセットの作成

スキーマサブセットを作成するには：

1. 目的の XML スキーマがスキーマビューにてアクティブな状態で、メニューコマンドの **『スキーマ設計 | スキーマサブセットの作成』** を選択します。この操作により、スキーマコンポーネントの選択ダイアログが表示されます (以下のスクリーンショットを参照)。
2. ダイアログにて、単一のスキーマサブセットとして作成したいコンポーネントにチェックを入れ、**『次へ』** をクリックします。ペイン以下のチェックボックスにより、参照されている全てのファイルにあるコンポーネントを表示することができます。



3. 以下のスクリーンショットにある) スキーマサブセット生成ダイアログにて、スキーマサブセットパッケージの名前を入力します。同時に、新たなスキーマサブセットファイルが保存されるフォルダーも指定する必要があります。オリジナルのスキーマにてインポートされていたコンポーネントが、作成されるコンポーネントに含まれている場合、スキーマサブセットパッケージには複数のファイルが含まれることになります。スキーマサブセットでは、各名前空間に対して、異なるファイルが作成されます。デフォルトでダイアログに表示されるファイル名はオリジナルのファイル名となりますが、オリジナルのファイルを上書きすることは許されないため、オリジナルのファイルがあるフォルダーにサブセットファイルを保存したい場合は、名前の変更を行う必要があります。

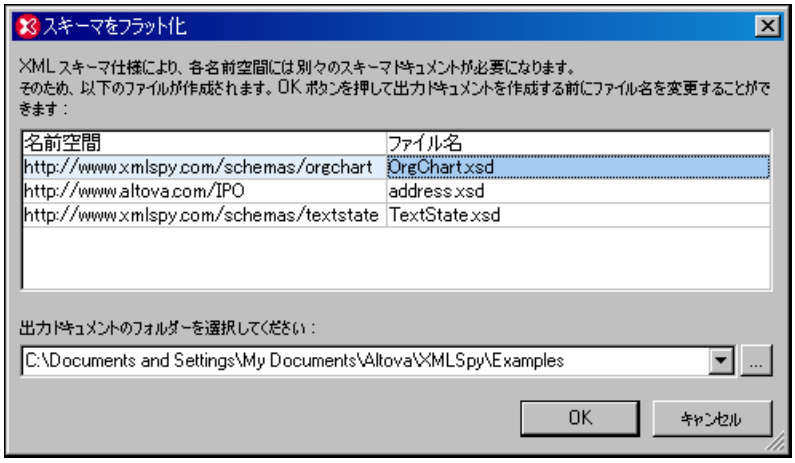


4. **「OK」**をクリックすることで、アクティブなファイルの名前空間を持ったスキーマサブセットファイルが、スキーマビューで開かれます。パッケージ内にあるその他のファイルは、作成されますが、スキーマビューでは開かれません。

スキーマのフラット化

スキーマビューにてアクティブなスキーマをフラット化すると、(i) 各スキーマのコンポーネントがアクティブなスキーマのグローバルコンポーネントとして追加され、(ii) インクルードされたスキーマが削除されます。

アクティブなスキーマをフラット化するには、メニューコマンドから **「スキーマ設計 | スキーマをフラット化」**を選択します。この操作により、以下のスクリーンショットにある) スキーマをフラット化ダイアログが表示され、各ファイルの名前が、フラット化されたスキーマで使用される名前空間と共に表示されます。デフォルトのファイル名はオリジナルのファイル名のままになっていますが、オリジナルのファイルの場所と同じフォルダーに保存しようとする場合には、ファイル名を変更する必要があります。フラット化されたスキーマと、関連するファイルを保存するフォルダーを指定することもできます。



「OK」をクリックすると、フラット化されたスキーマが、スキーマビューで開かれます。

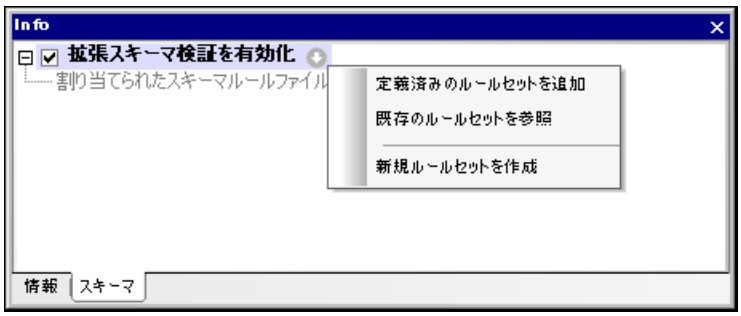
4.4 スキーマルール

スキーマルールセットとは XML スキーマの検証を行う際に使用することのできるルールセットのことです。例えばルールを使うことで、XML スキーマで定義されている様に、属性名を小文字で始まる様に指定したり、特定の型からしか複合型を作成できないように指定することができます。

スキーマルールのセットは XML (.xml) ファイルであるルールセットファイル内に保存されます。XMLSpy には[スキーマルールエディター]が搭載されており、グラフィカルにスキーマルールを編集することができます。XMLSpy では複数のルールセットファイルを[XML スキーマに割り当て]ることができます。XML スキーマビューにて **「XML | XML 検証 (F8)」** コマンドを使用すると、割り当てられたルールセットに対して XML スキーマが検証されます。

4.4.1 ルールセットの管理

アクティブな XML スキーマ (xsd ファイル) には 1 つ以上のスキーマルールセットファイル (xml ファイル) を割り当てることができます。この操作は情報ウィンドウのスキーマタブから行うことができます (以下のスクリーンショットを参照)。



ルールセットによる拡張検証

スキーマルールセットを追加するにはコンテキストメニューボタン をクリックします。(上のスクリーンショットで表示される) ポップアップメニューからルールセットファイルを XML スキーマに追加することができます。以下のオプションが利用可能です：

- **定義済みのルールセットを追加**：XMLSpy と共に提供される定義済みのスキーマルールセットのリストから選択を行うことができます。これらのルールセットファイルは XMLSpy アプリケーションフォルダー以下の Extended Schema Validation フォルダーに保存されます。このフォルダーに新たなルールセットファイルを追加することで、そのファイルをルールセットの追加時に定義済みルールセットダイアログから選択することができるようになります。
- **既存のルールセットを参照**：定義済みリストにはないスキーマルールセットファイルを参照することができます。
- **新規ルールセットを作成**：スキーマルールエディターを起動して、スキーマルールセットファイル内のスキーマルールを編集することができます。スキーマルールエディターの使用方法については、[ルールセットの定義]を参照ください。

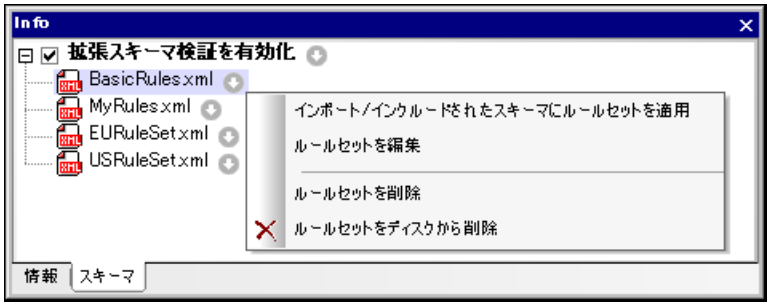
(上のスクリーンショットにあるように) 追加できるスキーマルールセットの数に上限はありません。2 つ以上のスキーマルールセットが XML スキーマに割り当てられている場合、全てのスキーマルールセットから得られたルールが XML スキーマの検証時 (**「XML | XML 検証」**) に使用されます。

拡張スキーマ検証の有効 /無効化

拡張スキーマ検証を有効化チェックボックスをクリックすることで、拡張スキーマ検証を有効化 /無効化することができます。

ルールセットの編集と削除

以下のスクリーンショットにあるように、コンテキストメニューボタン をクリックすることで表示されるコンテキストメニューから、XML スキーマに割り当てられた各ルールセットの管理を行うことができます。



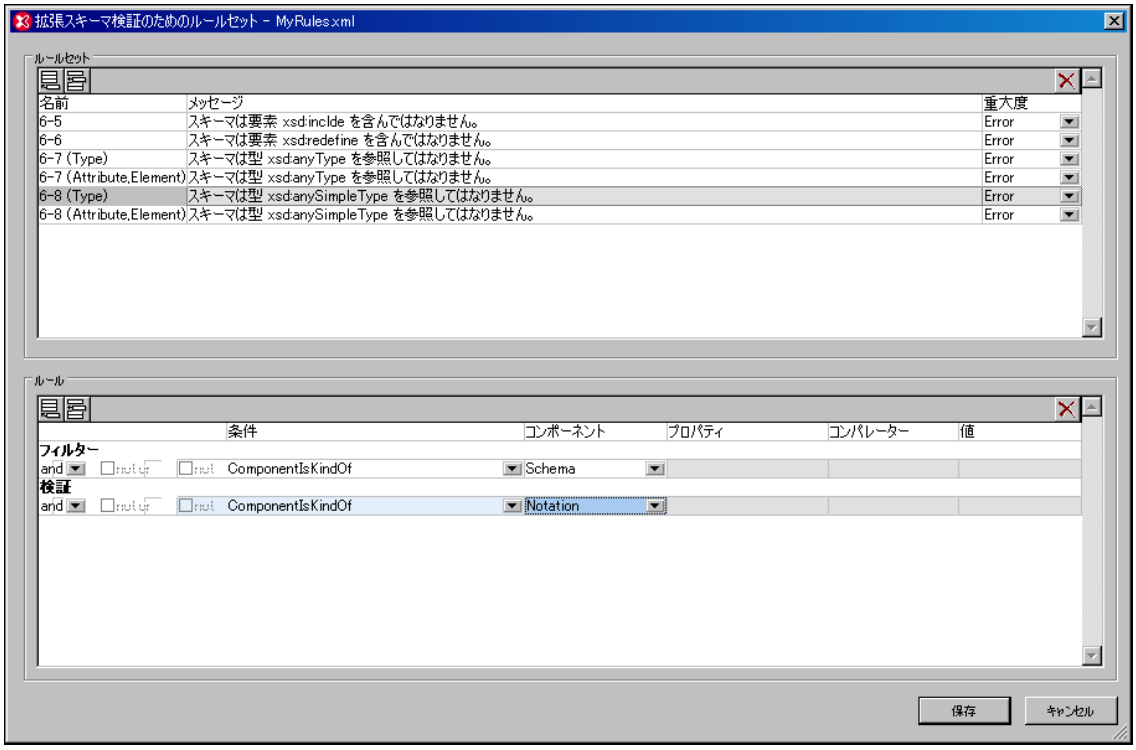
以下のオプションが利用可能です：

- **インポート/インクルードされたスキーマにルールセットを適用**：メインスキーマがインポートまたはインクルードする全てのスキーマに対してルールセットが適用されます。
- **ルールセットを編集**：スキーマルールセットをスキーマルールエディターで開きます。
- **ルールセットを削除**：ルールセットのリストからルールセットを削除します。
- **ルールセットをディスクから削除**：このコマンドは定義済みのリストにはないルールセットに対して有効になります。ルールセットのリストからルールセットを削除するだけでなく、ルールセットのファイルそのものを削除します。

4.4.2 ルールセットの定義

スキーマルールセットは、以下のスクリーンショットにあるように、スキーマルールエディターにて開き、編集することができます。スキーマルールセットファイル内にてスキーマルールを作成、編集、削除することができます。スキーマルールエディターでルールセットを開くには以下の操作を行ってください：

1. 情報ウィンドウにてルールセットをリストから選択します。
2. そのルールセットのコンテキストメニューボタン をクリックします。
3. 表示されるコンテキストメニューから、ルールセットを編集を選択します。



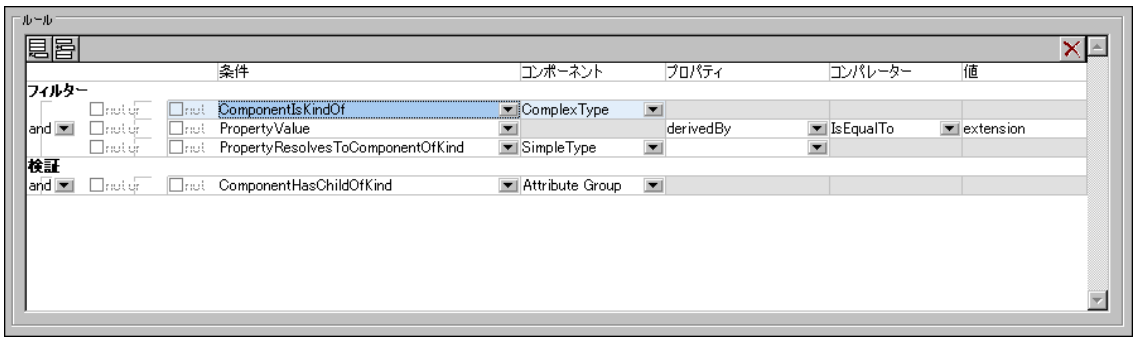
スキーマルールエディターには2つのペインがあります：

- (エディターの上部にある) ルールセットペインでは、ルールの追加や削除を行うことができます。対応する（**追加**」または **挿入**」）ボタンをクリックすることで、ルールを表す空の行が追加または挿入されます。ルールの削除を行うには、目的の行を選択した後ペインの右上にある **削除**」ボタンをクリックします。このペイン内の各ルールには名前と メッセージテキスト、そして (ルールの違反を発見した際にバリデーターがエラーまたは警告を返すかを決定する) 重大度を入力するフィールドが存在します。
- (エディターの下部にある) ルールペインには、上部のルールセットペインで選択されているルールの詳細が表示され、ルールの詳細を変更することができます。ルールの定義に関する詳細については、以下のルールの定義を参照ください。

ルールセットにあるルールの編集が完了したら、 **保存**」をクリックしてルールセットファイルのルールを保存します。

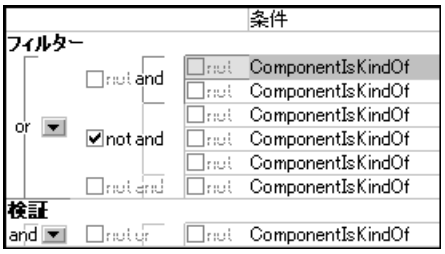
ルールの定義

ルールの定義や編集を行うには、上部のルールセットペインからルールを選択します。選択されたルールの定義がルールペインに表示され、編集を行うことができるようになります。以下のスクリーンショットに示されるルールは以下のように定義されています：*複合型が単純型の拡張であれば、Attribute Group の子を持たなければなりません。*

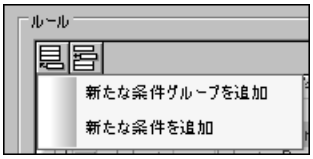


検証条件セットとフィルター条件セット：

- 各ルールには検証条件ならびにフィルター条件のセットがそれぞれ1つずつあります（上のスクリーンショットの第 1列を参照）。
- 検証条件が評価されるにはフィルター条件のセットが真（true）とならなければなりません。
- 条件の各セット（検証ならびにフィルター）は 1つ以上の条件グループにより構成され、各条件グループには 1つ以上の条件が含まれます。上のスクリーンショットの例では、1つの条件が含まれる条件グループが検証セットにあり、各自 1つの条件を持った3つの条件グループがフィルターセットにはあります。以下のスクリーンショットでは、フィルターセットに3つの条件グループが存在します：最初のグループには 2つの条件、2つ目のグループには 3つ、そして最後のグループには 1つの条件が含まれます。



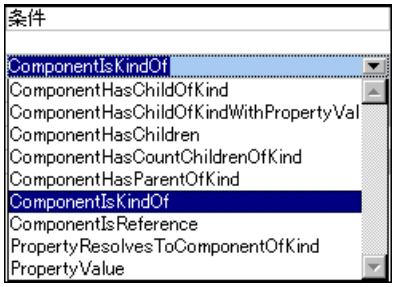
- （条件の左にある）*not* チェックボックスをクリックすることで各条件の否定を行うことができます。
- 条件グループにある論理演算子 **and** を使えば、グループ内の条件すべてが真となることで条件グループ全体が真として評価され、**or** では条件グループ内の条件が1つでも真と評価されれば条件グループ全体が真として評価されます。
- （条件グループの論理演算子の左にある）*not* チェックボックスをチェックすることで各条件グループの否定を行うことができます。
- 外側にある**and** 論理演算子を使えば（検証またはフィルター以下の）条件グループ全てが真となることで（検証またはフィルター）セット全体が真になり、**or** 論理演算子を使えば条件グループのうち1つが真であればセット全体が真となります。
- 外側にある（条件グループのための）コンボボックスの内容を変更することで論理演算子を変更することができます。内側の（条件グループ内の）論理演算子の値は外側の論理演算子とは逆のものに変更されます。
- 選択された条件を起点として条件グループまたは条件を追加、挿入することができます。条件を選択している状態で、（ペインの左上にある）**「追加」**または **「挿入」**ボタンをクリックして、メニューから目的のアイテム（条件ループまたは条件）を選択します（以下のスクリーンショットを参照）。



条件の種類

条件は以下のうちのどれかに分類されます（以下のスクリーンショットも参照）：

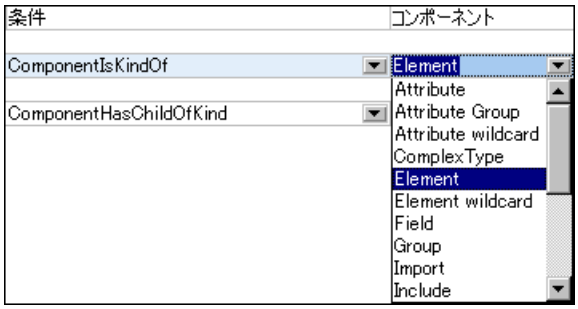
- コンポーネント（ドロップダウンリストの中で *コンポーネント*から始まるもの：以下のスクリーンショットを参照）
- プロパティ（*Property 値*）
- コンポーネントとプロパティの混合（名前に*Property* と*Component* が入っているもの）



(上のスクリーンショットにあるように) *条件*列のドロップダウンリストから条件の種類が選択されます。3つある条件の種類については以下の記述を参照ください。

コンポーネント条件

コンポーネントの条件 (*Component* で始まる条件) では、(以下のスクリーンショットにあるように) コンポーネント列にてコンポーネントを指定する必要があります。コンポーネントフィールドにあるドロップダウンリストからコンポーネントを選択します。コンポーネント条件ではその他のフィールド (プロパティやコンパレーター) の値を指定する必要が無いので、他のフィールドはグレーて表示されます。



上のスクリーンショットの例では、コンポーネントの種類が *Element* となっているものに対してフィルター条件が指定されています。例えば、これに加えて検証条件にコンポーネントはNotation という子を持たなければならないと指定されていれば、『*Element* コンポーネントは *Notation* という種類の子を持たなければならない』という条件が完成します。検証条件の `not` オプションがチェックされていれば、『*Element* コンポーネントは *Notation* という種類の子を持ってはならない』という条件になります。

プロパティ条件

プロパティの条件は (以下のスクリーンショットにあるように) *プロパティ値*で決定されます。このような条件ではプロパティの値そのものが指定されるため、プロパティならびにコンパレーター両方の列 (そして必要な場合は値列) で入力を行う必要があります。*コンポーネント*列への入力を行う必要は無いのでグレーで表示されます。プロパティ列のドロップダウンリストに表示されるアイテムには (`default` や `maxOccurs` といった) XML 属性の他にも (`derivedBy` といった) コンポーネントの論理プロパティが含まれます。

上のスクリーンショットでは *model* プロパティの値が *all* に等しいというルールが (`not` チェックボックスにより) 否定されています。*Model group* コンポーネントに対するフィルター条件と組み合わせることで、スキーマは `xsd:all` 要素を含んでいけないという意味になります。

注意：以下の点に注意してください：

- `IsQNameEqualTo` コンパレーターを使用するときは、対応する値を `{URI}localName` の形式で記述する必要があります。例えば、以下のような値になります：`{http://www.w3.org/2001/XMLSchema}NOTATION`
- `default` プロパティは空の値と共に記述 (`<element name default=""/>`)、または除外することができます (`<element name/>`)。

コンポーネントとプロパティを混合する条件

コンポーネントとプロパティの混合した条件は以下になります：

- `ComponentHasChildOfKindWithPropertyValue` ：子要素のコンポーネントの種類と、プロパティの値を指定します。
- `PropertyResolvesToComponentOfKind` ：コンポーネントの種類を解決するプロパティが指定されます。*コンパレーターと値*の列は空になります。

条件の否定

条件の左にある (内側の) *not* チェックボックスをチェックすることで、その条件は否定されます。条件グループの論理演算子の左 (外側にある) *not* チェックボックスをチェックすることで、条件グループを否定することができます。

4.5 XMLSpy のカタログ

XMLSpy は OASIS XML カタログ機能のサブセットをサポートしています。XMLSpy のカタログ機能により、広く使われているスキーマ (またはスタイルシートなどその他のファイル) をローカルユーザーフォルダーから取得することができます。これにより全体的な処理スピードが向上し、(ネットワークに接続していない状態でも) ユーザーはオフラインで作業することができるようになり、(URI の変更はカタログファイル内部に限定されるので) ドキュメントのポータビリティーが向上します。

XMLSpy のカタログ機能は以下のように動作します。

RootCatalog.xml

XMLSpy が開始されると、以下に示されるような構造を持つ (RootCatalog.xml というファイルがロードされ、ファイル内部に記述されているカタログファイルが検索されます。このファイルを修正して、検索するカタログファイルを増やすこともできます (各カタログファイルは nextCatalog 要素にて記述されます)。各カタログファイル内に記述されている URI が、ファイル内で指定されたマッピングに従って解決されます。

```
<?xml version="1.0" encoding"=UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
      xmlns:spy="http://www.altova.com/catalog_ext"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:oasis:names:tc:entity:xmlns:xml:catalog Catalog.xsd">
  <nextCatalog catalog="%PersonalFolder%/Altova/%AppAndVersionName%/CustomCatalog.xml"/>
  <nextCatalog catalog="CoreCatalog.xml"/>
  <!-- Include all catalogs under common schemas folder on the first directory level-->
  <nextCatalog spy:recurseFrom="%AltovaCommonFolder%/Schemas" catalog="catalog.xml" spy:depth="1"/
>
  <!-- Include all catalogs under common XBRL folder on the first directory level-->
  <nextCatalog spy:recurseFrom="%AltovaCommonFolder%/XBRL" catalog="catalog.xml" spy:depth="1"/>
</catalog>
```

上で示されるファイルの内容にて、%AltovaCommonFolder% という変数で指定されている Schemas ならびに XBRL フォルダー内部に catalog.xml という名前のカタログファイルがあることに注目してください (%AltovaCommonFolder% 変数の値については以下を参照ください)。

Altova 共通フォルダー以下にあるカタログファイルにより (SVG や WSDL) よく使われる定義済みのスキーマや XBRL タクソノミのパブリックならびにシステム識別子が、ローカルにコピーされた対応するスキーマを指し示す URI にマッピングされます。これらのスキーマは XMLSpy がインストールされる際に Altova 共通フォルダーにインストールされます。エラーが発生する可能性があるので、これらファイル内のマッピングを複製しないように注意してください。

CoreCatalog.xml、CustomCatalog.xml、そして Catalog.xml

上で記されている RootCatalog.xml にて、CoreCatalog.xml と CustomCatalog.xml が検索のために使用されていることに注目してください:

- CoreCatalog.xml には Altova 共通フォルダーを指し示すために使われる Altova 独自のマッピングが含まれています。
- CustomCatalog.xml はマッピングを作成するためのスケルトンファイルです。Altova 共通フォルダー以下のカタログファイルで指定されていないスキーマファイルへのマッピングを CustomCatalog.xml に追加することができます。OASIS カタログ機能によりサポートされている要素を使用してください (以下を参照)。
- Altova 共通フォルダーには数多くの Catalog.xml ファイルが格納されています。各ファイルは Altova 共通フォルダー以下のスキーマフォルダー (または XBRL タクソノミ フォルダー) に格納され、ローカルにコピーされたパブリックならびにシステム識別子のスキーマを指し示す URI のマッピングを行ないます。

カタログファイルとスキーマの場所

RootCatalog.xml ならびに CoreCatalog.xml は XMLSpy アプリケーションフォルダー以下にインストールされます。CustomCatalog.xml は MyDocuments\Altova\XMLSpy フォルダー以下に納されています。catalog.xml ファイルは各スキーマフォルダー以下にあり、これらスキーマフォルダーは %AltovaCommonFolder%

\Schema ならびに %AltovaCommonFolder%\XBRL に収められています。

シェル環境変数とAltova 変数

nextCatalog 要素ではシェル環境変数が使用され、システムに関係する場所へのパスが参照されます (上の *RootCatalog.xml* の内容を参照)。以下のシェル環境変数がサポートされます：

%AltovaCommonFolder%	C:\Program Files\Altova\Common2017
%DesktopFolder%	現在のユーザーのデスクトップフォルダーへのフルパス
%ProgramMenuFolder%	現在のユーザーのプログラムメニューフォルダーへのフルパス
%StartMenuFolder%	現在のユーザーのスタートメニューフォルダーへのフルパス
%StartUpFolder%	現在のユーザーのスタートアップフォルダーへのフルパス
%TemplateFolder%	現在のユーザーのテンプレートフォルダーへのフルパス
%Admin ツール Folder%	現在のユーザーの管理ツールが収められているシステムディレクトリへのへのフルパス
%AppDataFolder%	現在のユーザーのアプリケーションデータフォルダーへのフルパス
%CommonAppDataFolder%	全てのユーザーのアプリケーションデータが収められているファイルディレクトリへのフルパス
%FavoritesFolder%	現在のユーザーのお気に入りフォルダーへのフルパス
%PersonalFolder%	現在のユーザーの個人用フォルダーへのフルパス
%SendToFolder%	現在のユーザーのSentTo フォルダーへのフルパス
%FontsFolder%	システムフォントフォルダーへのフルパス
%ProgramFilesFolder%	現在のユーザーのProgram Files フォルダーへのフルパス
%CommonFilesFolder%	現在のユーザーのCommon Files フォルダーへのフルパス
%WindowsFolder%	現在のユーザーのWindows フォルダーへのフルパス
%SystemFolder%	現在のユーザーのシステムフォルダーへのフルパス
%LocalAppDataFolder%	ローカルアプリケーションのデータリポジトリとして使用されるファイルシステムディレクトリへのフルパス
%MyPicturesFolder%	マイ ピクチャフォルダーへのフルパス

カタログの動作：DTD

DTD が呼び出された際のリダイレクトにカタログは使用されます。カタログファイル内に記述されているマッピング情報を使用することで、パブリックまたはシステム識別子をローカルの URI にマッピングします。例えば XML ファイル内で宣言されている DOCTYPE 宣言が読まれたとき、カタログファイルマッピングにより PUBLIC または SYSTEM 識別子がローカルにあるリソースを参照します。

よく使われるスキーマや PUBLIC 識別子は通常定義が定まっており、ローカルにコピーされている URI をカタログファイルが指し示すだけで事足りてしまいます。XML ドキュメントが解析されると内部にある PUBLIC 識別子が読み込まれます。カタログファイル内に識別子が見つかった場合、カタログファイル内の対応する URL が検索され、その場所からスキーマが読み込まれます。例えば、以下の SVG ファイルが XMLSpy で開かれた場合：

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="20" height="20" xml:space="preserve">
  <g style="fill:red; stroke:#000000">
    <rect x="0" y="0" width="15" height="15"/>
    <rect x="5" y="5" width="15" height="15"/>
  </g>
</svg>
```

このドキュメントが読み込まれ、PUBLIC 識別子のカタログが検索されます。例えば、カタログファイルに以下のような記述があるとしましょう：

```
<catalog>
  ...
  <public publicId="-//W3C//DTD SVG 1.1//EN" uri="schemas/svg/svg11.dtd"/>
  ...
</catalog>
```

これで PUBLIC 識別子に対するマッチが見つかりました。SVG DTD は schemas/svg/svg11.dtd にリダイレクトされ（パスはカタログファイルからの相対パスになります）、このローカルファイルが DTD として使用されます。カタログ内に対応するマッピングが存在しない場合、XML ドキュメントに記述されている URL（上の例の場合だと http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd）が使用されます。

カタログの動作：スキーマ

XMLSpy では、XML スキーマへのリダイレクトにカタログを使用することもできます。XML インスタンスファイルでは、XML ドキュメントのトップレベルドキュメント要素内部の xsi:schemaLocation 属性内にてスキーマへの参照が記述されます。

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

通常、（上の例にて太字で示されている）属性値の URI には、実際に存在するスキーマの位置が記述されます。カタログによりスキーマが参照された場合、実際に存在する XML スキーマを指し示す必要は無いものの、xsi:schemaLocation 属性の妥当性を満たすために何らかの値を入力する必要があります。属性の URI 部分には、例えば hoge などの値を入力することでも、妥当性を確保することができます。xsi:schemaLocation 属性値の名前空間部分（上の例における http://www.xmlspy.com/schemas/orgchart）により、カタログ内にスキーマを配置することができます。カタログでは、以下のエントリーにより名前空間部分をベースにしたスキーマの識別が行われます。

```
<uri name="http://www.xmlspy.com/schemas/orgchart" uri="C:\MySchemas\OrgChart.xsd"/>
```

XMLSpy によりサポートされるカタログサブセット

CustomeCatalog.xml（または XMLSpy により読み込まれる他のカタログファイル）内でエントリを作成する際、以下

に示されるOASIS カタログ仕様の要素だけを使うようにしてください。以下に各要素と属性の説明を示します。更に詳しい情報については、XML カタログ仕様 (英文) を参照ください。各要素は xml:base 属性を持ち、要素のベースURI を指定できることに注意してください。

- <public publicId="リソースの PublicID" uri="ローカルファイルの URL"/>
- <system systemId="リソースの SystemID" uri="ローカルファイルの URL"/>
- <uri name="ファイル名" uri="ファイル名により識別されるファイルの URL"/>
- <rewriteURI uriStartString="書き直す URI の最初の文字列" rewritePrefix="StartString を置き換える文字列"/>
- <rewriteSystem systemIdStartString="SystemID の最初の文字列" rewritePrefix="ローカルのリソースを検索するための置き換え文字列"/>

殆どのスタイルシートの様に、パブリックな識別子が存在しない場合、system 要素を使うことでシステム識別子をURL に直接マッピングすることができます。また、uri 要素によりURI を別のURI にマッピングすることもできます。rewriteURI とrewriteSystem 要素によりURI またはシステム識別子の最初の文字列を書き換えることができます。これによりファイルパスの最初の部分を置き換えて、別のディレクトリにターゲットを変更することができます。これら要素に関する詳しい情報については、XML カタログ仕様 (英文) を参照ください。

リリース 2014以降 XMLSpy はXML カタログ仕様 (OASIS Standard V1.1, 7 October 2005) 仕様に準拠します。この仕様は (パブリックID またはシステムID を持つ)外部識別子の参照を (パブリックID またはシステムID ではないURI を持つ)URI の参照と厳密に区別します。名前空間 URI はですから? パブリックID またはシステムID ではなく? URI と見なされ、外部識別子の参照ではなくURI 参照に使用されます。ます。 バージョン 2014以前のXMLSpy バージョンでは、スキーマ名前空間 URI は、<public> マッピングを介して変換されていましたが、バージョン以降は<uri> マッピングが使用される必要があります。

```
Prior to v2014:  <public publicID="http://www.MyMapping.com/ref"
uri="file:///C:/MyDocs/Catalog/test.xsd"/>
V-2014 onwards:  <uri name="http://www.MyMapping.com/ref"  uri="file:///C:/
MyDocs/Catalog/test.xsd"/>
```

スキーマに従ったファイル拡張子とインテリジェントな編集

指定したスキーマのルールに準拠するようXMLSpy のインテリジェントな編集機能が適用された場合、カタログファイルを使うことで、ドキュメントに特定の拡張子を指定することができます。例えば独自のファイル拡張子 .myhtml を作成して、HTML DTD に従ったファイルの検証を行ないたい場合、以下の要素をCustomCatalog.xml 内の<catalog> 要素の子要素として追加することで、この拡張子を持つファイルのインテリジェント編集を行うことができます。

```
<spy:fileExtHelper ext="myhtml" uri="schemas/xhtml/xhtmll-transitional.dtd"/>
```

これによりXMLSpy 内で .myhtml ファイルを編集する際のインテリジェント編集 (自動補完、入力ヘルパーなど) がXHTML 1.0 Transition DTD に従った形で使えるようになります。似たようなエントリを含んでいる%AltovaCommonFolder%\Schemas\xhtml フォルダー以下のcatalog.xml ファイルも参照ください。

XML スキーマ仕様

XML スキーマの仕様情報はXMLSpy に収められており、XML スキーマ (.xsd) ドキュメントの妥当性は、内部にある情報をベースにチェックされます。そのため、XML スキーマ仕様を定義するスキーマに対して参照が行われることはありません。

%AltovaCommonFolder%\Schemas\schema フォルダー以下にあるcatalog.xml ファイルには、古いXML スキーマ仕様のDTD に対する参照が記述されています。このファイルから参照されるスキーマファイルを使ってXML スキーマドキュメントの検証を行うべきではありません。これらのファイルは、以前の勧告に従ったドキュメントを作成する際に、XMLSpy の入力ヘルパーにて該当する編集情報を表示させるためだけに使用されます。

更に詳しい情報

カタログに関する更に詳しい情報は、XML カタログ仕様 (英文) を参照ください。

4.6 SchemaAgent との連携

XMLSpy は Altova の SchemaAgent テクノロジーと連動するように設定することができます。

SchemaAgent テクノロジー
SchemaAgent テクノロジーにより、複数のスキーマ間のリレーションシップを構築することができます。以下のような機能が含まれます：

- (フォルダーやネットワークの) 検索パス上にある、複数のスキーマ間のリレーションシップに関する情報を保持、提供するSchemaAgent Server。
- SchemaAgent サーバーから得られるスキーマ情報を使用してサーバーに接続後、スキーマ間のリレーションシップの構築ならびにスキーマの管理 (名前の変更、移動、削除など) を行うSchemaAgent クライアント。

2種類のSchemaAgent サーバーが利用可能です：

- ネットワーク上にインストールし、利用することのできるAltova SchemaAgent Server。
- SchemaAgent クライアント製品であるAltova SchemaAgent。SchemaAgent には、より軽量なサーバーであるLocalServer が搭載されており、SchemaAgent がインストールされているマシンからのみアクセスすることができます。

XMLSpy から SchemaAgent テクノロジーを利用することで、SchemaAgent サーバーから得られるスキーマ情報を使用して、スキーマをスキーマビューにて直接編集することができます。ここでは、XMLSpy が SchemaAgent サーバーに接続し、SchemaAgent Client とのやり取りを行いながら SchemaAgent Server へリクエストを送信します。XMLSpy が SchemaAgent と連動するよう設定された場合、スキーマビューの入力ヘルパーには、現在のスキーマビューでアクティブなスキーマから得られたコンポーネントだけでなく、接続している SchemaAgent サーバーの検索パスにあるその他のスキーマから得られたコンポーネントも表示されます。この機能を使うことで、これらのコンポーネントに直接アクセスすることができます。他のスキーマに属しているコンポーネントのコンテンツモデルを、スキーマビューで確認したり、これらコンポーネントを再利用することができます。スキーマ間のリレーションシップを構築して、XMLSpy から複雑なスキーマの管理を直接行うこともできます。

SchemaAgent とSchemaAgent Server のインストール

SchemaAgent ならびに SchemaAgent Server のインストール、そしてサーバー上の検索パスの設定については、SchemaAgent ユーザーマニュアルを参照ください。

XMLSpy をSchemaAgent クライアントとして設定する
XMLSpy をSchemaAgent クライアントと連動して使用するには以下の操作を行う必要があります：

- Altova ウェブサイトからSchemaAgent をダウンロードします。SchemaAgent のLocalServer を使用することでスキーマを提供することができるようになります。LocalServer にて検索パスを設定する方法については SchemaAgent のユーザーマニュアルを参照ください。
  注意：無料トライアル期間が終了した後にSchemaAgent を使用するには有効なライセンスを購入する必要があります。Altova MissionKit 製品パッケージにもSchemaAgent とライセンスキーが含まれています (しかしながら Altova MissionKit パッケージ内に SchemaAgent Server は含まれていません)。
- 上記に加えて、必要ならばAltova ウェブサイトからネットワークベースのSchemaAgent Server をダウンロードしてください。
- SchemaAgent (そしてSchemaAgent Server の) 検索パスを定義します。検索パスの詳細な定義方法についてはSchemaAgent ユーザーマニュアルを参照ください (検索パスとはリレーションシップのマッピングに使われるスキーマが含まれるフォルダーへのパスのことです)。
- XMLSpy からSchemaAgent への接続を開始します。

重要：Altova から提供されるSchemaAgent ならびにSchemaAgent に関連する製品 (XMLSpy を含む) のバージョン 2005 リリース 3 以降の製品は、それ以前のバージョンのSchemaAgent またはSchemaAgent に関連する製

品との**互換性を持ちません**。

XMLSpy 内の SchemaAgent コマンド
XMLSpy 内の SchemaAgent 機能はスキーマビューからのみ利用することができ、スキーマ設計メニューのメニューコマンド（次のスクリーンショットを参照）ならびにスキーマビューの入力ヘルパーからアクセスすることができます。

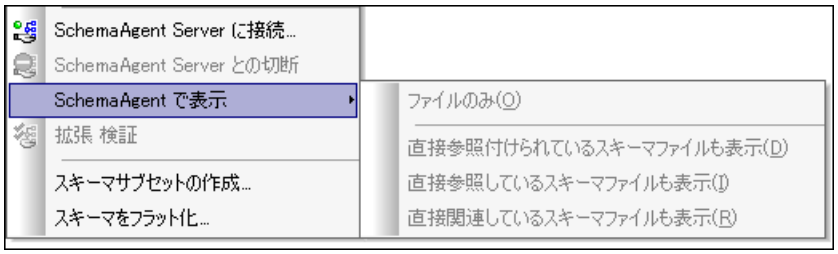
メニューコマンドでは一般的な管理機能が提供されます。入力ヘルパー（ならびにドラッグアンドドロップといった GUI を使った機能）によりスキーマの編集を実際に行うことができます。

このセクションではスキーマビューで利用可能な SchemaAgent の機能の使用方法について説明します。

4.6.1 SchemaAgent Server へ接続

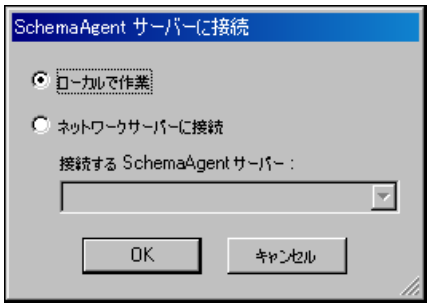
注意：接続を確立するためには SchemaAgent Client がインストールされている必要があります。

SchemaAgent Server に接続していない状態では、**スキーマ設計**メニューの**SchemaAgent Server に接続**コマンドのみが選択可能になっています。（以下のスクリーンショットにあるように）スキーマ設計メニューにあるその他の SchemaAgent コマンドは選択不可状態になっています。SchemaAgent Server への接続が確立されれば、その他のメニューアイテムも選択可能になります。



接続方法
SchemaAgent サーバーへ接続するには、以下の操作を行ってください：

1. SchemaAgent Server に接続アイコン をクリック（または**「スキーマ設計 | SchemaAgent Server に接続」**）します。（以下のスクリーンショットにある）SchemaAgent サーバーに接続ダイアログが表示されます：



2. ローカルサーバー（Altova SchemaAgent に同封されている SchemaAgent サーバー）またはネットワークサーバー（無料で利用することのできる Altova SchemaAgent Server 製品）を使用することができます。ローカルで作業を選択すると、SchemaAgent のローカルサーバーが起動され、接続が確立されます。ネットワークサーバーへ

接続を選択する場合、選択されたSchemaAgent Server が起動していなければなりません。

Windows XP SP2 上で動作しているサーバーについて
SchemaAgent Server の名前がリストに表示されているにも関わらず、接続することができない場合、サーバーがネットワークの名前解決処理を行っていない可能性があります。Windows XP SP2 ではファイヤーウォールにより名前解決をデフォルトでブロックしています。

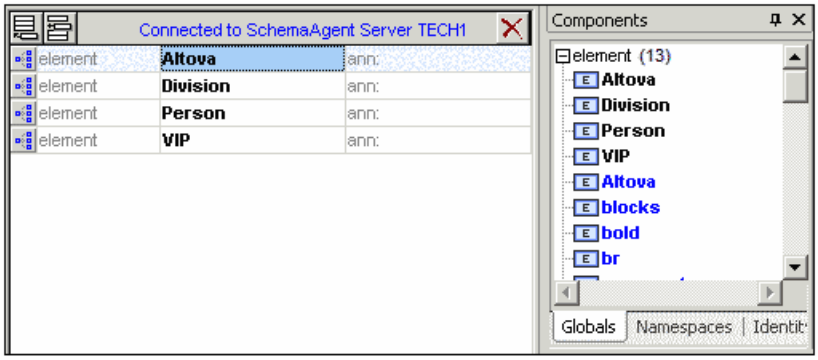
このようなサーバーに接続するには、以下のうちどれかを行ってください：

- サーバーの設定を変更して、名前解決を有効にする
- ダイアログボックスの編集フィールドにIP アドレスを入力する

SchemaAgent Client には最後に成功した接続の接続文字列が保存されるため、この操作は1度だけ行う必要があります。

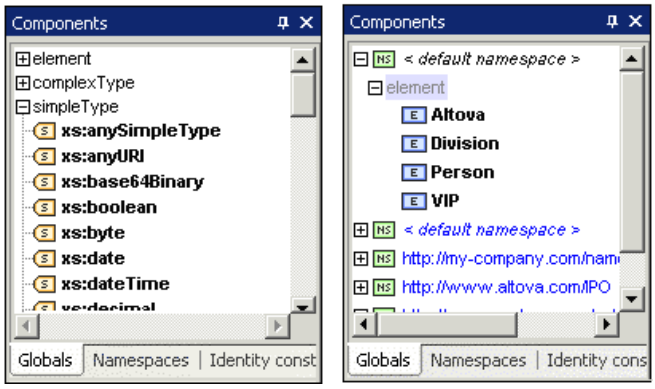
SchemaAgent サーバーに接続後のスキーマビュー

SchemaAgent サーバーへの接続が確立された後は、スキーマビューの表示は以下のようになります：



以下の点に注目してください：

- グローバルビューの上部には「SchemaAgent Server と接続」という文字列が表示され、接続を確立したサーバー名が表示されます。
- サーバー検索パス上にある全てのスキーマやスキーマコンストラクトへアクセスすることができるようになります。グローバル要素や複合型、または単純型といったSchemaAgent スキーマコンストラクトは、アクティブなスキーマの下に、**太字の青いテキスト**で表示されます。

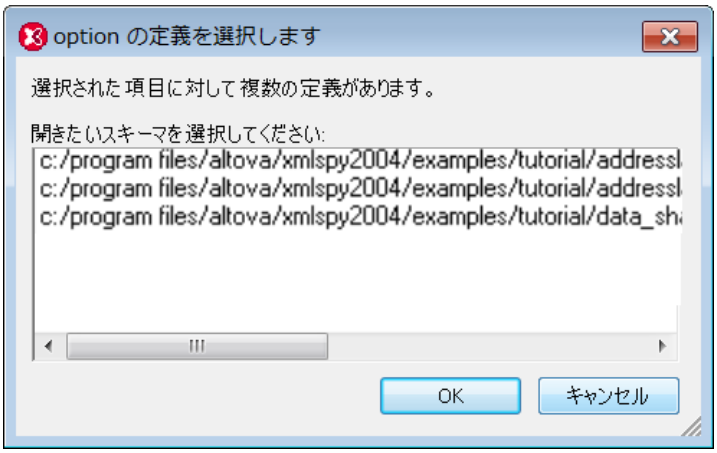


スキーマコンストラクトは、コンポーネント入力ヘルパーの対応するタブである型（グローバル）、名前空間、同一性制約から参照することができます。

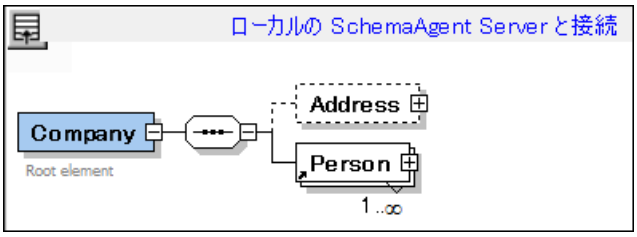
4.6.2 検索パス上のスキーマを開く

この例では、SchemaAgent Server で定義された検索パス上にあるスキーマを開く方法を説明します。ここでは . . \Tutorial フォルダー以下にある DB2schema.xsd ファイルを、アクティブなスキーマとして使用します。コンポーネント入力ヘルパーのグローバルタブをアクティブにします。

1. コンポーネント入力ヘルパーにある青色で記された Company アイテムまで移動し、アイテムをダブルクリックします。定義へ移動ダイアログが開かれます。



2. MFCompany.xsd エントリを選択して、「OK」ボタンで確定します。これにより MFCompany.xsd スキーマが開かれ、Company 要素のコンテンツモデルが表示されます。



メモ: 要素や複合型、または単純型といった SchemaAgent スキーマコンストラクトをダブルクリックすることで、関連するスキーマ (そしてインクルードされた他のスキーマ) が XMLSpy で開かれます。

4.6.3 IIR の使用

XML スキーマでは、Import、Include、そして Redefine (IIR) ステートメントを使用することでスキーマをモジュール化することができます。各メソッドでは異なる名前空間が必要になります。IIR を行う際に必要となる項目は IIR を作成する際に SchemaAgent Client と XMLSpy により自動的にチェックされます。

インポート (Import)、インクルード (Include)、そして再定義 (Redefine) (IR)
スキーマコンストラクトを「挿入」するにはいくつか方法があります:

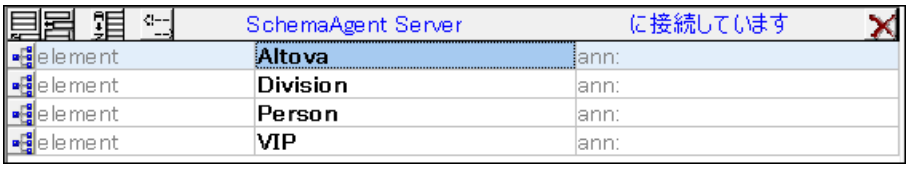
- グローバル要素はコンポーネント入力ヘルパーから、(スキーマビューの) スキーマコンポーネントのコンテンツモデルに直接ドラッグすることができます。
- 新規要素や属性などを定義する際に自動的に開かれるリストボックスから、複合型や単純型のようなコンポーネントを選択することができます。
- 複合型の様なコンポーネントを作成、更新する際に、詳細入力ヘルパーからコンストラクトを選択することができます。

スキーマコンポーネントの組み込み

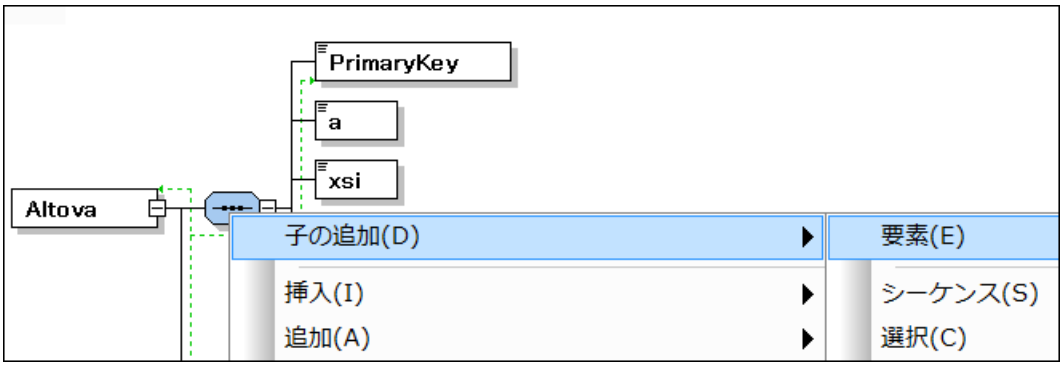
この例では `..\Tutorial` フォルダーにある `DB2schema.xsd` ファイルを使用します。コンポーネント入力ヘルパーのグローバルタブをアクティブにします。

SchemaAgent Server スキーマにあるスキーマコンストラクトを使用するには：

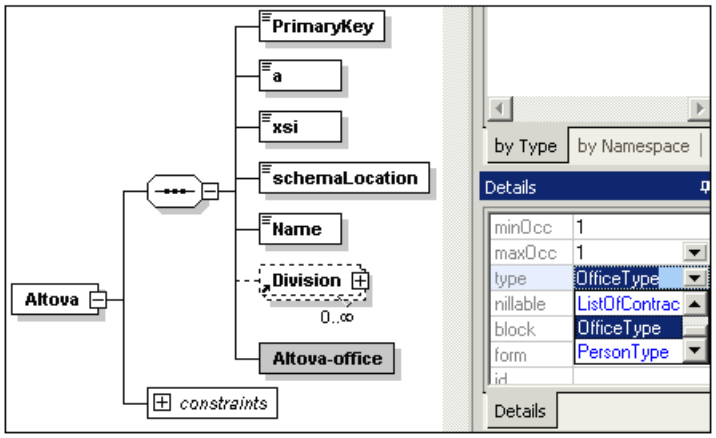
1. SchemaAgent サーバーに接続していることを確認します（$schemaAgent サーバーへ接続を参照）。
2. `DB2schema.xsd` ファイルを、この例のために 例えば `Altova-office.xsd` という名前に変更します。



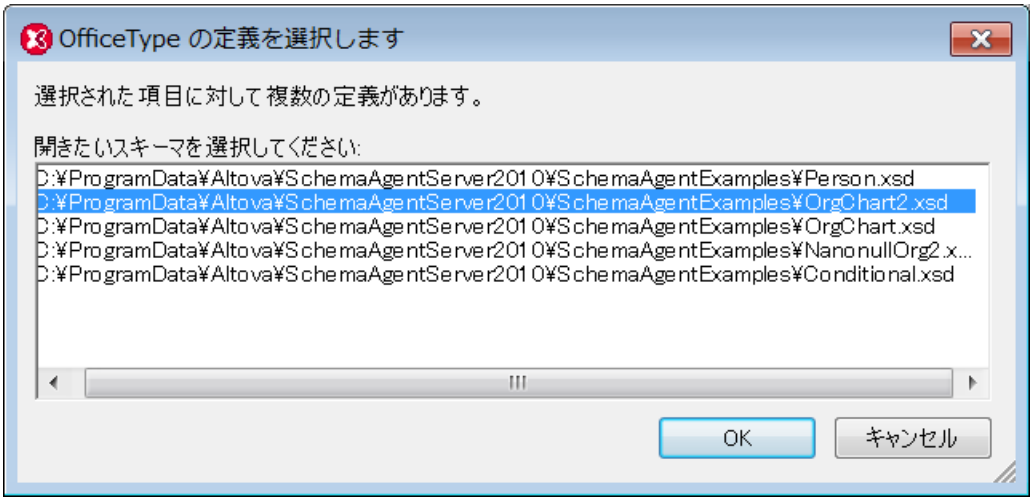
3. スキーマ概要に表示されている `Altova` 要素の アイコンをクリックして、コンテンツモデルを表示します。
4. `Altova` 以下にあるシーケンスコンポジッターを右クリックして、メニューオプションから **「子を追加 | 要素」** を選択します。サーバーのパス上にある全てのグローバル要素を含んだリストボックスがこの時点で自動的に開かれます。



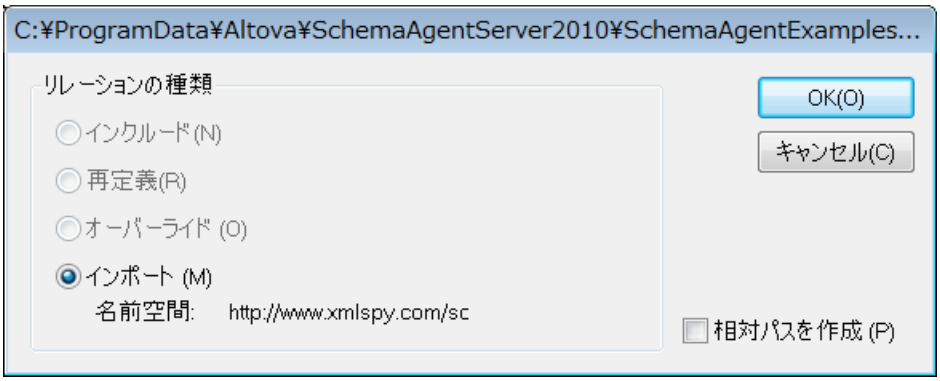
5. 新たに作成した要素の名前を `Altova-office` として、 **「Enter」** で確定します。
6. 詳細入力ヘルパーにある `type` コンボボックスをクリックして、`OfficeType` を選択します。



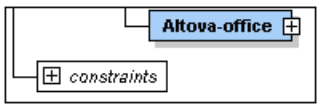
OfficeType の定義を選択するダイアログボックスが表れます。



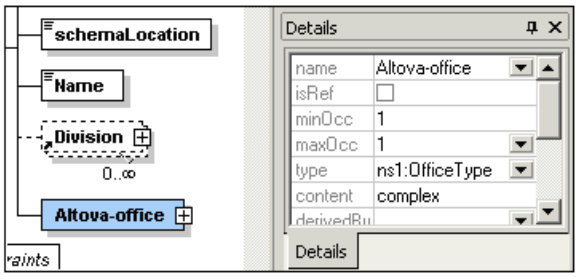
7. `OrgChart2.xsd` を選択し、「OK」で確定します。



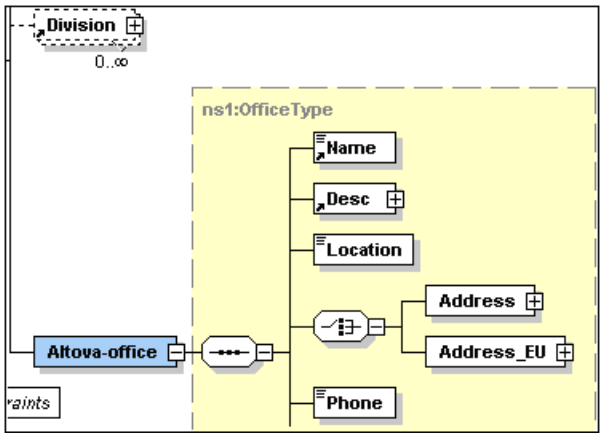
8. 「OK」をクリックします。インポートコマンドが自動的に選択されます。`Altova-office` 要素に展開アイコンが表示されます。



メモ `OrgChart2.xsd` スキーマファイルがインポートされると、各スキーマのターゲット名前空間は異なるものと見なす必要があるため、詳細入力ヘルパーの `type` エントリは `ns1:OfficeType` に変更されます。インポートコマンドもスキーマに追加されます。



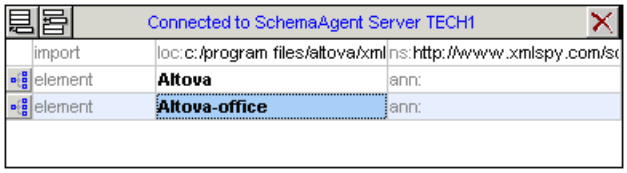
9. 展開ボタンをクリックして、`OfficeType` コンテンツモデルを確認します。



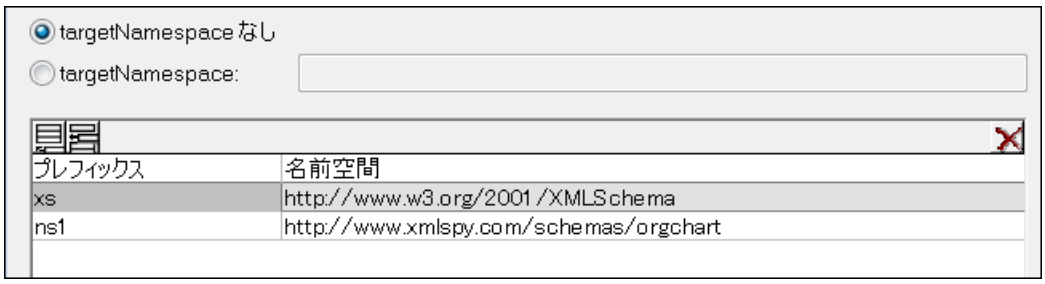
10. **F8** キーを押してスキーマを検証します。「スキーマは妥当です」というメッセージが表示されるはずです。

スキーマの整理:

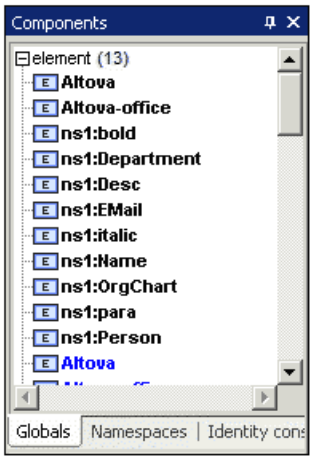
1. コンテンツモデルの `Division` 要素を削除します。
2. 概要へ戻るアイコン をクリックして、スキーマ概要へ画面を切り替えます。
3. 以下のグローバル要素を削除します：`Division`、`Person`、`VIP`。



4. メニューオプションから **「スキーマ設計 | スキーマ設定」** を選択し、名前空間の設定が変更されていることを確認します。



`ns1` プレフィックスが `www.xmlspy.com/schemas/orgchart` 名前空間に追加されています。（スクリーンショットにある）コンポーネント入力ヘルパーには `ns1:` 名前空間プレフィックスからインポートされた全てのコンストラクトが表示されます。



注意:

- XMLSpy を使って SchemaAgent Server コントロールを変更することで、SchemaAgent Server のパス上にある (変更されたスキーマを参照している) 他のスキーマも自動的に更新されます。
- 作業中のスキーマが SchemaAgent Server パス上にある場合、入力ヘルパーで複製されたコンストラクト要素や単純型を (黒色と青色の違いにより) 確認することも可能です。

4.6.4 SchemaAgent でスキーマを閲覧

アクティブならびに関連したスキーマを SchemaAgent を使って確認、編集するには、メニューオプションから **「スキーマ設定 | SchemaAgent で表示 | スキーマまたは関連するスキーマ」** を選択します (以下のスクリーンショットを参照)。

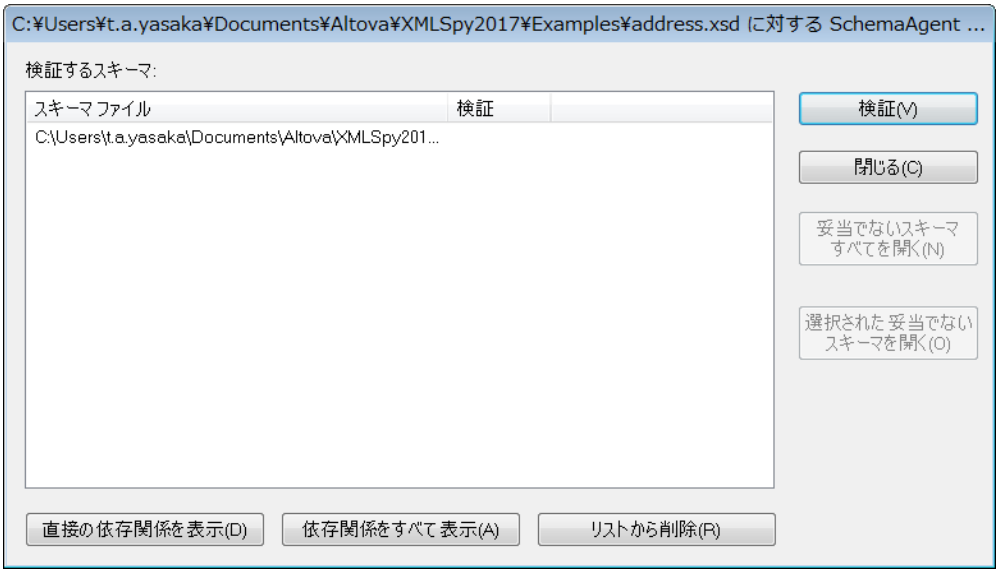
SchemaAgent を使って、アクティブなスキーマのみ (**「ファイルのみ」** コマンド)、またはアクティブなスキーマと、(i) 直接参照されている全てのスキーマか、(i) 直接参照している全てのスキーマか、(ii) 直接関連する全てのスキーマを開くことができます。

4.6.5 SchemaAgent 検証

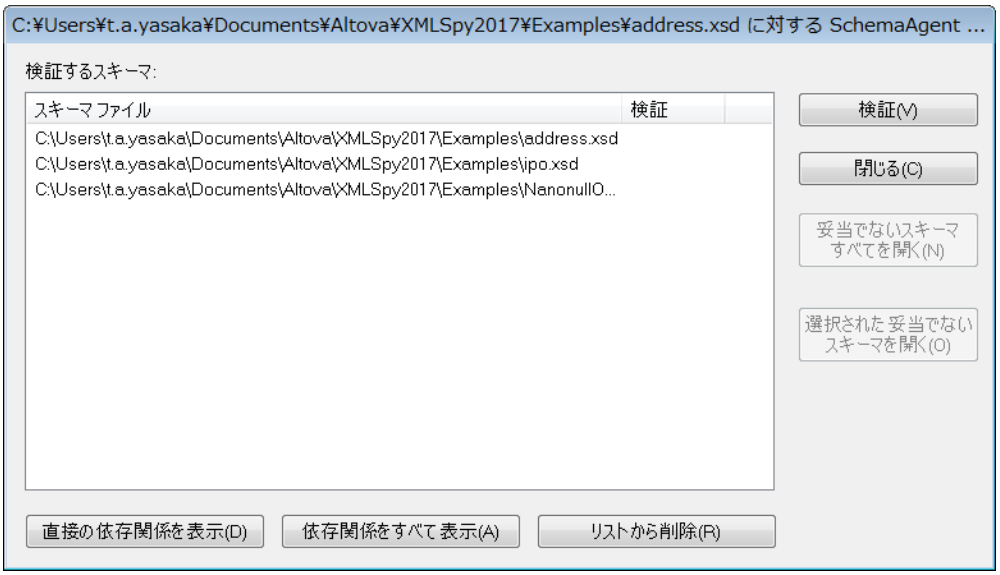
SchemaAgent と組み合わさった XMLSpy により、現在アクティブなスキーマのみならず、それに関連したスキーマの検証も行うことができます。このようなスキーマの検証は、SchemaAgent 検証と呼ばれます。関連するスキーマには (i) 直接の依存関係を持ったスキーマ (直接参照している、または直接参照されているスキーマ) と、(i) 全ての依存関係を持ったスキーマ (直接の依存関係に加え、中間的なスキーマを介することで、他のスキーマに関連したスキーマセットの間接的な依存関係も含まれます) の 2 種類があります。

SchemaAgent 検証がどのように行われるかは、以下の例により示されます。この例では `address.xsd` というファイルが XMLSpy のスキーマビューのアクティブなスキーマであることを前提にしています。SchemaAgent 検証のコマンドを使用にするには、SchemaAgent Server の検索パスにアクティブなファイルと依存関係を持ったファイルが含まれることを確認してください。以下の操作を行ってください:

1. ツールバーにある **「SchemaAgent 検証」** アイコン をクリックするか、メニューアイテムの **「スキーマ設計 | SchemaAgent 検証」** を選択します。この操作により SchemaAgent 検証ダイアログが開かれ、アクティブなスキーマのみ、または関連するスキーマまでも検証を行うか選択することができます (以下のスクリーンショットを参照)。

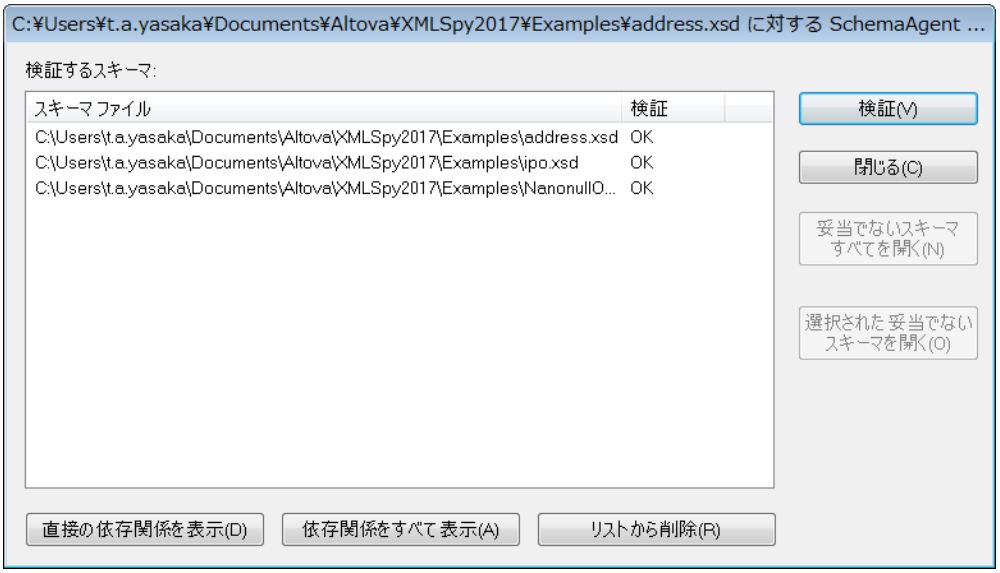


2. スキーマをリストに追加するには、**「直接の依存関係を表示」**または**「依存関係を全て表示」**ボタンをクリックします。この例では、**「依存関係を全て表示」**ボタンをクリックすることで、直接参照されているファイル、または間接的に参照されているファイルがリストに表示されます。



この時点で、必要に応じて（**「リストから削除」**により）スキーマをリストから取り除くことができます。

3. **「検証」**ボタンをクリックして、リストボックスに表示されているスキーマ全ての検証を行ないます。



検証列に検証結果が表示されます。

妥当でないスキーマ全て、または選択された妥当でないスキーマセットを XMLSpy で開くことができます。

4.7 スキーマの中を検索

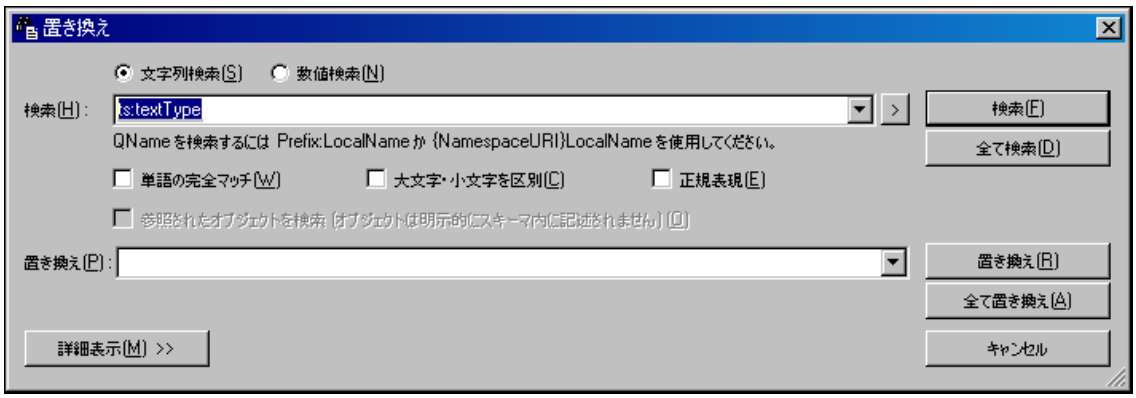
XMLSpy の検索と置き換え機能をスキーマビューで使うことで、インテリジェントな検索を行うことができます。

スキーマビューの検索と置き換え機能は、スキーマファイルがスキーマビューにてアクティブになっている時に使用することができます。以下の2つの方法でこの機能にアクセスすることができます：

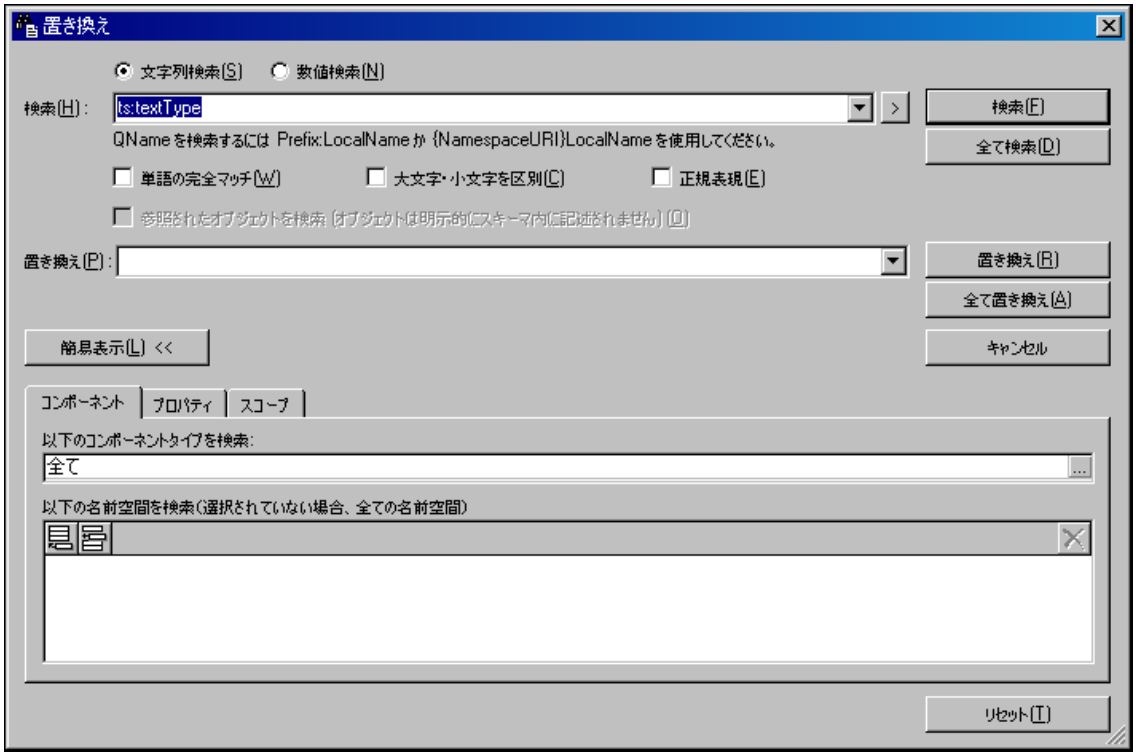
- メニューコマンドから **編集 | 検索」や 編集 | 置換」** を選択する
- スキーマの中を検索ウィンドウにある **検索」** または **置き換え」** ボタンを押下する

コマンドを選択またはボタンをクリックすることで、検索または置き換えダイアログが表示されます。検索ダイアログと比べると、以下のスクリーンショットにある置き換えダイアログには置き換えを行うテキストを表示するテキストフィールドが追加されています。

デフォルトの置き換えダイアログは以下のように表示されます：



詳細表示」 ボタンをクリックすることで、以下のスクリーンショットにあるように、より詳細な検索を行うための条件が表示されます。



使用方法は以下のとおりです：

- 検索ならびに置き換えテキストフィールドにて検索、置き換えを行う単語を入力
- コンポーネントタブにて検索するスキーマコンポーネントを指定
- 検索するコンポーネントのプロパティを指定して検索の対象を狭める
- 現在のドキュメント、プロジェクト、またはフォルダーを指定して検索の範囲をセット
- コマンドの実行
- スキーマの内を検索ウィンドウを使用してコンポーネントの素早いナビゲーションを行う

ダイアログ下部にある **「リセット」** ボタンにより、以下にあるようなデフォルト状態に設定を戻します：

- 検索語無し、置き換え語無し
- コンポーネント：全て
- 名前空間：選択無し
- プロパティの制約：どこでも
- プロパティの追加制約：無し
- スコープ：現在のファイル

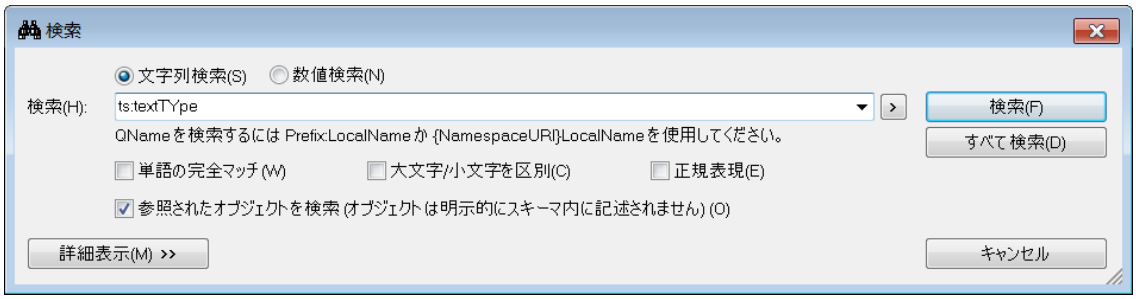
メモ： 置き換えフィールドにて正規表現はサポートされていません。

4.7.1 検索語

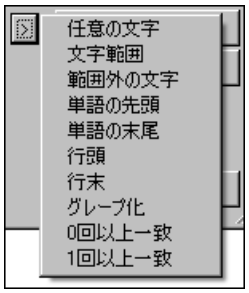
検索語は文字列（文字列ラジオボタン）または数値（数値ラジオボタン）として入力することができます。

文字列検索

以下のスクリーンショットに示される文字列検索において、検索語として入力できるのは、(i) テキスト、(ii) QName、(iii) 正規表現のどちらかになります。QName を使った検索では、テキストフィールドに入力されたドキュメントで使用されているプレフィックス、もしくは名前空間 URI をベースに名前空間が決定されます。以下のスクリーンショットでは、`ts:` プレフィックスが名前空間を決定するためにドキュメント内で使用されているプレフィックスになります。



正規表現を使った検索を行うには、正規表現チェックボックスにチェックを入れて、テキストフィールドに正規表現を入力してください。(以下のスクリーンショットに示される) 検索語入力フィールドの右にある矢印ボタンをクリックすることで表示されるメニューから正規表現の入力を行うことができます。

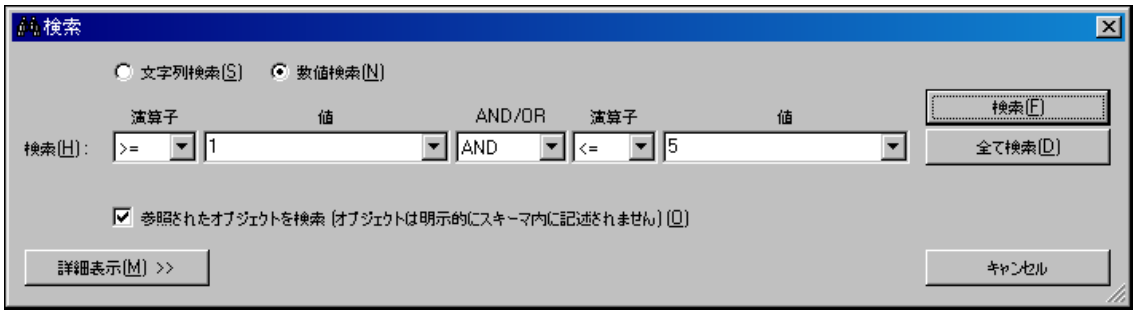


検索語がドキュメント内の単語に完全一致するか、または大文字/小文字を区別した検索を行うか設定することができます。テキストフィールドの下にあるチェックボックスを使って、これらのオプションを使用してください。

複合型の定義やグローバル要素といった参照されているオブジェクト内で検索を行ないたい場合は、参照されたオブジェクトを検索チェックボックスのチェックを入れてください。このオプションは検索ダイアログだけで使用できるもので、置き換えダイアログでは無効化されます。

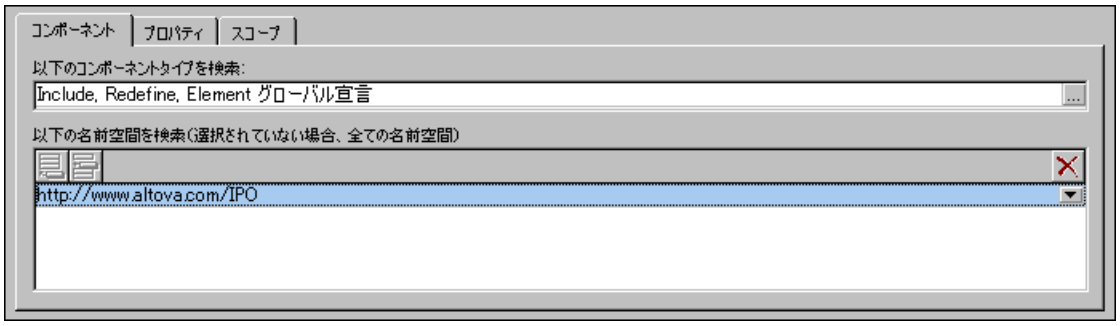
数値検索

数値検索ラジオボタンが選択されていれば、1つの演算子と数値からなる検索条件か、2つの検索条件を AND または OR の論理式により組み合わせることで検索を行うことができます。以下のスクリーンショットでは、2つの検索パラメーターにより、「1以上かつ5以下」という検索条件が指定されています。



4.7.2 コンポーネント

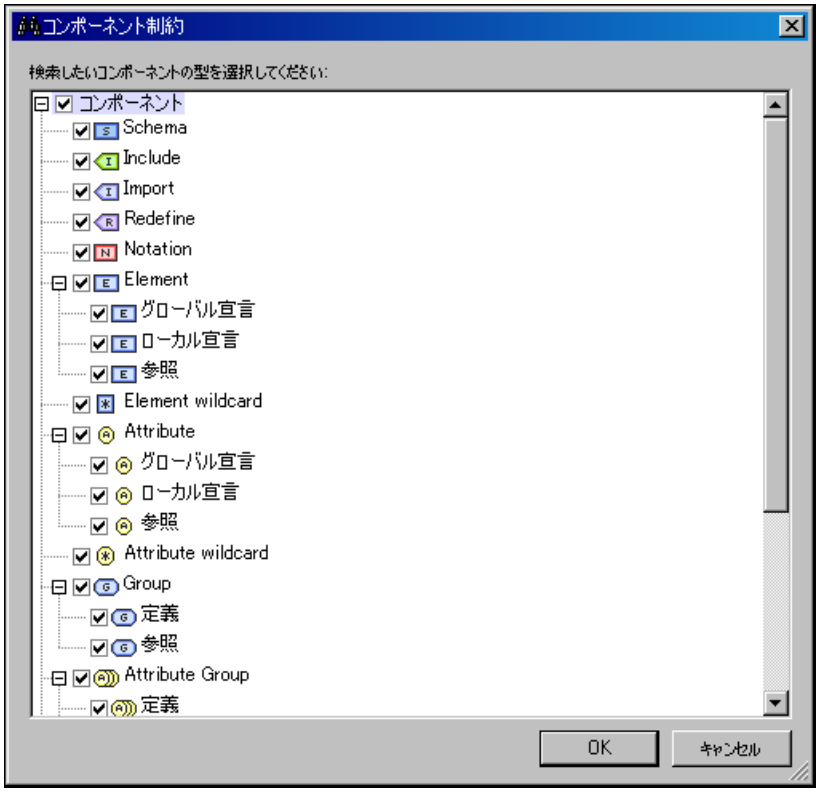
コンポーネントの種類やターゲット名前空間により検索を行う対象を制限することができます。これらのオプションはコンポーネントタブから利用することができます。検索または置き換えダイアログにて **「詳細表示」** ボタンをクリックしてください。検索の対象を狭めるタブが開かれ、そのうちの1つが (以下のスクリーンショットに示される) コンポーネントタブになります。



コンポーネントタブは、(i) 検索を行うコンポーネントの種類と、(ii) 検索を行うターゲット名前空間という2つの部分から構成されます。

コンポーネントの選択

上のスクリーンショットにあるように、テキストフィールドの右に位置している **「追加」** アイコン ... をクリックすることで、検索を行うコンポーネントの種類を選択することができます。この操作により (以下のスクリーンショットに示される) コンポーネント制約ダイアログが表示され、クリック操作で検索を行う対象コンポーネントの種類を選択することができます。リストの上部にある *コンポーネント* アイテムをクリックすることで全てのコンポーネントが選択され、再度クリックしてチェックを外すことで (個々に選択されたコンポーネントも含めて) 全てのコンポーネントの選択を解除します。従って、各コンポーネントを個々に選択するには、*コンポーネント* アイテムの選択が解除されていなければなりません。選択されたコンポーネントはカンマ区切りでテキストフィールドに表示されます (上のスクリーンショットを参照)。



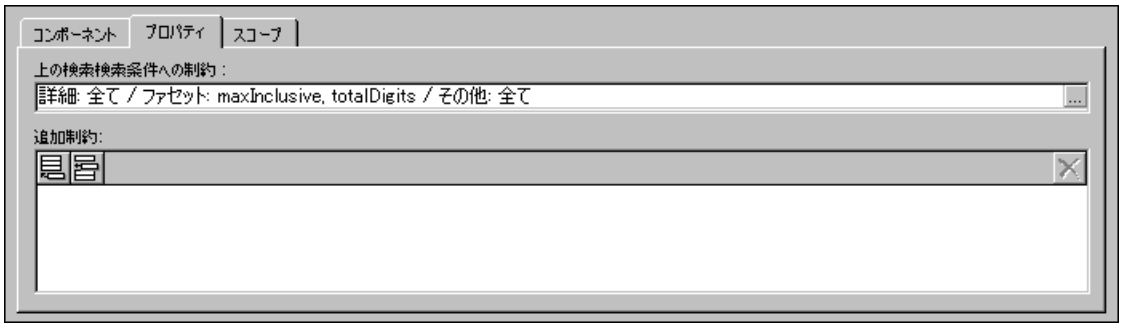
注意: コンポーネントタブまたは検索、置き換えダイアログが再度開かれたとしても、前回のコンポーネント選択状態は保持されたままになります。

名前空間の選択

検索の対象とするターゲット名前空間を設定するには、**追加**または**挿入**アイコンをクリックして、目的の名前空間を入力してください。ターゲット名前空間が指定されていない場合、全てのターゲット名前空間が選択されます。ペインに入力されたターゲット名前空間を削除するには、そのターゲット名前空間を選択して、**削除**アイコンをクリックしてください。

4.7.3 プロパティ

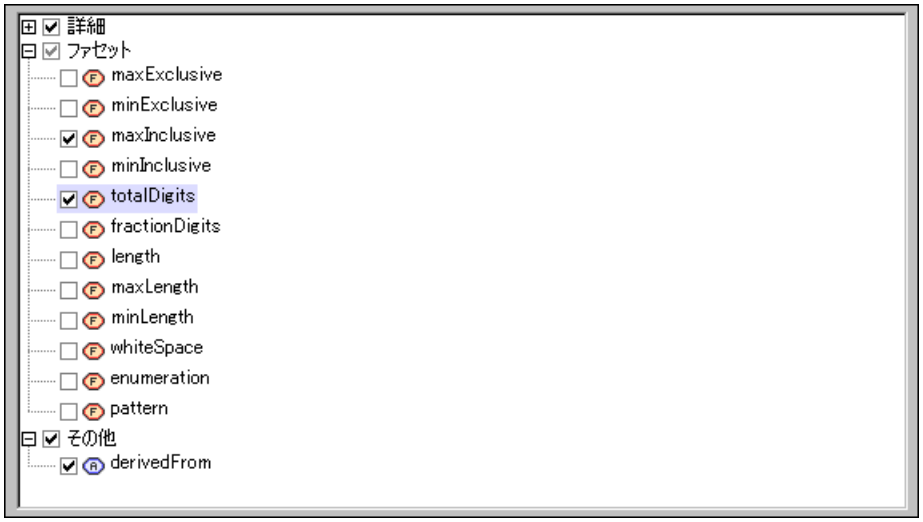
プロパティタブのオプションを使用することで、コンポーネントプロパティ(詳細ならびにファセット)や、プロパティの内容で検索結果を絞り込むことができます。**詳細表示**ボタンをクリックすることで検索または置き換えダイアログが展開表示されるので、プロパティタブを選択します(以下のスクリーンショットを参照)。



プロパティタブは、(i) (メインの検索テキストフィールドに表示される)検索対象の制約と、(ii) コンテンツ制約(以下の[追加制約](#)を参照)の2つの部分から構成されます。

プロパティの選択

テキストフィールドの右にある**追加**アイコン[...]をクリックすることで、(以下のスクリーンショットにあるように)検索の対象となるプロパティの種類を選択することができます。プロパティ制約ダイアログが表示され、アイテムをクリックすることで検索を行うプロパティを選択することができます。プロパティは(i) 詳細、(ii) ファセット、(iii) その他の3つのグループに分類されています。詳細、ファセット、またはその他をクリックすることで、グループにある全てのプロパティが選択され、グループを再度クリックすると、そのグループのプロパティ全てが(個別に選択されたプロパティを含めて)選択解除された状態になります。個々のプロパティはグループアイテムが選択されていない状態でないと選択することができません。選択されたプロパティは(上のスクリーンショットにあるように)テキストフィールドに表示されます。

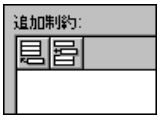


注意: プロパティタブまたは検索、置き換えダイアログが再度開かれたとしても、前回のプロパティ選択状態は保持されたままになります。

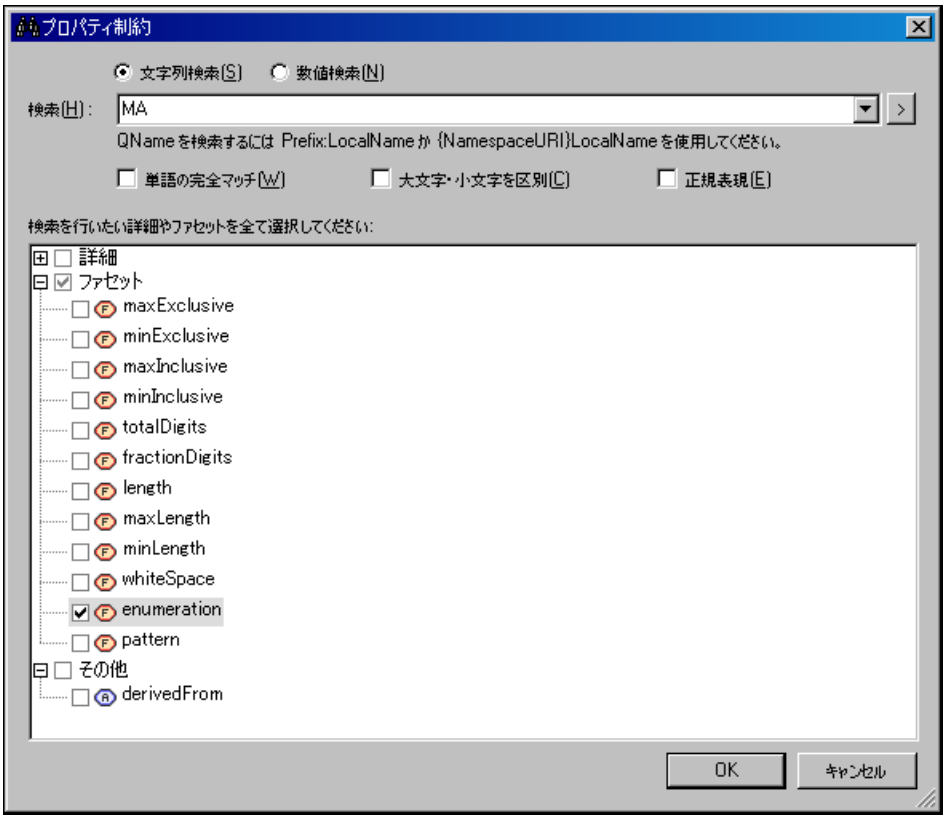
追加制約

追加制約により検索を行うプロパティの持つ値を指定することができます。例えば、列挙の MA (米国のマサチューセッツ州) という値を持つ state という名前の要素を探している場合、enumeration と MA という値を追加制約オプションで指定することで希望する検索を行うことができます。以下のような操作を行ないます：

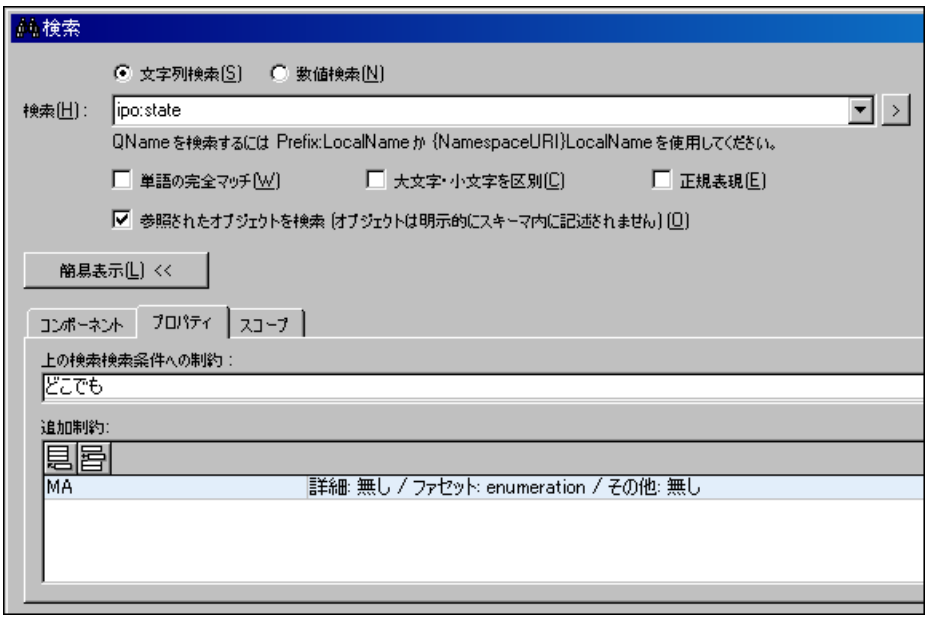
1. 追加制約ペインにて、以下のスクリーンショットに示される **「追加」** または **「挿入」** アイコンをクリックします。



2. ペインに新たな行が追加され、プロパティ制約ダイアログが表示されます。全てのアイテムを選択解除して、enumeration プロパティだけを選択します。



3. ダイアログの上部にあるテキストフィールドに、検索を行ないたい列挙の値 (上のスクリーンショットにあるように、この例では MA) を入力します。
4. **「OK」** をクリックすると、以下のスクリーンショットにあるように追加制約ペインにて新たに作成された行に制約が入力されます。



上で示されるスクリーンショットにて検索語が `ipo:state` となっていることに注目してください。プロパティタブにある「どこでも」により、全てのプロパティが検索の対象となりますが、追加制約により検索は `MA` という値を持つ列挙に制限されます。

複数の追加制約により、検索結果を更に狭めることができます。追加制約を削除するには、追加制約を選択して削除アイコンをクリックしてください。

注意: プロパティタブまたは検索、置き換えダイアログが再度開かれたとしても、前回の追加制約選択状態は保持されたままになります。

4.7.4 スコープ

検索の範囲を（以下のスクリーンショットにある）スコープタブにより設定することができます。検索範囲をファイル、またはスキーマビューで現在選択されているスキーマコンポーネントから選択することができます。

ファイルオプションが選択されている場合、更に以下のオプションから選択範囲の指定を行うことができます：

- **現在のファイル**：現在アクティブなファイルが検索範囲になり、インクルード、インポート、再定義されたファイルも検索対象に含めるオプションが表示されます。
- 開いたファイル：XMLSpy で開かれている全ての XML スキーマ (XSD) ファイルが検索範囲になります。全て検索ならびに全て置き換えコマンドのみが有効になり、通常の検索を使用することはできません。
- **プロジェクト**：現在アクティブなプロジェクトが検索範囲になり、外部フォルダーをスキップするオプションが表示されます。全て検索ならびに全て置き換えコマンドのみが有効になり、通常の検索を使用することはできません。（**ツール**

| オプション| ファイルタイプ| デフォルトビュー」で設定することができる) .xsd ファイル拡張子のデフォルトビューがスキーマビューでない場合、.xsd ファイルは検索対象となりません。

- **フォルダー** 検索範囲となるフォルダーを指定することができ、サブフォルダーを含めるオプションが表示されます。全て検索ならびに全て置き換えコマンドのみが有効になり、通常の検索を使用することはできません。(**「ツール| オプション| ファイルタイプ| デフォルトビュー」**で設定することができる) .xsd ファイル拡張子のデフォルトビューがスキーマビューでない場合、.xsd ファイルは検索対象となりません。
- インクルード/インポート/再定義されたファイルのオプションをチェックすることで、これらのファイルを検索範囲に含めることができます。

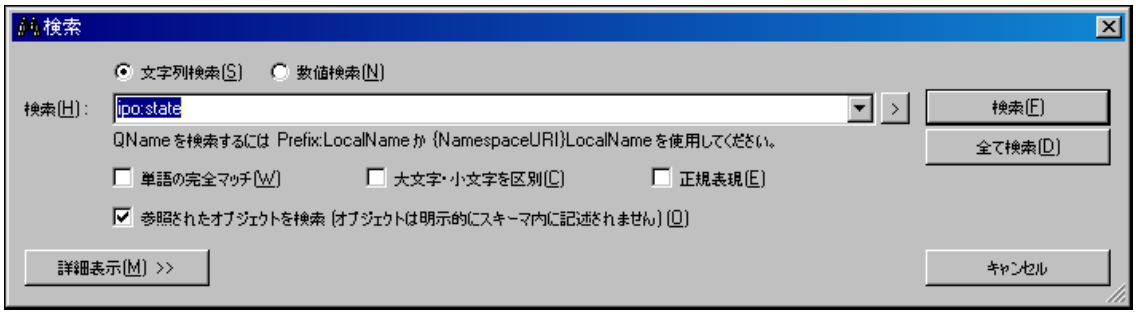
置き換えダイアログでは、ディスク上のファイルを直接置き換えるか、ファイルをXMLSpy にて開くかを選択することができます。ダイアログにあるボタンから選択してください。

4.7.5 検索と置き換えコマンド

検索ならびに置き換えダイアログ内の **「検索」**コマンドの動作はこれまでの検索方法とは多少異なっています。ここでは両ダイアログの **「検索」**コマンドならびに **「置き換え」**コマンドの解説を行ないます。

検索ダイアログ

検索語、そして必要な場合はその他の条件を入力した後、 **「検索 (F3)」**または **「全て検索」**コマンドをクリックします。



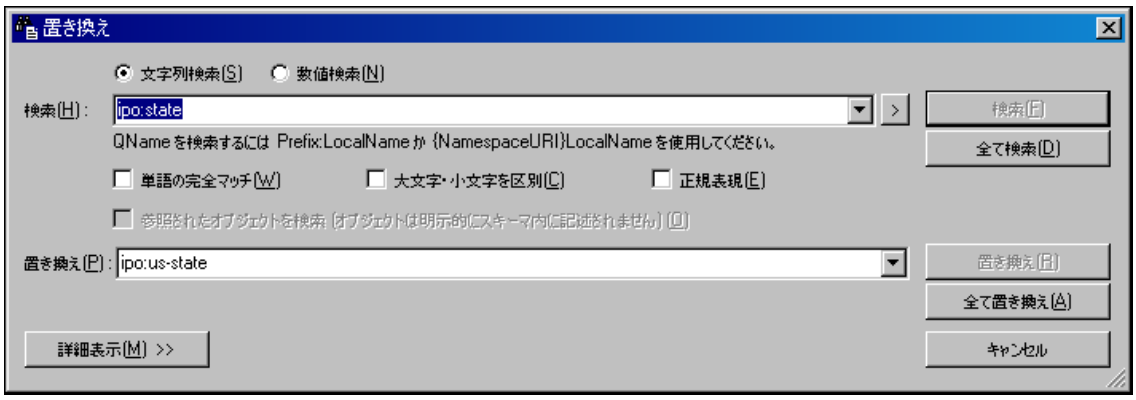
ダイアログにある **「検索 (Ctrl+F)」**コマンドをクリックすることにより検索ダイアログが閉じられ、検索条件や検索語といった検索条件にマッチする検索範囲内の次の出現場所が特定されます。スキーマビューで現在選択されているコンポーネントの位置を基準にして、次の検索結果は求められます。検索範囲の最後まで検索が行われても、検索範囲の最初から自動的に検索が続けられることはありません。そのため、検索を始める前に、現在選択されているコンポーネントの位置が検索を行ないたい箇所の前にあることを確認する必要があります。

「検索」の結果はスキーマビューにてハイライトされ、スキーマの中を検索ウィンドウにもその結果が表示されます。スキーマの中を検索ウィンドウにある結果をクリックして、スキーマビューの対応するアイテムをハイライトさせることもできます。

「全て検索」コマンドをクリックすると、検索ダイアログが閉じられ、検索結果の全てがスキーマの中を検索ウィンドウに表示されます。

置き換えダイアログ

置き換えダイアログでは **「検索」**コマンドをクリックすることで、スキーマビューで現在選択されている箇所から、次の検索語の出現箇所を特定します。 **「置き換え」**をクリックすることで、この出現を置き換えることができます。



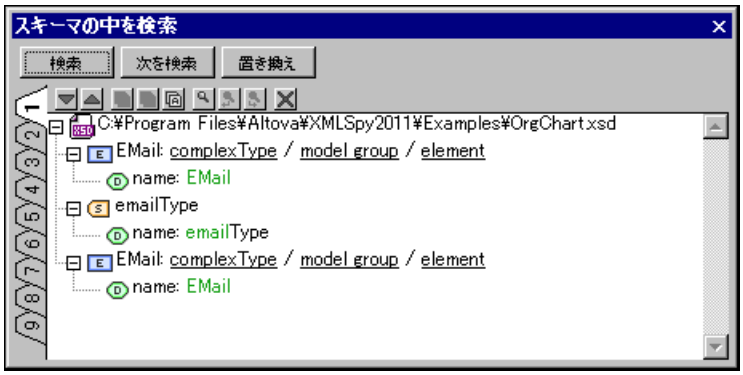
全て検索」コマンドを使うと、置き換えダイアログが閉じられ、検索の結果全てがスキーマの内を検索ウィンドウに表示されます。

全て置き換え」コマンドは、検索により見つかった箇所全ての置換えを行ないます。置き換えダイアログが閉じられ、発見された箇所全てがスキーマの中を検索ウィンドウに表示されます。

メモ　置き換えフィールドにて正規表現はサポートされていません。

4.7.6 結果と情報

検索」、全て検索」、置き換え」、または **全て置き換え」**コマンドが実行されると、コマンド実行の結果が（以下のスクリーンショットにあるように）スキーマの中を検索ウィンドウに表示されます。検索された単語は緑色で表示されます（以下のスクリーンショットの例だと検索語の `email` が緑色で表示されます）。スキーマファイル内の位置も表示されていることに注目してください。



全て検索」または **全て置き換え」**コマンドを使うと、ドキュメント内の出現箇所全てが表示されます。

メモ　ウィンドウの上部にある検索や置き換えボタンを使えば、それぞれ検索、置き換えダイアログを表示させることができます。**次を検索」**ボタンを使えば、検索語が次に出現する箇所を特定することができます。

スキーマの内を検索ウィンドウの機能

検索などの結果は（1から9までの番号が振られた）9つの異なるタブの中に表示されます。例えばある検索結果を1つのタブに収めておき、別のタブで新たな検索をすることにより結果の比較を行うことができます。スキーマの中を検索ウィンドウの結果をクリックすることで、メインウィンドウのスキーマビューに表示されている関連するコンポーネントがハイライトされます。これらのナビゲーション機能により、目的のコンポーネントへ素早くたどり着くことができます。

スキーマの中を検索ウィンドウでは以下のツールバーコマンドを利用することができます：

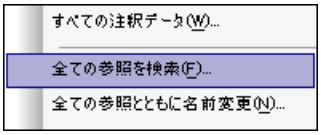
- **「次」**ならびに **「前」**アイコンにより、現在選択されているメッセージの次（もしくは前）にあるメッセージが選択されます。
- **「メッセージをコピー」**コマンドにより、選択されたメッセージ、子を含む選択されたメッセージ、そして全てのメッセージをクリップボードにコピーします。
- **「検索」**コマンドによりスキーマの中を検索ウィンドウ内にあるテキストを検索します。
- **「クリア」**コマンドにより、現在アクティブなタブのメッセージが全て消去されます。

4.7.7 グローバルコンポーネントの検索と名前変更

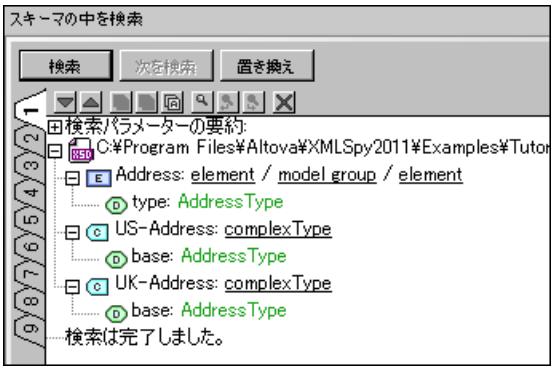
ユーザーは、選択されたファイルのXML スキーマや、関連する全てのスキーマファイル内にある名前付きグローバルコンポーネントの検索ならびに名前変更を行うことができます。名前付きのグローバルコンポーネントとは、以下を除く全てのグローバルコンポーネントです：Include、Import、Redefine、Annotation、Comment、PI。

以下の操作を行ってください：

1. スキーマ概要にて検索または名前の変更を行うグローバルコンポーネントを選択します。
2. 選択されたコンポーネントを右クリックすることで表示されるコンテキストメニューから、目的のコマンド（**「全ての参照を検索」**または **「全ての参照とともに名前変更」**）を選択します。



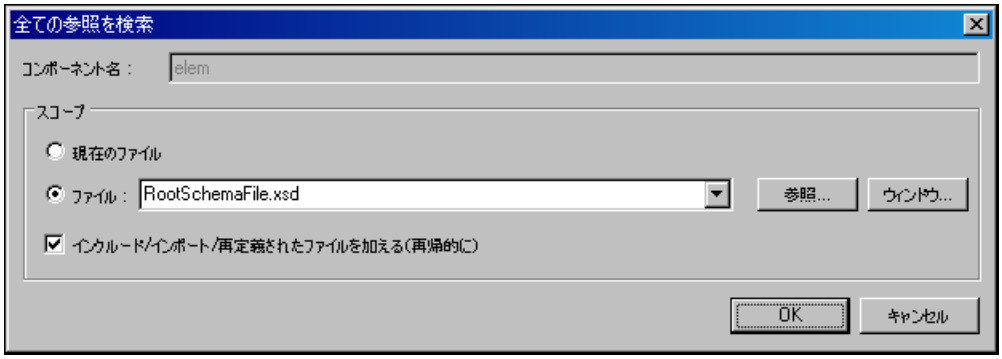
3. 表示されるダイアログにて、検索を行う範囲（または名前変更オペレーション）を選択します。名前変更オペレーションの場合、新たなグローバルコンポーネントの名前を入力します。
4. **「OK」**をクリックすることで、検索の結果がスキーマの中を検索ウィンドウに表示されます（以下のスクリーンショットを参照）。



グローバルコンポーネントへの参照を行っているファイルの場所が表示されます（上のスクリーンショットを参照）。名前の変更が行われたコンポーネントも全て表示されます。

全ての参照を検索

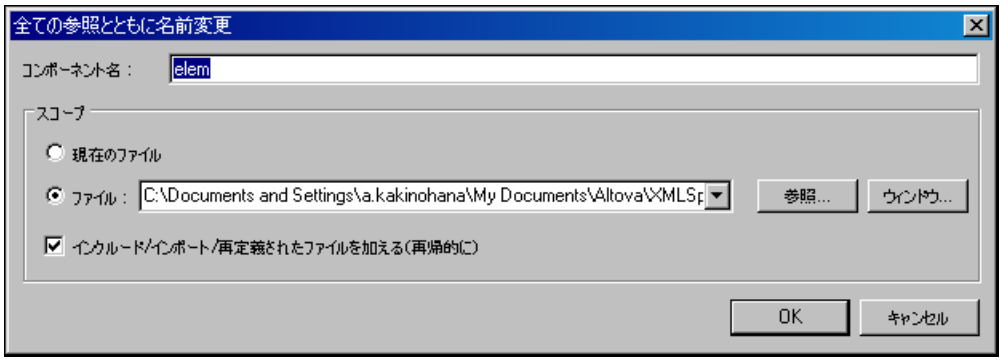
全ての参照を検索ダイアログ（以下のスクリーンショットを参照）を開くには、(i) スキーマ概要にてグローバルコンポーネントを右クリックし、(ii) コンテキストメニューから、**「全ての参照を検索」**コマンドを選択します。



グローバルコンポーネント名が**コンポーネント名**フィールドにグレーで表示されます（編集は行えません）。検索の範囲対象を現在のファイルで行うか、または参照により選択したファイル内で行うかを指定することができます。ダイアログ下部にあるチェックボックスを有効にすることで、（Include、Import、Redefine により）関連するファイルを検索の対象に加えることもできます。

全ての参照とともに名前変更

グローバルコンポーネントの名前を変更するには、コンポーネントを右クリックし、表示されるコンテキストメニューから**「全ての参照とともに名前変更」**を選択します。全ての参照とともに名前変更ダイアログが表示されます（以下のスクリーンショットを参照）。



選択されたグローバルコンポーネントに対して、新たに使用したい名前をコンポーネント名テキストフィールドに入力します。検索ならびに名前変更の範囲対象を、現在のファイルにするか、または参照により選択されたファイル内で行うかを指定することができます。ダイアログ下部にあるチェックボックスを有効にすることで、（Include、Import、Redefine により）関連するファイルを検索の対象に加えることもできます。

5 XSLT

Altova ウェブサイト: XSLT エディター

このセクションでは以下のような構成でXSLT の説明を行ないます：

- XSLT ドキュメントの編集: XMLSpy におけるXSLT ドキュメントの編集機能について説明します。
- XSLT 処理: 選択されたエンジンを使って XMLSpy GUI からXSLT 変換を行う方法について解説します。このセクションではXMLSpy で行うことができるXSLT の設定についても説明されます。
- XSL アウトライン: 複数のXSL ファイルの閲覧、ナビゲーション、そして管理を行うのに便利なXSL アウトラインならびにXSL 情報ウィンドウについて説明します。

XPath の評価

XML ドキュメントがアクティブになっている際、XPathウィンドウを使用することで、XPath 条件式の評価を行うことができます。この機能により、XPath 条件式が評価される様子を素早く確認することができます。XPath 条件式を入力すると、ドキュメントルートまたはXML ドキュメント内で選択されたコンテキストノードに相対的な評価が行われます。条件式評価の結果は、すぐにXPath ウィンドウにて表示されます。XPath ウィンドウの使用方法については、イントロダクション| XPath ウィンドウのセクションを参照ください。

XSLT デバッガーとプロファイラー

XMLSpy にはXSLT プロファイラーならびにXSLT デバッガーが搭載されており、正確で効率のよいXSLT スタイルシートを素早く作成することができます。これらの機能はXSLT とXQuery セクションにて記述されます。

その他の XSLT 機能

このセクションで記述される様々な機能のより詳しい情報については、ユーザーリファレンスの関連するメニューコマンドを参照ください。

Altova XSLT エンジン

Altova XSLT 1.0、2.0ならびに 3.0 エンジンについては付録のXSLT とXQuery エンジンの情報を参照ください。

RaptorXML を使ったコマンドラインやバッチ処理

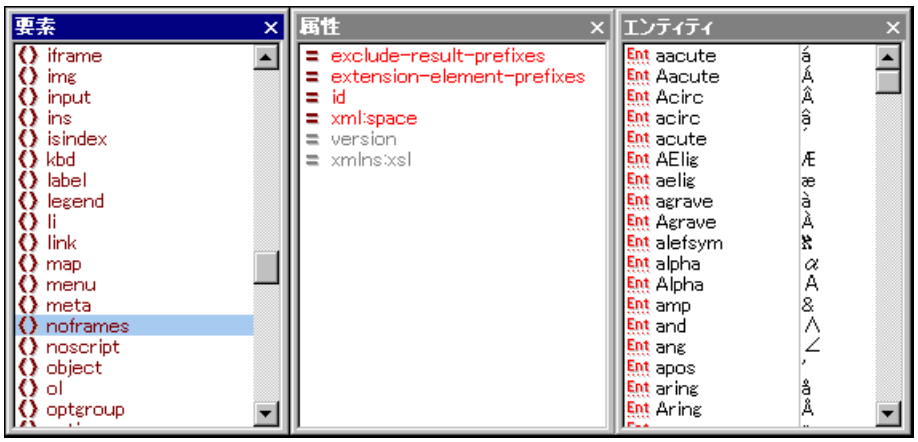
XMLSpy GUI のプロジェクト機能を使用して、バッチ処理を行うことができます。より柔軟性のある処理を求める場合は、速いXML 検証　XSLT 変換、XQuery実行機能を搭載したAltova のRaptorXML 製品を使用してください。をお使いください。コマンドラインやバッチ処理によりXSLT 変換を行うのにRaptorXML は最適のツールです。

5.1 XSLT ドキュメント

XSLT 1.0、2.0、および 3.0 ドキュメントは、その他の XML ドキュメントと同じようにテキストビューやグリッドビューにて編集することができます。ドキュメントが開かれるデフォルトのビューは、オプションダイアログのファイルタイプタブにて設定することができます。

入力ヘルパー

要素、属性、そしてエンティティ入力ヘルパーを利用することができます。入力ヘルパーにて表示されるアイテム情報は XMLSpy に内蔵されており、XSLT ドキュメント内で行われる参照とは独立したものです。



以下の点に注意してください：

1. 新規 XSLT ドキュメントが（**「ファイル | 新規作成」**メニューの）新規ドキュメント作成ダイアログにて作成されたとき、XSLT 1.0、XSLT 2.0、または XSLT 3.0 により作成されたドキュメントの種類によって違いはありますが適切な XSLT 要素と属性が入力ヘルパーにロードされます。それに加えて HTML 要素や属性、HTML 2.0 エンティティセット、Latin-1、特殊な文字、そしてシンボルがロードされます。
2. **「ファイル | 新規作成」**の新規ドキュメントダイアログにて XML ドキュメントが作成され、XSLT コンテンツがそこから作成された場合、XML 文字エンティティ以外のアイテムは入力ヘルパーに表示されません。
3. 新規ドキュメントダイアログを通して作成された XSLT ドキュメントが開かれた場合、上の 1 番目の様に、入力ヘルパーにアイテムが表示されます。
4. 新規ドキュメントダイアログにて XSLT ドキュメントとして作成された**なかった** XSLT ドキュメントが開かれた場合、上の 1 番目の様に入力ヘルパーに XSLT アイテムが表示されます。更に、XSL-FO 要素や属性がテキストビュー入力ヘルパーに表示されます。
5. 要素入力ヘルパーにある要素のプレフィックスは以下の様に固定されています：XSLT 要素の `xsl:`、HTML 要素のプレフィックス無し、XSL-FO 要素の `fo:`。入力ヘルパーを使用するには、入力ヘルパーに表示される内蔵のプレフィックスに合致させる形で、XSLT ドキュメントで宣言された名前空間にもプレフィックスが記述されている必要があります。

自動補完

テキストビューでは入力中に自動補完のポップアップが表示され、入力されたテキストにマッチした最初のアイテムがハイライトされます。要素が入力されている時、要素のリストがポップアップされ、アルファベット順で最も近いアイテムがハイライトされます。同様に、属性が入力されているとき、適用することのできる属性のリストがポップアップで表示されます。入力ヘルパーの以前のセクションで記述されたルールに従ってリストに表示されるアイテムは決定されます。

XPath インテリジェント編集

XSLT ドキュメントにてXPath 条件式を入力することができる場合（例えば、select 属性の値など）、インテリジェントなXPath 編集機能を利用することができます。

- 入力中にかっこを一致させるなどの、XPath のための構文の色分けをビルドします。
- XPath 関数にカーソルをポイントすると、関数に関する情報が含まれるヒントが表示されます。
- XPath 関数と軸は、入力中にポップアップ内で提示されます。**上/下** カーソルを使用して、提示されたリストを上下に移動することができます。ポップアップ内で関数がハイライトされる場合、関数に関する情報が追加ポップアップ内に表示されます。
- XML ファイルが情報ウィンドウに割り当てられている場合、XML ファイルの要素ならびに属性がポップアップに表示されます。

XSLT ドキュメントの検証

XMLSpy に内蔵されているXSLT スキーマに対してXSLT ドキュメントの検証を行うことができます（**XML | XML 検証 (F8)**）をクリックします）。XSLT ドキュメントがXSLT 1.0、 2.0および 3.0（`xsl:stylesheet` 要素の `version` 属性により指定されます）により、適切な内蔵スキーマが選択されます。

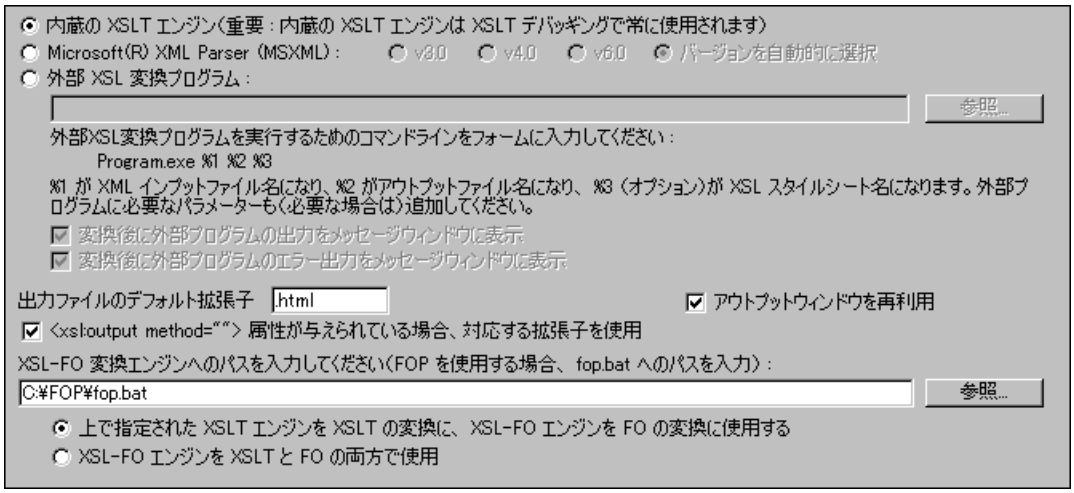
5.2 XSLT 処理

XMLSpy GUI では、2種類のXSLT 変換を利用することができます：

- 「**XSL/XQuery | XSL 変換 (F10)**」コマンドにより、XSLT スタイルシートを使ったXML 変換処理を行ない、スタイルシートに記述されたフォーマットに従った変換が行われます。
- 「**XSL/XQuery | XSL:FO 変換**」コマンドにより、(i) XML からFO、そしてFO からPDF への2ステップの変換、または (ii) FO からPDF への変換が行われます。

変換を行うXSLT プロセッサーの指定

変換に使用されるXSLT エンジンは[オプションダイアログのXSL タブ](#)にて指定されます（以下のスクリーンショットを参照）。



利用可能なオプションについては[ユーザーリファレンス](#)セクションを参照ください。XSL タブで指定されたエンジンがXSLT 変換で使用されます。XSL:FO 変換では、それに加えて他の FO プロセッサーに同封されているXSLT というXSLT エンジンオプションを使用することができます。このオプションを選択するには、XSL タブの下部に表示される、対応するラジオボタンを選択してください。 (*see screenshot above*).

FO プロセッサーの指定

FO からPDF への変換時に使われるFO プロセッサーは、[オプションダイアログのXSL タブ](#)の下部にあるテキストフィールドから指定することができます（上のスクリーンショットを参照）。

XSLT 1.0、2.0、3.0 Altova のXSLT エンジン

スタイルシートのXSLT バージョンは`xsl:stylesheet` （または`xsl:transform`）要素の`version` 属性により指定されます。XMLSpy にはAltova XSLT 1.0、Altova XSLT 2.0、およびAltova XSLT 3.0 エンジンが内蔵されており、`version` 属性の値（`1.0` または`2.0` または`3.0`）により、適切なエンジンが使用されます。

XSLT 変換

XSL 変換 (F8) コマンドは以下のようなシナリオで使用することができます：

- GUI にてアクティブかつXSLT ドキュメントが割り当てられているXML ドキュメントの変換。XSLT ドキュメントが割り当てられていない場合、**「XSL 変換(F8)」**コマンドを選択した際に割り当てを行うよう求められます。
- GUI にてアクティブなXSLT ドキュメントの変換。**「XSL 変換(F8)」**コマンドをクリックすることで、アクティブなXSLT スタイルシートにより変換を行なうXML ファイルを選択するよう促されます。
- プロジェクトフォルダーやファイルの変換。プロジェクトフォルダーまたはファイルを右クリックし、コマンドを選択してください。

バックマッピング

バックマッピング機能が有効化されていると、XSLT 変換が実行され、結果ドキュメントが元のXSLT+XML ドキュメントにバックマッピングされます。結果ドキュメント内のノードをクリックすると、**XSLT 命令** とその特定の結果フラグメントを生成した**XML ソースデータ**また、XSLT 命令、または、XML データノードをクリックすると、他の2つのドキュメント内の対応するノードがハイライトされます。詳細に関しては、**XSL/XQuery | XSLT/XQuery バックマッピングの有効化** を参照してください。

XSL:FO 変換

XSL:FO 変換コマンドは以下のようなシナリオで使用することができます：

- GUI にてアクティブかつXSLT ドキュメントが割り当てられているXML ドキュメントの変換。XML ドキュメントは、まず指定されたXSLT エンジンを使ってFO に変換されます。その後、FO ドキュメントは指定されたFO プロセッサーによりPDF 出力へと変換されます。XSLT ドキュメントが割り当てられていない場合、**「XSL:FO 変換」**コマンドを選択した際に割り当てを行うよう求められます。
- 指定されたFO プロセッサーによりFO ドキュメントをPDF へ変換。
- GUI にてアクティブなXSLT ドキュメントの変換。**「XSL:FO 変換」**コマンドをクリックすることで、アクティブなXSLT スタイルシートにより変換を行なうXML ファイルを選択するよう促されます。
- プロジェクトフォルダーやファイルの変換。プロジェクトフォルダーまたはファイルを右クリックし、コマンドを選択してください。

XSL:FO 出力ダイアログについての詳細は、ユーザーリファレンスを参照ください。

XSLT のパラメーター

Altova XSLT エンジンを使用しているのであれば、XSLT パラメーターをGUI ダイアログに保管しておくことができます。全ての保管されたパラメーターは、変換時にXSLT ドキュメントへと送られます。詳しい情報については、XSLT パラメーター / XQuery 変数コマンドに関する記述を参照ください。

RaptorXMLを使ったバッチ処理

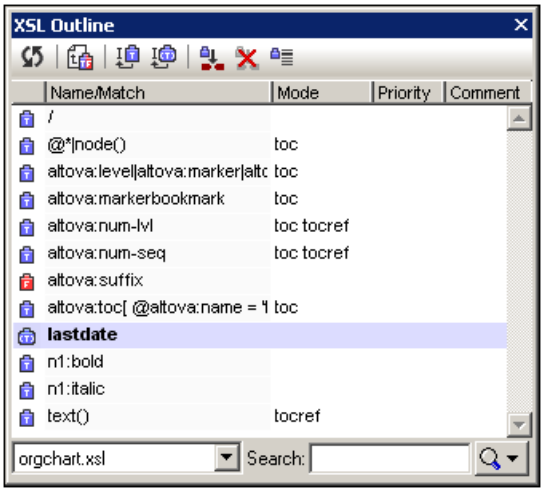
RaptorXMLは無料のスタンドアロンアプリケーションで、Altova の最新 XML バリデーター、XSLT エンジン、そしてXQuery エンジンが搭載されています。コマンドライン、COM インターフェース、Java プログラム、そして .NET アプリケーション内から使用することができ、XML ドキュメントの検証、XSLT スタイルシートを使用したXML ドキュメントの変換、そしてXQuery ドキュメントの実行を行うことができます。

RaptorXMLを使うとXSLT 変換タスクを自動化することができます。例えば、RaptorXMLを呼び出してドキュメントのセットを変換するバッチファイルを作成することができます。詳細はRaptorXML ドキュメンテーションを参照ください。

5.3 XSL アウトライン

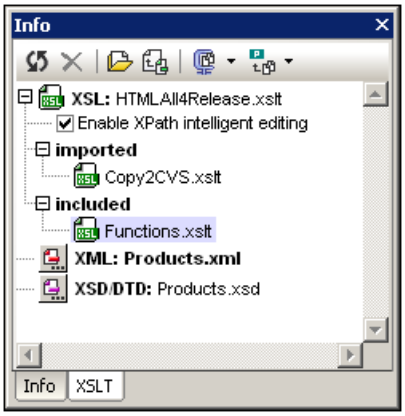
XSLT ドキュメントがXMLSpy のアクティブなドキュメントである場合、ドキュメントの構造に関する情報がXSL アウトラインウィンドウに表示され、XSLT ドキュメントに関連したファイル情報が (XSLT ドキュメントがXMLSpy でアクティブになっている時にだけ表示される) 情報ウィンドウのXSLT タブに表示されます。更に、これら2つのウィンドウにてXSLT ドキュメントの編集や、関連するファイルの管理を行ないやすくする数々のコマンドが利用可能になります。

下のスクリーンショットにある**XSL アウトラインウィンドウ**では、以下の操作を行うことができます：



- アクティブなXSLT ドキュメント、インポート/インクルードされた全てのドキュメントにあるテンプレートと関数を閲覧。
- 名前、マッチ、モード、優先度、またはコメントによるテンプレートと関数の並び替え。
- 名前 表現により特定のテンプレートを検索。
- XSL アウトラインを使ったXSLT ドキュメント内の対応するテンプレートをナビゲーション。
- 名前付きテンプレートに呼出しを素早く追加
- 選択された名前付きテンプレートを変換のエントリポイントとしてセット

下のスクリーンショットにある**情報ウィンドウのXSLT タブ**を使って以下の操作を行うことができます：



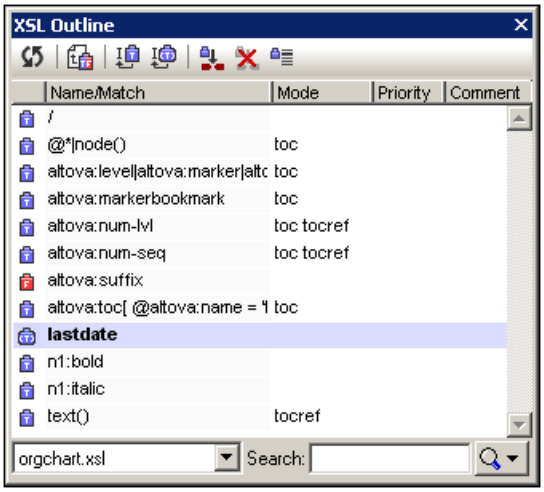
- インポート/インクルードされたファイルの位置といった、アクティブなXSLT ドキュメントに関連した全てのファイルに関する情報の閲覧。
- アクティブなXSLT ドキュメントを使って変換するXML ファイルのセット。更に、選択されたXML ファイルの検証を行うスキーマ (XSD/DTD) ファイルをセットすることもできます。
- 情報ウィンドウから関連するファイルを開く。

- XMLSpy プロジェクトにて関連するファイルを素早く管理。
- 関連する全てのファイルをZIP ファイルにまとめて、ユーザーの指定した場所に保存。

XSL アウトラインウィンドウならびに情報ウィンドウのXSLT タブに関する詳しい情報は、このセクション以下のサブセクションを参照ください。

5.3.1 XSL アウトラインウィンドウ

(以下のスクリーンショットにある) XSL アウトラインウィンドウでは、アクティブなXSLT ドキュメントにある全てのテンプレートと関数が表示されます。テンプレートは青色のアイコン (パラメーターを持たないテンプレートは で、パラメーターを含むテンプレートは) で表示されます。関数は赤の アイコンで表示されます。ウィンドウの左下にあるコンボボックスを使うことで、(i) (以下のスクリーンショットにあるように) アクティブなXSLT ドキュメントからのみ、または (ii) アクティブなXSLT ドキュメントとインポート/インクルードされた全てのスタイルシートから得られるテンプレートと関数を表示するよう指定することができます。



テンプレートには、(i) 名前付きテンプレートと、(ii) XPath 条件式にマッチするテンプレートの2つの種類があります。各テンプレートは以下の情報と共に表示されます：

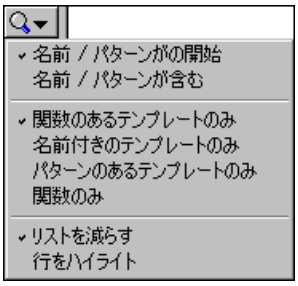
- (テンプレートにname 属性があれば) テンプレートの名前、そして (テンプレートにmatch 属性があれば) XPath 条件式。テンプレートが両方の属性を持っている場合、両方の情報が表示され、name 属性、namevalue 属性、そしてmatchvalue 属性の値と共に表示されます (次のスクリーンショットに太字で表示されているテンプレートに注目してください)。
- (モードがある場合) モード。テンプレートには1つ以上のモードがある可能性があることにご注意してください。(上のスクリーンショット参照)。
- (優先度がある場合) 優先度。
- (テンプレートまたは関数に先行するコメントがある場合) コメント

スタイルシートの関数は名前により表示されます。関数にはモードも優先度もありません。

オペレーション

XSL アウトラインウィンドウでは以下のオペレーションを実行することができます：

- *フィルタリング*：ウィンドウに表示されているリストを、以下のうちのどれかを表示するようにフィルタリングすることができます：(i) 全てのテンプレートと関数 (XMLSpy が起動されるたびに適用されるデフォルト設定)、(ii) 名前付きテンプレートのみ、(iii) XPath 条件式テンプレートのみ、(iv) 関数のみ。目的のフィルタリングを選択するには、ウィンドウ右下に位置している検索ボックスをクリックして、表示されたリスト (メニューの第2グループ) から目的のフィルターを選択します。選択されたフィルターはただちに適用され、設定が変更されるかXMLSpy を終了するまで有効になります。



- *並べ替えと検索* :列のヘッダーをクリックすることで、各列をアルファベット順に並べ替えることができます。再度クリックすることで並べ替えの順序を反転させることができます。この操作で列の並べ替えを行った後、リスト内で任意のアイテムを選択し、並べ替えられた列にて単語を入力すると、その単語を含むリスト内のアイテムがハイライトされます。この機能を使えば、目的の名前、条件式、モード、または優先度を持つテンプレートへ素早くたどり着くことができます。
- *検索* :(ウィンドウの右下にある)検索テキストフィールドに、検索を行ないたい名前またはXPath 条件式を入力することができます。入力中にも検索の結果は表示されます。(上のスクリーンショットにある)検索ボックスに表示されるリストから、以下のオプションを指定することができます:(i) 名前または条件式が検索語を含む(または検索語から始まる)。検索語から始まるオプションはXMLSpy が起動されるたびにデフォルトオプションとして指定されます。(ii) 検索結果をハイライトするか、それ以外を表示しないよう設定。表示を減らすオプションはXMLSpy が起動されるたびにデフォルトオプションとして指定されます。これらの選択は直ちに適用され、設定を変更するか、XMLSpy を終了するまで有効になります。
- *再ロード* :スタイルシートが変更された後、ウィンドウのツールバーにある **「同期」** アイコン によりXSL アウトラインを更新することができます。
- *アイテムへ移動* :XSL アウトラインウィンドウにてテンプレートまたは関数が選択されたとき、ウィンドウのツールバーにある **「定義へ移動」** アイコン をクリックすることで、ドキュメント内にあるテンプレートまたは関数がデザインビューでハイライトされます。もしくはエントリーをダブルクリックすることでも同様の効果を得ることができます。
- *名前付きテンプレートアクション* :名前付きテンプレートに関しては、2種類の操作を行うことができます。(i) (`xsl:call-template` を使った)名前付きテンプレートへの呼び出しをスタイルシート内のカーソル位置に挿入することができます。(ii) 名前付きテンプレートを変換のエントリポイントとしてセットすることができます。これらのオペレーションのコマンドは以下に記述されるようにツールバー上のアイコンから起動することができます。

変換のためのテンプレートモード

変換のための設定モードと呼ばれるツールバーのコンボボックスには、以下がリストされています:(i) スタイルシートの全てのモード (ii) XSLT 3.0 スタイルシートの場合 (デフォルトモードを選択する)空のエントリ (iii) `#unnamed` モード。ドロップダウンリストからモードを選択して、選択されたモードを変換のために設定します。(すべてのXSLT バージョンの)`#unnamed` モードは、`mode` 属性を持たないすべてのテンプレートモードに適用することができます。

XSLT 1.0 とXSLT2.0 スタイルシートの場合、デフォルトのモードは `#unnamed` です。ですから、空のエントリを選択することは、(`#unnamed` モードで、結果すべてのテンプレートに`mode` 属性無しで適用される)デフォルトのモードを選択することになります。

XSLT 3.0 スタイルシートでは、トップレベルの`xslt` 要素は、変換のためのデフォルトモードを持つ`default-mode` 属性を持つことができます。変換のためのモード設定コンボボックスでは、空のエントリ (デフォルトモード)が選択されていて、`default-mode` 属性で指定されているモードが変換モードとして使用されます。`#unnamed` モードがコンボボックスで選択されている場合、変換はすべてのテンプレートに名前のないモードで適用されます。すなわち、テンプレートにはモード属性がありません。

メモ: テンプレートにモードの値 `#all` を与え、全てのモードに適用することができます。

名前付きテンプレート

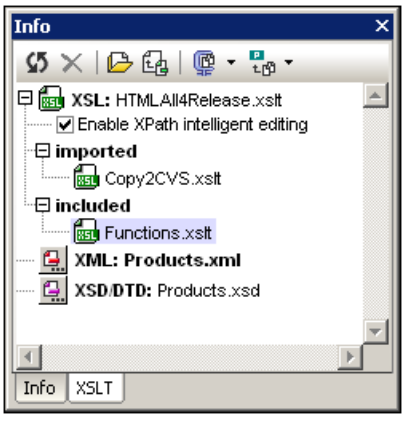
名前付きテンプレートが選択されていれば、以下のスクリーンショットにあるように、ウィンドウのツールバーにある名前付きテンプレートに関連したコマンドが選択可能になります。

ツールバーのコマンドは左から順に以下のようになります：

- *xsl::call-template の挿入* ：このコマンドはXSL アウトラインウィンドウにて名前付きテンプレートが選択された時に有効になります。このコマンドにより`xsl:call-template` 要素がスタイルシートのカーソル位置に挿入されます。スタイルシートに挿入される`xsl:call-template` 要素の`name` 属性値は、選択された名前付きテンプレートの`name` 属性の値になります。これにより`xsl:call-template` が選択された名前付きテンプレートへの呼び出しになります。
- *パラメーターを含むxsl:call-template の挿入* ：このコマンドはXSL アウトラインウィンドウにて選択された名前付きテンプレートが`xsl:param` 要素を持っている時に選択可能になります。**xsl:call-template** の挿入コマンドのように、`xsl:call-template` 要素が挿入されますが、このコマンドの場合、選択された名前付きテンプレートの`xsl:param` 子要素に対する`xsl:with-param` 子要素も一緒に挿入されます。挿入される`xsl:call-template` と その`xsl:with-param` 子要素の名前は、選択された名前付きテンプレートとその`xsl:param` 子要素の名前に対応します。
- *選択された名前付きテンプレートを変換のエントリポイントとしてセット* ：名前付きテンプレートが変換のエントリポイントとしてセットされている場合、XMLSpy で実行される変換はこの名前付きテンプレートから開始されます。XSL アウトラインウィンドウでは、エントリポイントとなる名前付きテンプレートが（このセクションの始めにあるスクリーンショットにあるように）太字で表示されます。
- *名前付きテンプレートを変換のエントリポイントから解除* ：名前付きテンプレートが変換のエントリポイントとしてセットされると選択可能になります。
- *変換のエントリポイントとしてセットされた名前付きテンプレートへ移動* ：名前付きテンプレートが変換のエントリポイントとしてセットされると選択可能になります。変換のエントリポイントとしてセットされた名前付きテンプレートとは別のアイテムにXSL アウトラインウィンドウのフォーカスがある場合、このアイコンをクリックすることでXSL アウトラインウィンドウにある名前付きテンプレートがハイライトされ、素早くアクセスすることができるようになります。

5.3.2 情報ウィンドウ

XMLSpy でXSLT ドキュメントがアクティブになっている時に表示される情報ウィンドウのXSLT タブには、アクティブなXSLT ドキュメントに関連するインポート/インクルードされた全てのファイルが表示され、XSLT ドキュメントを使った変換に、ソースXML ドキュメントとして使用することのできるXML ファイルを簡単に選択することができます。

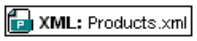


情報ウィンドウのXSLT タブには以下のファイルが表示されます：

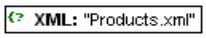
- *XSLT ファイル* ：上のスクリーンショットにあるように、インポート/インクルードされた全てのXSLT ファイルが表示さ

れます。ファイル上にマウスを移動させると、各ファイルの位置がポップアップで表示されます。インポートまたはインクルードされたファイルをダブルクリックするか、選択して情報ウィンドウツールバーにある **開く** アイコンをクリックすることで、ファイルが新しいウィンドウで開かれます。ツールバーにある **「include/Import 位置へ移動」** アイコンにより、アクティブな XSLT ドキュメントの include/import 宣言部がハイライトされます。

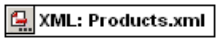
- *XML ファイル* 変換を行うためにXML ファイルをアクティブな XSLT スタイルシートに割り当てることができます。割り当てられたXML ファイルの場所は、ファイル上にマウスを移動すると表示されるポップアップで確認することができます。XML ファイルを指定した後、そのXML ファイル上でメニューコマンド **「XML/XQuery | XSL 変換 (F10)」** を選択することで、定義済みのXSLT スタイルシートを使った変換が実行されます。XML ファイルは **XML** アイコンをクリックすることで選択することができます（選択されたファイルは太字で表示されます）。その他にもプロジェクトプロパティダイアログ（*XSL ファイルの XSL 変換*）や XSLT ドキュメントの処理命令（`<? altova_samplexml "products.xml"?>`）からXML ファイルを割り当てることができます。どのような方法でも、以下のようなアイコンと共にXML ファイルは Info ウィンドウに表示されます：

  プロジェクトプロパティダイアログにより割り当てられている

  XSLT ドキュメントの処理命令により割り当てられている

  XML アイコンをクリックし、ファイルの参照を行うことにより割り当てられている（太字で表示されます）

  同じファイルに対して2つ以上の方法で割り当てがなされている場合、選択の優先順位は：(i) プロジェクト、(i) 処理命令、(ii) ユーザーによる参照、となります。アイテムをダブルクリックするか、選択後に **開く** ツールバーアイコンをクリックすることでXML ファイルを開くことができます。

- *XSD/DTD ファイル*：選択されたXML ファイルがスキーマ（XML スキーマまたはDTD）への参照を含む場合、XSD/DTD エントリーに、そのスキーマファイルが表示されます。その他にも、（XML ファイルの場合と同じように）プロジェクトプロパティダイアログ（*検証*）からスキーマファイルを選択したり、**「XSD/DTD」** アイコンをクリックし、目的のスキーマファイルを選択することもできます。スキーマファイルがプロジェクトプロパティダイアログから選択された場合、エントリーの隣にプロジェクトアイコンが表示され、それ以外の場合には（**「XSD/DTD」** アイコンによりスキーマが選択された場合は太字で表示される）ファイルエントリーの隣に **「XSD/DTD」** アイコンが表示されます。スキーマファイルが2つ以上の方法で割り当てられている場合、選択の優先順位は：(i) プロジェクト、(i) ユーザーによる参照、(ii) XML ドキュメント内の参照、となります。ファイルにマウスを移動させると、割り当てられたXSD ファイルの位置がポップアップで表示されます。ダブルクリックまたは選択後に **開く** ツールバーアイコンをクリックすることでスキーマファイルを開くことができます。

注意：XML またはXSD/DTD ファイルがプロジェクトプロパティダイアログから選択された場合、この選択を解除するにはプロジェクトプロパティダイアログを開き、そこで選択を解除する必要があります。選択が **「XML」** または **「XSD/DTD」** アイコンにより選択がなされた場合、設定を解除するには、ファイルを選択して、情報ウィンドウツールバーにある **「クリア」** アイコンをクリックしてください。

オプション

XPath インテリジェント編集：XML ファイルが割り当てられている場合、XML ドキュメントの構造が走査され、インテリジェント XPath 編集が要素と属性にも拡張されます。XSLT ドキュメント内のXPath 条件式を入力することができる場所では、使用可能な要素と属性がポップアップで表示されます。このオプションはデフォルトで有効になっており、チェックボックスのチェックを外すことでXPath インテリジェント編集を無効にすることができます。設定はファイルが閉じられるときに各XSLT ファイルに個別に保存され、ファイルが開かれる際にその設定が読み込まれます。

ツールバーアイコン

（以下のスクリーンショットにある）情報ウィンドウツールバーアイコンを、左から右に説明します：

- *再ロード*: XSLT ドキュメント内で行われた修正を反映するよう情報ウィンドウを更新します。
- *XML/XSD割り当てをクリア*: XML または XSD/DTD アイコンを使ってなされた XML や XSD/DTD の割り当てを解除します。解除を行うファイルを選択して、このアイコンをクリックしてください。
- *ドキュメントを開く*: 選択されたドキュメントを開きます。
- *import/include 位置へ移動*: インポートまたはインクルードされたファイルが選択されているとき、このアイコンをクリックすることで XSLT ドキュメント内の関連するインポートまたはインクルード宣言がハイライトされます。
- *全てのドキュメントを圧縮*: 情報ウィンドウに表示されている全てのドキュメントを圧縮して、ユーザーの定義した場所に保存します。または、ドキュメントを選択して、アイコンのドロップダウンメニューから **「選択したドキュメントを圧縮」** コマンドを選択することで、選択されたファイルだけを圧縮することができます。
- *プロジェクトに全てのファイルを追加*: 現在のプロジェクトに全てのファイルを追加します。または、ドキュメントを選択し、アイコンのドロップダウンメニューから **「プロジェクトに選択したファイルを追加」** コマンドを選択することで、選択したファイルだけをプロジェクトに追加することができます。

5.4 XSL スピードオプティマイザー

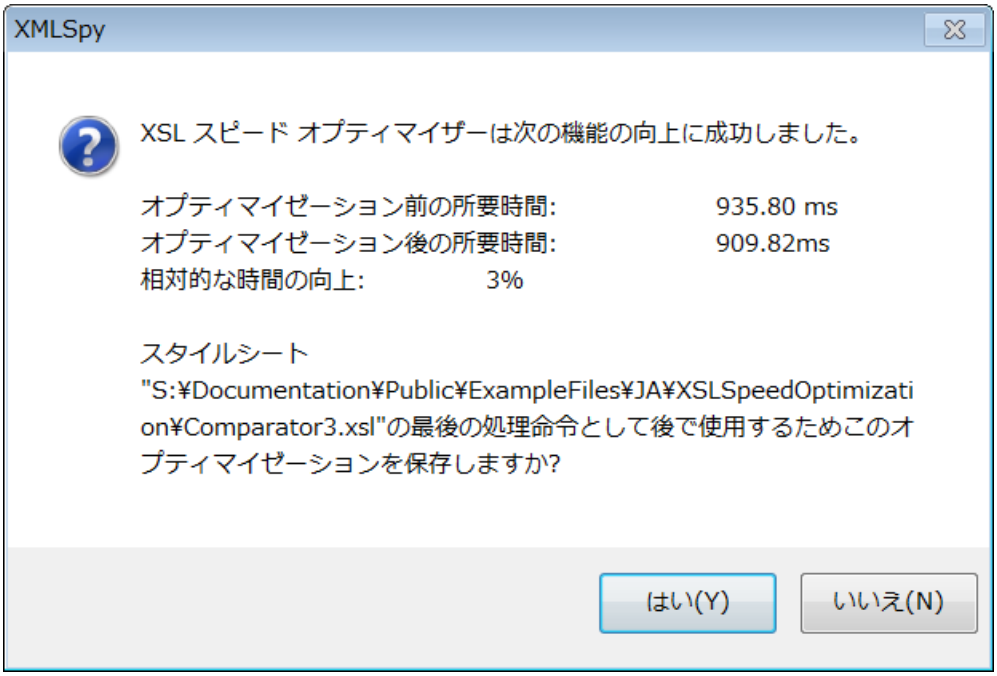
(このセクションではオプティマイザーと称される)XSL スピードオプティマイザーにより、XSLT スタイルシートを最適化することができ、結果変換が速く行うことができます。オプティマイザーは、XML データセット(1つ以上のXML ドキュメント)に対してXSLT スタイルシートを実行することにより作動し、スタイルシートのパフォーマンスを分析します。最適化の方略はこの分析から割り出され、処理情報としてスタイルシートの最後に(XSLT スタイルシートと共に保存されます。スタイルシートは、後により早い変換のために使用されます。

XSLT スタイルシートの最適化

XSLT スタイルシートを最適化するには、XSLT スタイルシートに加え、サンプルデータセットとしてサーブするXML ドキュメントが必要です。XSLT スタイルシートが的確に分析されるために、データセットは、のすべてのXSLT スタイルシート部分のために十分な大きさである必要があります。

オプティマイゼーションを以下のとおり行います:

1. XSLT スタイルシートまたはXML ドキュメントがアクティブな場合、メニューコマンド「**XSL/XQuery | XSL スピードオプティマイザー**」をクリックまたは、メインツールバー内のオプティマイザーのアイコンをクリックします。
2. XSLT またはXML ドキュメントがアクティブかどうかにより、XML ドキュメントまたはXSLT スタイルシートを選択するようにプロンプトされます。「**OK**」をクリックすると、分析が開始されます。(XSLT またはXML ドキュメントが既にドキュメント内にXML 割り当てまたはXSLT 割り当てを持つ場合、このステップはスキップされます。分析開始されると、コマンドが直接呼び出されます。)
3. オプティマイゼーション分析に失敗した場合、この結果についてのメッセージが表示されます。(オプティマイゼーション分析の失敗の理由は、下に説明されています)分析に成功すると、分析の結果を表示するダイアログがポップアップします(下のスクリーンショット)。



ダイアログは、オプティマイゼーション(命令)を(スタイルシートの最後に処理命令として)XSLT スタイルシートに保存するオプションを与えます。「**はい**」をクリックしてオプティマイゼーションを保存、「**いいえ**」をクリックして、オプティマイゼーションを破棄します。オプティマイゼーションが保存されるたびに、以前に保存されたオプティマイゼーションを上書きします。

最適化された *スタイルシート*をより速い変換のために使用することができます。

オプティマイゼーション分析の失敗の理由

XSL スピード オプティマイザーが *オプティマイゼーション*を引き起こせない場合、 1つまたは複数の以下の理由の可能性があります :

- XSLT スタイルシート既に効率化されており、 最適化される必要がない場合。
- 提出された、XML データセットが最適化するにはお小さすぎる可能性があります。より大きなデータセットで再試行してください。
- *オプティマイゼーション*のしきい値が高すぎる可能性があります。 *オプションダイアログの*XSL *スピード オプティマイザー タブ*のしきい値を変更してください。以下を参照。
- この特定のXSLT 構造のための*オプティマイゼーション*構造を *オプティマイザー*で使用することができません。Altova サポートに問い合わせてください。

XSL スピードオプティマイザーの設定

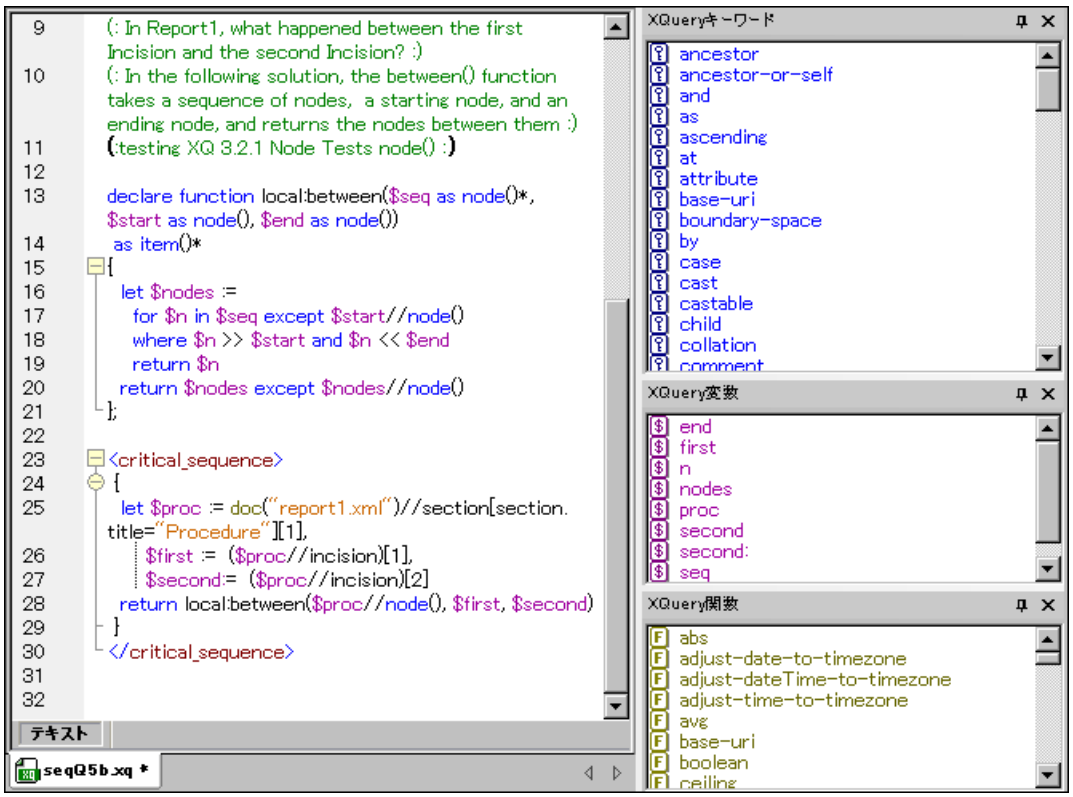
*オプティマイザー*のための設定は、 *オプションダイアログの*XSL *スピード オプティマイザー タブ*で行うことができます (**ツール | オプション...**、 *下のスクリーンショット*)。

XSLT スタイルシート内の単一のXSLT 命令ためのしきい値が*オプティマイザー*のために指定されることができます。値の範囲を 0.1% から、変換の総時間の 99% に設定します。命令がしきい値として指定されたものよりも長く実行にかかる場合、分析が開始されます。それ以外の場合、分析は行われません。*オプティマイゼーション*分析に失敗した場合、その理由としては、 *オプティマイザー*設定内のしきい値の設定が高すぎた可能性があります。しきい値を低くすることを考えてください。

6 XQuery

Altova ウェブサイト: XQuery エディター

XQuery および XQuery Update ドキュメントはテキストビューで編集することができます。このビューにより入力ヘルパーや、構文による色分け、インテリジェント編集といった機能を使用することができます。(*スクリーンショット参照*) また、XQuery の検証を行ない XQuery ドキュメント内にあるコードを内蔵の Altova XQuery エンジンを使って実行することができます。



注意: XQuery ファイルはテキストビューでのみ編集することができます。その他のビューは XQuery に対してご利用になれません。

XQuery と XQuery Update ファイルの関連付け

XMLSpy 内では、XQuery と XQuery Update ドキュメントは 2 つの異なるドキュメントのタイプとして認識されます。通常 XQuery ドキュメントの拡張子は `.xq` で、XQuery Update ドキュメントのファイル拡張子は `.xqu` です。追加ファイル拡張子をこれらのファイルタイプと関連付けることができ、ファイルタイプの関連付けも、オプションダイアログのファイルタイプタブ (**ツール | オプション | ファイルタイプ**) で変更することができます。

XQuery 実行または XQuery アップデートにより、**「XQuery/ Update 実行」**コマンドが実行されるため、ファイル拡張子のドキュメントタイプの関連付けは重要です。

このセクション

このセクションは以下のように構成されています：

- XQuery ドキュメントの編集
- XQuery 評価
- XQuery の検証
- XQuery/Update 実行
- XQuery Update Facility
- XQuery とXML データベース

他の関連した機能と情報:

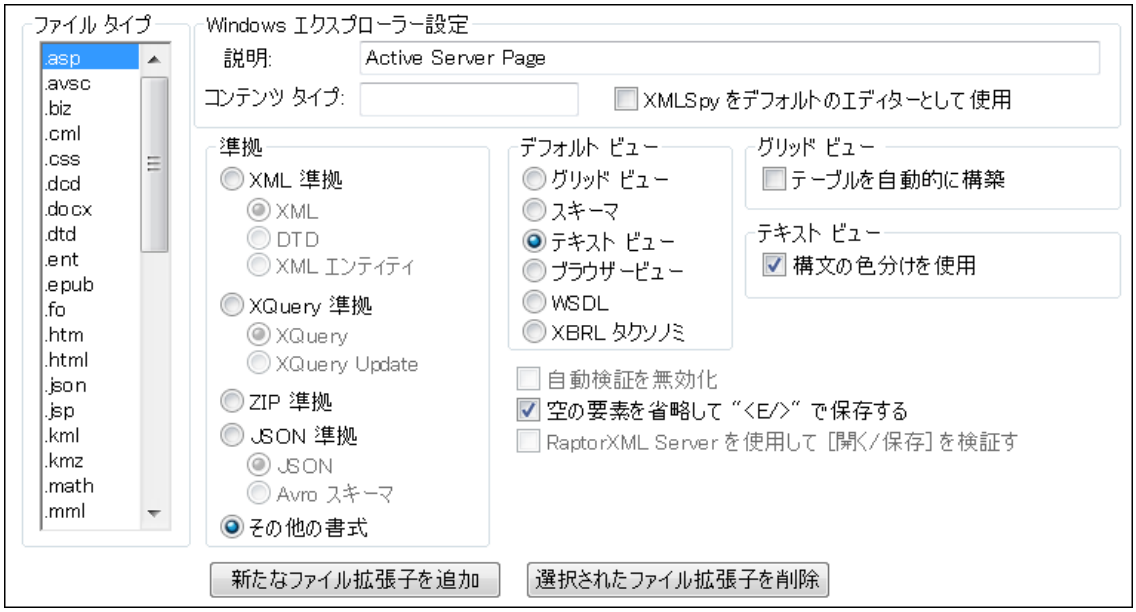
- XSLT/XQuery デバッガーとプロファイラー
- XQuery エンジンの実装
- 出力ウィンドウ:XPath/XQuery
- ツール| オプション| ファイルタイプ
- ツール| オプション| XQuery

コマンドラインやバッチ処理のための RaptorXML

XMLSpy のプロジェクト機能を使うことで、GUI からバッチ処理を行うことができます。より柔軟性のある処理をお望みの場合、XMLSpy にも搭載されているAltova XQuery エンジンを含むAltova のRaptorXML 製品を使用することができます。AltovaXML はコマンドラインやバッチ処理からXQuery の実行を行う理想的なツールになります。

6.1 XQuery ドキュメントの編集

XMLSpy 内では、XQuery とXQuery Update ドキュメントは 2 つの異なるドキュメントのタイプとして認識されます。ドキュメントタイプ (XQuery または XQuery Update) オプションダイアログのファイルタイプ内のファイル拡張子に割り当てられます。 (**ツール | オプション | ファイルタイプ** 下のスクリーンショット)。XQuery または XQuery Update タイプのファイルが、XMLSpy で開かれた場合、そのファイルのテキストビューの XQuery 編集機能を使用することができます。



XMLSpy でXQuery とXQuery Update として現在定義されているファイル拡張子

XQuery	`.xq` `.xql` `.xqr` `.xquery`
XQuery Update	`.xqu`

メモ: このセクションで説明される編集機能はXQuery とXQuery Update ドキュメント内の説明と同様です。

XQuery /Update 実行

XSL/XQuery | XQuery/ Update 実行 のGUI コマンドは、自動的に選択された、XQuery ファイルのXQuery 実行 またはXQuery /Update を実行します。詳細に関しては、**『XQuery /Update 実行』** を参照してください。

6.1.1 XQuery ドキュメント

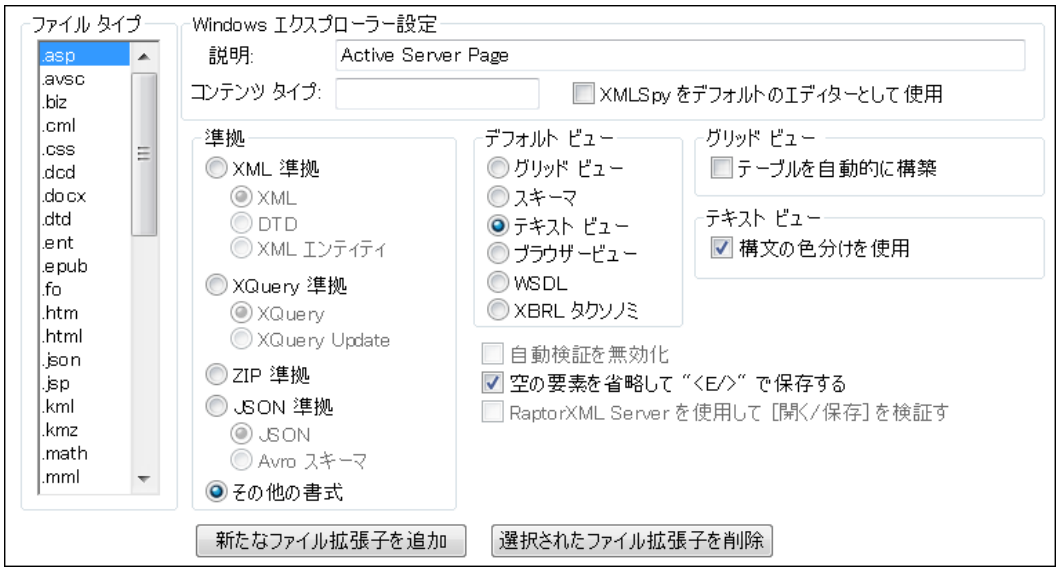
XQuery またはXQuery Update ドキュメントはXQuery またはXQuery Update に準拠している場合、自動的に、テキストビューのXQuery 編集モードで開かれます。ファイルはオプションダイアログ (**ツール | オプション | ファイルタイプ** 下のスクリーンショット) のファイルタイプタブの特定のドキュメント型に準拠するように定義されています。

XMLSpy でXQuery とXQuery Update として現在定義されているファイル拡張子

XQuery	`.xq` `.xql` `.xqr` `.xquery`
XQuery Update	`.xqu`

XQuery 準拠のファイル拡張子を追加する
他のファイル拡張子を持つドキュメントをXQuery 準拠として認識させるには：

1. **「ツール | オプション」**を選択。以下のスクリーンショットにある）オプションダイアログが表示されます。



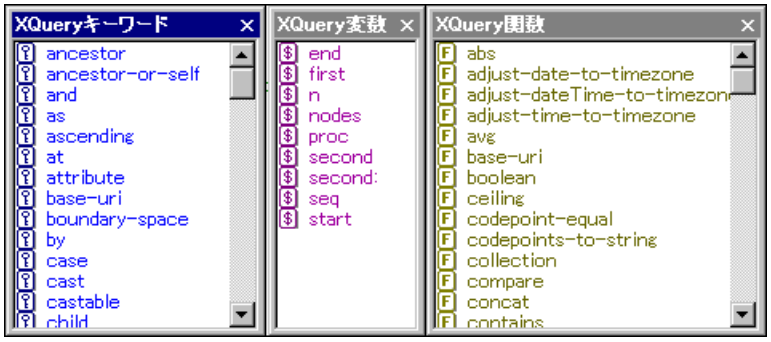
2. **「ファイルタイプ」**タブを選択。
3. **「新たなファイル拡張子を追加」**をクリックして、ファイルタイプのリストに新たな拡張子を追加。
4. **「準拠」**にて、**「XQuery 準拠」**を選択。*XQuery* または *XQuery Update* を選択する。

次の設定も行ってください：

- *説明*：`XML Query 言語` または `XQuery Update Facility`
- *コンテンツタイプ*：`text/xml`
- XMLSpy をXQuery ファイルの編集を行うデフォルトエディターとして使用する場合、**「XMLSpy をデフォルトのエディターとして使用」**チェックボックスのチェックを入れてください。

6.1.2 XQuery 入力ヘルパー

テキストビューのXQuery モードには3つの入力ヘルパーがあります（XQuery キーワード（青色）、XQuery 変数（紫色）、XQuery 関数（オリーブ色））。



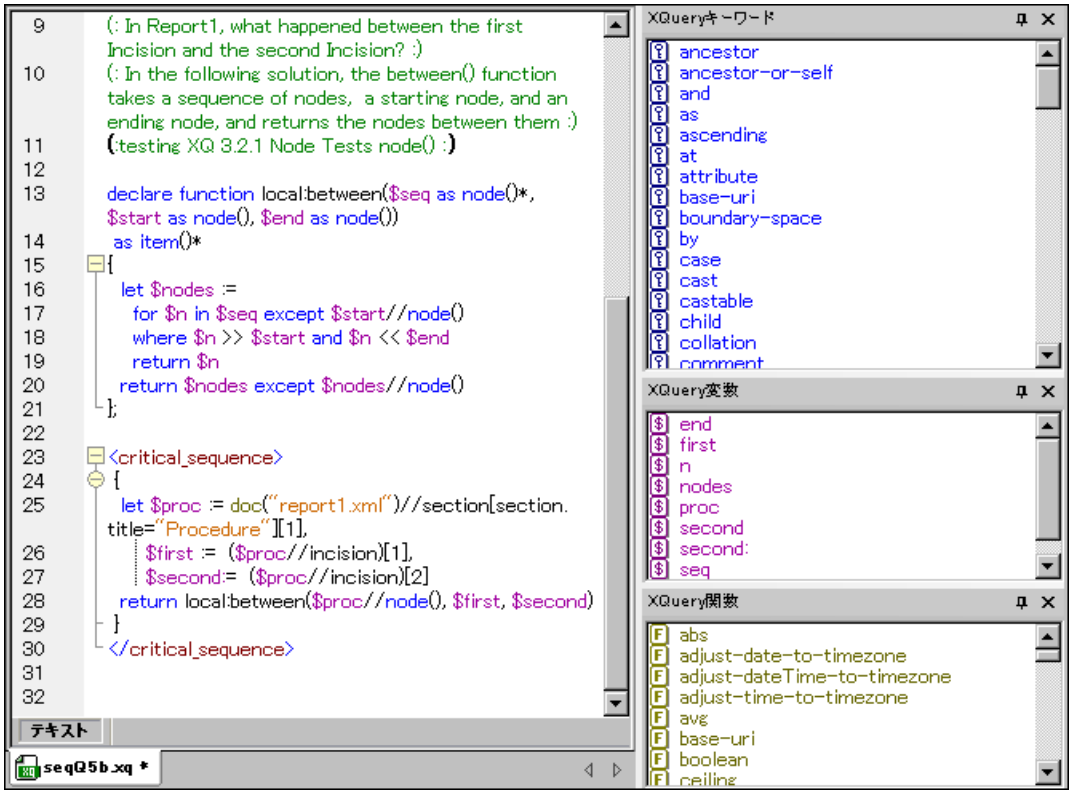
以下の点に注意してください：

- 入力ヘルパー内にあるアイテムの色は、それぞれテキストにて使用されている構文の色に対応しています。これらの色を変更することはできません。
- 表示されているキーワードと関数はAltova XQuery エンジンによりサポートされているものです。
- 関数はXQuery ドキュメント内で定義されたものです。テキストビューにて $ と文字列が入力された場合、その文字列が変数入力ヘルパーに表示されます。入力された変数が既に存在する変数名と同一の場合、新たに入力された変数名が入力ヘルパーから削除されます。
- 入力ヘルパーにてナビゲーションを行うには、入力ヘルパー内のアイテムをクリックして、スクロールバー、マウスホイール、またはpage-down や page-up キーを使って、リストを移動することができます。

入力ヘルパーにて表示されているアイテムをドキュメント内に挿入するには、挿入を行ないたい場所にカーソルを移動させ、アイテムをダブルクリックします。XQuery では文字列によってキーワードと関数の両方を表す場合があります (例 'empty、unordered または except)。これらの文字列は (たとえ関数入力ヘルパーからその名前の関数を選択しても) 常に (青色の) キーワードとして入力されます。関数が青色で表示された場合、その名前の後に続くかっこにより、関数と認識させることができます。

6.1.3 XQuery の構文による色分け

XQuery ドキュメントはXQuery コードとXML コードを内部に含むことができます。このセクションではデフォルトのXQuery 構文による色分けについて説明します。XQuery ドキュメント内のXML コードに対して使用される構文色分けは、通常のXML ドキュメントに対するものと変わりません。構文による色分けの色は XQuery ならびにXML コードに対して全てオプションダイアログ (**ツール| オプション**) のテキストフォントタブにて設定することができます。以下のスクリーンショットの例にあるように中かっこ { } を使うことで、XML 要素内部にXQuery コードを埋め込めることにご注意してください。



XQuery モードのXQuery コード内では、以下の構文色分けが使用されます：

- **(: コメントは区切り文字も含めて緑色で表示されます :)**
- XQuery キーワードは青色で表示されます：**キーワード**
- XQuery 変数は、ドル記号も含めて紫色で表示されます：**$start**
- かっこを除くXQuery 関数はオリーブ色で表示されます：**function()**
- 文字列はオレンジ色で表示されます：**"Procedure"**
- パス表現など、その他のテキストは、以下の下線部分にあるように黒色で表示されます：

```
for $s in doc("report1.xml")//section[section.title = "Procedure"]
return ($s//incision)[2]/instrument
```

これらの色やフォントプロパティはオプションダイアログ（**ツール | オプション**）のテキストフォントタブで変更することができます。

注意：上のスクリーンショットではコメント部分にあるかっこのペアの1つが太字の黒色で表示されていますが、これはかっこマッチング機能によるものです（XQuery インテリジェント編集を参照ください）。

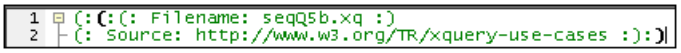
6.1.4 XQuery インテリジェント編集

テキストビューのXQuery モードでは以下のインテリジェントな編集機能が提供されます。

- かっこマッチング
- キーワード
- 変数
- 関数
- ビジュアルなガイド

かっこマッチング

かっこマッチング機能により、かっこのペアとなる開始ならびに終了かっこがハイライトされ、かっこ内に含まれるコンテンツを簡単に確認することができます。この機能は、特にXQuery コメント内にてかっこがネストされているような状況で役に立ちます。



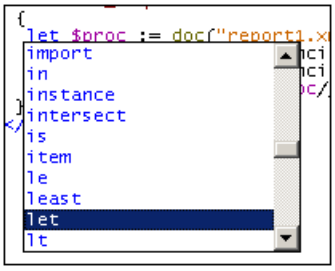
かっこマッチングはカーソルが（開始または終了）かっこの直前または直後に位置しているときに有効になります。かっこは対となるかっこと一緒に太字に黒色でハイライトされます。上のスクリーンショットにあるカーソルの位置に注目してください。

かっこマッチングは、丸かっこ()、角かっこ[]、そして中かっこ{}で有効になります。例外は、XML タグに使用される不等号記号 <> です。

メモ：XML 要素の開始または終了かっこのすぐ内にカーソルがあると、両方のかっこがハイライトされます。**「Ctrl」+E** のキーを押すことで、対となるペアがハイライトされます。**「Ctrl」+E** を連続して押下することで、カーソルの位置を開始ならびに終了かっこ間で切り替えることができます。この機能を使うと、ドキュメントのナビゲーションを素早く行うことができます。

キーワード

XQuery キーワードはクエリー表現で使用される命令で、青色で表示されます。カーソルをキーワードの内部、直前、または直後に移動させることでキーワードを選択できます。キーワードが選択された状態で、**「Ctrl」+「Space」**を押すことにより、キーワードのリストがポップアップで表示されます。リストをスクロールし、目的のキーワードをダブルクリックすることで、選択されていたキーワードを置き換えることができます。

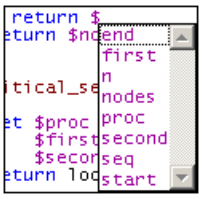


上のスクリーンショットでは、カーソルが`let` キーワード上に位置されており、ポップアップにあるキーワードをダブルクリックすることで、`let` キーワードを置き換えることができます。

変数

変数名の前には`$` 記号が配置され、紫色で表示されます。このインテリジェント編集機能の仕組みはキーワードの編集機能のそれと似たようなものです。ドキュメント内の変数リストが表示されるポップアップへアクセスするには2つの方法があります：

- `$` 文字を入力後に`Ctrl+Space` を押下
- 変数を選択後に`Ctrl+Space` を押下（`$` 文字の直後、変数名の内部や直後にカーソルを位置させることで、その変数が選択された状態になります）

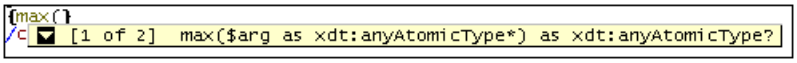


$ 記号の直後に変数を挿入する、または選択された変数を置き換えるには、ポップアップメニューから目的の変数をダブルクリックしてください。

関数

キーワードならびに変数の場合と同様に、 (オリーブ色で表示される) 関数を選択するか、 **「Ctrl」 + 「Space」** を押下することで内蔵の関数がポップアップメニューにより表示されます。関数内、または関数名の直前、直後にカーソルを位置させることで、関数が選択された状態になります。関数名の直後にあるかっこ内部にカーソルを位置させても選択されている状態にはなりません。ポップアップメニュー内の関数名をダブルクリックすることで、選択された関数名をポップアップメニューの関数に置き換えることができます。

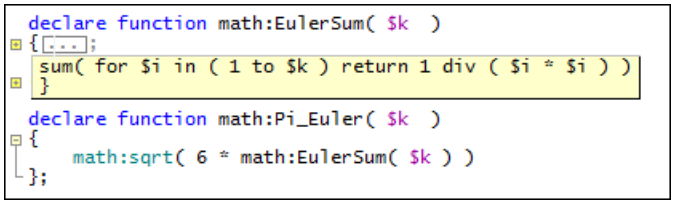
以下のスクリーンショットにあるように関数の開始かっこの直後にカーソルをおき、 **「Ctrl」 + 「Space」** を押下することで、その関数のヒントを表示することができます。標準 XQuery 関数でしかヒントは表示され無いことに注意してください。



ヒントの先頭にある下向きの矢印は、異なる引数や戻り値の型を持った同名の関数が存在することを表しています。矢印をクリックすることで次の関数を表示することができ、クリックを繰り返すことでその名前の関数を全て表示させることができます。他にも **「Ctrl」 + 「Shift」 + 「Up」** または **「Ctrl」 + 「Shift」 + 「Down」** キーの組み合わせで、同様の操作を行うことができます。

ビジュアルなガイド

テキストの折りたたみ (またはソースのおりたたみ) は XQuery 中かっこ、XQuery コメント、XML 要素、XML コメントにより有効化され、これらのノードを展開または折りたたむ機能を意味します。このようなノードは、+/- サインによりソース折りたたみ余白内で表示されます (*下のスクリーンショット参照*)。余白はテキストビュー設定ダイアログにより切り替えのオンとオフを行うことができます。ノードが折りたたまれると、視覚的に省略記号により表示されます (*下のスクリーンショット参照*)。マウスのカーソルが省略にポイントされると、ポップアップで折りたたまれたノードの内容が表示されます (*スクリーンショット参照*)。内容がポップアップでの表示に大きすぎる場合、ポップアップのしたの部分に省略記号で表示されます。



テキストツールバー内の **「全ての折りたたみを切り替える」** アイコン は、**全ての**ノードを展開された書式でまたはトップレベルのドキュメント要素まで折りたたみ表示します。

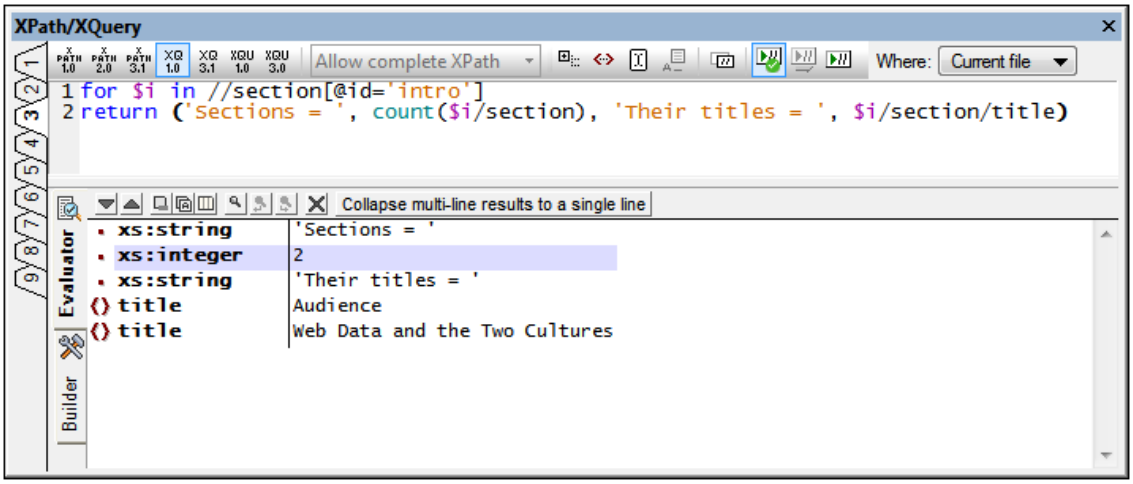
ノードの +/- アイコンをクリックすることにより以下のオプションを使用することができます :

Click [-]	ノードを折りたたみます

Click [+]	子孫ノードが展開されて表示されるか、ノードが折りたたまれた前の状態に従い折りたたまれ、ノードが展開されます。
Shift+[−]クリック	すべての子孫ノードを折りたたみ、展開フォーム内でクリックされたノードを残します。
Ctrl+ [+]クリック	クリックされたノードとそのすべての子孫ノードを展開します。

6.2 XQuery 評価

XQuery 式は、XPath/XQuery 出力ウィンドウ内の1つまたは複数のドキュメントに対して評価されることができます (*下のスクリーンショット*)。



以下をおこなってください:

1. ウィンドウの上のペインにXQuery 式 を入力します。
2. *Where* コンボボックス内から (*上のスクリーンショット参照*) クエリされるXML ドキュメントの場所を選択します。オプション: (i) 現在のファイル; (ii) ファイルを開く; (ii) プロジェクト; (v) フォルダー.
3. **「XPath/XQuery 式の評価」 (F5)** をクリックします。XML ファイルに対して式が評価されます。指定された (*Where*) ロケーションが、1つ以上のXML ファイルを含む場合、XML ファイル内で、式に一致するデータ構造またはコンテンツが検索されます。使用可能なすべての一致は、下のペイン内に表示されています。

上のスクリーンショット内で、属性 `@id='intro'` を持つセクション要素のためにクエリが作成されます。クエリは、この `intro` セクションサブセクションの数量とタイトルを返します。

詳細に関しては、出力ウィンドウ:XPath/XQuery および アップデートのプレビューと適用 を参照してください。

6.3 XQuery 検証

XQuery またはXQuery Update ドキュメントを検証するには以下を行います :

1. XQuery ドキュメントをアクティブなドキュメントにします。
2. **「XML | 検証」**を選択、 **「F8」**キーを押す、または **検証」**ツールバーアイコンをクリックします。

検証ツールバーアイコン

XQuery 構文に対してドキュメントが検証されます。

6.4 XQuery/ Update 実行

XQuery または XQuery Update ドキュメントは以下の方法で実行することができます：

- XQuery または XQuery Update ドキュメントがアクティブな場合。
- XML ドキュメントがアクティブな場合。

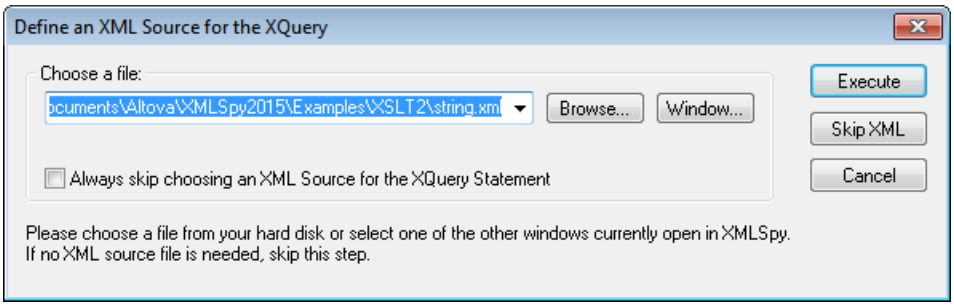
メモ: ドキュメントが XQuery ドキュメントまたは XQuery Update ドキュメントであるかは、ドキュメントのファイル拡張子によって決定されます。XMLSpy は、ファイルの種類の関連付けをオプションダイアログのファイルタイプタブによる定義に従い認識します。 (**ツール | オプション | ファイルタイプ**)。

メモ: XQuery Update には、XPath/XQuery 出力ウィンドウに アップデート式を入力することができます。アップデートが受け入れられた場合、アップデートを適用して、アップデートされたファイルを保存します。詳細に関しては、XQuery Update Facility と アップデートのプレビューと適用を参照してください。

XQuery または XQuery Update ドキュメントを実行する

XQuery または XQuery Update ドキュメントを *'XQuery / XQuery Update ドキュメントをアクティブにする'* で実行する場合は以下を行います。

1. XQuery または XQuery Update ドキュメントをアクティブなドキュメントにします。
2. **'XSL/XQuery | XQuery/ Update 実行'** を選択または、コマンドのツールバーアイコンをクリックします。これによりXQuery のためのXML ソースを定義するダイアログが表示されます (下のスクリーンショット)。
   XQuery/ Update 実行ツールバーアイコン



3. XML ファイルを参照するか、XML ソースの選択をスキップします。

通常、XQuery ドキュメントは、単一のドキュメントに関連付け**られていません**。これは、XQuery 式が doc() 関数を使用して、XML ドキュメントをいくつも選択することができるからです。XMLSpy 内では、しかしながら、個別の XQuery ドキュメントを実行する前に、実行のためのソースXML ドキュメントを選択することができます。この場合、選択されたXML ソースのドキュメントノードは、XQuery ドキュメントのルートレベルで使用することのできるコンテキストメニューから開始します。行頭スラッシュから開始するパスは、コンテキストアイテムのドキュメントノードとしてこのドキュメントノードと共に解決されます。

メモ: **'XQuery/ Update'** 実行コマンドは、プロジェクトウィンドウアイテムのコンテキストメニューから使用することができます。

実行 /アップデートの結果

- *XQuery 実行* :実行の結果は一時ファイルとして生成され、希望するファイル形式やファイル拡張子で任意の場

所に保存することができます。

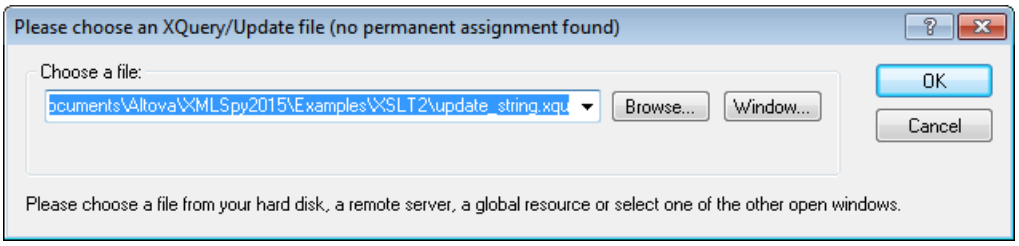
- *XQuery アップデート:* アップデートはファイルに保存されるか、アップデートされたファイルが開かれると、プレビューし保存または閉じることができます。どちらのアクションを実行するか指定することができます。これはオプションダイアログの *XQuery* タブ(**「ツール| オプション| XQuery」**)で行うことができます。

アクティブな XML ドキュメントでの実行

XQuery または XQuery Update ドキュメントをアクティブなドキュメントで実行するには以下を行います。

1. XML ドキュメントをアクティブにします。
2. **「XSL/XQuery | XQuery/ Update 実行」**を選択するか、 ツールバーアイコンをクリックします。これによりXQuery のための XML ソースを定義するダイアログが表示されます。**「XSL/XQuery | XQuery/ Update 実行」**を選択または、コマンドのツールバーアイコンをクリックします。これによりXQuery のための XML ソースを定義するダイアログが表示されます (下のスクリーンショット)。

   XQuery/ Update 実行ツールバーアイコン



3. XQuery または XQuery Update ファイルを参照して **「OK」**をクリックします。

実行 /アップデートの結果

- *XQuery 実行 :* 実行の結果は一時ファイルとして生成され、希望するファイル形式やファイル拡張子で任意の場所に保存することができます。
- *XQuery アップデート:* アップデートはファイルに保存されるか、アップデートされたファイルが開かれると、プレビューし保存または閉じることができます。どちらのアクションを実行するか指定することができます。これはオプションダイアログの *XQuery* タブ(**「ツール| オプション| XQuery」**)で行うことができます。

バックマッピング

バックマッピング機能が有効化されていると、XSLT 変換が実行され、結果ドキュメントが元の XQuery+XML ドキュメントにバックマッピングされます。結果ドキュメント内のノードをクリックすると、**XQuery 命令** とその特定の結果フラグメントを生成した**XML ソースデータ**がハイライトされます。また、XQuery 命令、または、XML データノードをクリックすると、他の2つのドキュメント内の対応するノードがハイライトされます。詳細に関しては、**XSL/XQuery | XSLT/XQuery バックマッピングの有効化** を参照してください。

XQuery 変数

Altova XQuery エンジンを使用している場合、XQuery 変数を便利な GUI ダイアログ内に補完することができます。XMLSpy を使ったXQuery ドキュメントの実行時に、全ての保管された変数はXQuery ドキュメントに渡されます。より詳しい情報については、XSLT パラメーター /XQuery 変数コマンドの記述を参照ください。

Altova XQuery エンジン

Altova XQuery エンジンの実装やXQuery ファイルの処理に関する詳細については、XQuery エンジンの実装を参照ください。

6.5 XQuery Update Facility

XQuery Update Facility は、ドキュメントの部分を変更することのできるXQuery 言語の拡張機能です。通常のXQuery 実行では、ドキュメント全体が再生成され、そして、元の場所に保管される必要があります。これはドキュメントの小さいパートのみが変更される場合、非効率になる可能性があります。Update Facility を使用することにより変更された部分のみを更新することができます。

XQuery Update Facility は、XQuery 1.0 およびXQuery 3.1 の拡張機能として次の仕様内で説明されています：

- XQuery Update Facility 1.0 (2011 年 3月 17日版W3C 勧告)
- XQuery Update Facility 3.0 (2013 年 1月 8日版W3C 作業ドラフト)

XMLSpy 内のXQuery Update Facility

以下の要点は、XQuery Update がXMLSpy 内でどのように作動するかを説明しています：

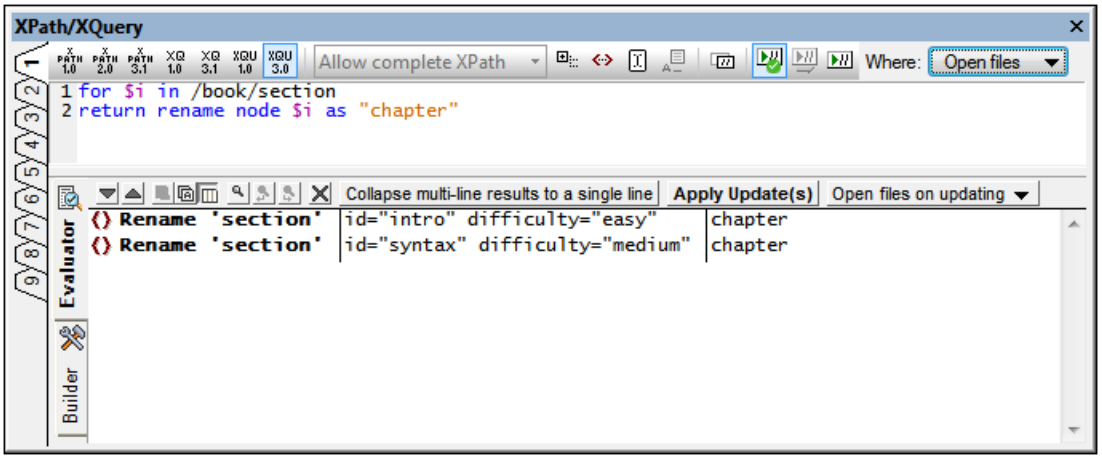
- アップデートはアップデート式により行われます。例えば、アップデート式は、XML ドキュメント内のノードの名前の変更を指定することができます：
  ```
  rename node /documents/doc-01 as "document-01"
  ```
- 実際のところは、複数のアップデート式が単一のXQuery Update ドキュメントに入力されます。
- アップデートドキュメント内の各 アップデート式が実行されると、結果はすぐに適用されませんが、*保留中の アップデート リスト (PUL)*に追加されます。 この結果、PUL は すべてのアップデート式の結果を含みます。すべてのupdates in PUL のすべてのアップデートは、すべてに一度に適用されます。
- XMLSpy 内で、PUL アップデート以下の 2つの方法で適用されます：
  (i) GUI 内でユーザーによりプレビューする利点は、プレビューは、予期しない結を果表示します。XQuery/Update 実行コマンド または XPath/XQuery 出力ウィンドウ 内のXQuery アップデート式評価中でプレビューを使用することができます。プレビューオプションに関しては、関連する箇所で説明されています。
  (ii) 直接 そしてユーザーによる中断なし。利点は、アップデートが警告なしにユーザーによる中断なしに実行されます。アップデートの直接のアプリケーションはXQuery/Update 実行コマンド またはXPath/XQuery 出力ウィンドウ 内でXQuery アップデート式の評価を実行中に使用することができます。(プレビューなしの)直接的アップデートのアプリケーションがが直接のアップデートオプションで使用可能な場合、オプションに関する情報は関連する箇所で説明されています。

XMLSpy は、アクティブなXML ドキュメントに対して使用されたアップデート式の効果をプレビューすることのできる、パワフルなXQuery Update プレビュー機能を与えます。この機能については、アップデートのプレビューと適用のセクションで説明されています。

6.5.1 アップデートのプレビューと適用

XQuery Update を使用して、XML ドキュメントを修正する場合、XML ドキュメントに適用する前にプレビューして、変更されたドキュメントを保存することができます。

XPath/XQuery 出力ウィンドウ (*下のスクリーンショット*)内に 1つまたは複数の アップデート式を入力することができます、そして下のペインに表示された**保留中のアップデートリスト (PUL)** 内でアップデートをプレビューすることができます (*下のスクリーンショット参照*)。PUL が希望する状態の場合、ドキュメントのアップデートを適用し、変更されたドキュメントを保存することができます。PUL に変更を付け加える場合は、変更を適用しない、または保存しないことを選択できます。

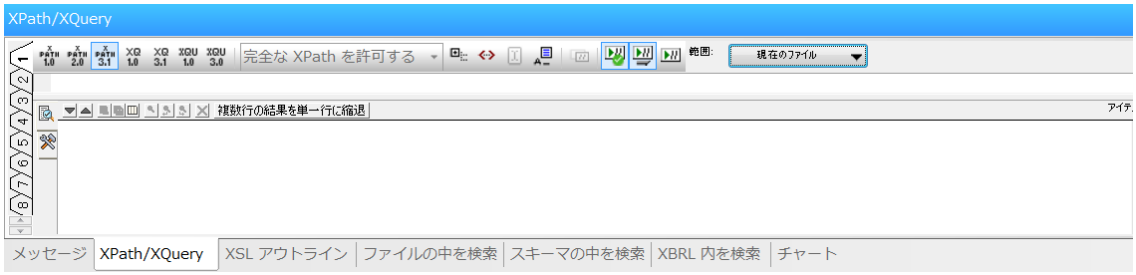


アクティブな XML ファイルのために PUL を作成するには、次を行います：

1. XPath/XQuery 出力ウィンドウのツールバーから (*上のスクリーンショット*) **XQU 1.0** または **XQU 3.0** アイコン (それぞれ XQuery Update 1.0 または XQuery Update 3. 0 のため) に選択します。
2. 1 つまたは複数の アップデート式をウィンドウの上のペインに入力します。アップデート式と構文の説明に関しては、操作と構文のアップデートのセクションを参照してください。
3. ツールバーの スキャンロケーションコンボボックスから、アップデートのためにスキャンされるロケーションを選択します：
   ファイルを開く: XMLSpy で現在開かれているすべてのファイルがスキャンされます。
   現在のファイル: 現在アクティブな ファイルのみがスキャンされます。スキャンのために選択されたロケーションが*現在のファイル*の場合、 **「XPath/入力中に XQuery 式の評価する」** ツールバーアイコンが有効化されます。
   プロジェクト: 現在アクティブなプロジェクトがスキャンされます。
   フォルダー: スキャンするフォルダーを選択することができます。
4. アップデート式 を実行して、PUL を表示するには、 **「XPath/XQuery 式の評価」** ツールバーアイコンをクリックします。

XPath/XQuery 出力ウィンドウ ツールバー

以下の XPath/XQuery 出力ウィンドウツールバー コマンド (*下のスクリーンショット内で赤枠の中*) を使用することができます。

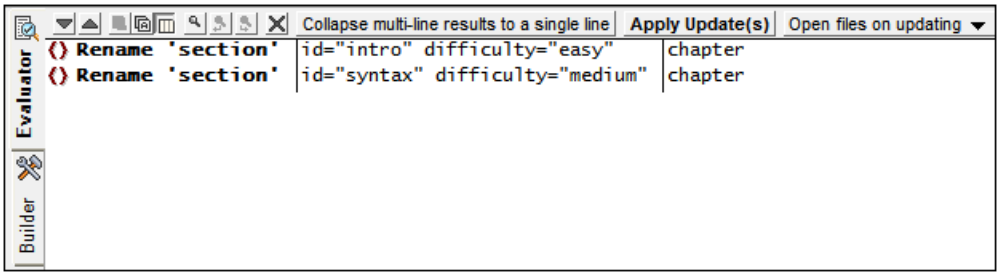


- ヘッダーの表示は、アップデートロケーションのファイル名の表示の切り替えを行います。
- 完全な結果の表示は、ノードコンテンツ全体またはノードの属性のみの表示の切り替えを行います。
- XPath/XQuery オリジンの設定は、アクティブなファイル内のカーソルロケーションを相対的な XPath 式のオリジンとして設定します。式 は、このノードに対して相対的に評価されます。
- 手動でノードのロケーターパス式を入力するかわりに、XMLSpy が入力を行うことができます。これは以下のように行います：(i) XPath 式にロケーターパスを入力する箇所にカーソルをポイントします、(i) ターゲットするノードの開始タグ内にカーソルをポイントします、(ii) 式内にロケーターパスを入力するために、編集フィールドアイコン の

現在の選択のXPath のコピーをクリックします。
- XML の検証 ファイルは、XML ファイル検証の切り替えを行います。
- 入力中のXPath/XQuery 式の評価コマンドは、解析オプションが現在のファイルに設定されていると有効化されます。コマンドがオンに切り替えられている場合、アップデート式が入力されている場合でもPUL は、生成されます。
- XPath/XQuery 式の評価 コマンドは、アップデート式を評価し、PUL を生成します。
- ロケーションのスキャン コンボボックス オプションは、上で説明されています。

保留中のアップデートリスト (PUL) ペイン

PUL ペインは、実行されるすべてのアップデートを表示します。ヘッダーの表示オプションがウィンドウの ツールバー内で切り替えられている場合、ターゲットファイルのロケーションが表示されます。PUL の表示は、3つの垂直のセクションに分割されています (*下のスクリーンショット参照*): (i) 実行するアップデート アクション、 (i) ターゲットノードのコンテンツはアップデートされます。(ii) アップデートアクションの結果



次のPUL ペイン ツールバー コマンドを使用することができます :

- **次と前** アイコンは、現在選択されているメッセージの次および前のメッセージをそれぞれ選択します。
- **ラインのコピー コマンド** は、選択されたラインおよびすべてのラインをクリップボードにそれぞれコピーします。
- **検索** コマンドは、PUL ペイン内のテキストを検索します。
- **クリア** コマンドは、PUL ペイン内のすべてのラインを削除します。
- **複数ラインの結果のおりたたみ** コマンドは、複数のラインのアップデートアイテムを単一のラインに折りたたみます。
- **アップデートの適用** コマンドは、保留中の アップデートをターゲットの場所に適用します。アップデートする際に、アップデートは、ファイルに保存、または、アップデートされたファイルが表示されます (そして、手動で 保存されるかを決定することができます)。次のオプションを参照してください。
- アップデートファイル ディレクトリ コンボボックスにより以下を選択することが許可されます (i) ファイルがダイアログを表示せずにディスクにアップデート、または (i) アップデートされたファイルが開かれそしてアクティブになるか。後者のオプションが選択された場合、非 アクティブなターゲット ファイルはアクティブに設定されます。変更された ドキュメントを保存または保存しないオプションがあります。

メモ: 1 つまたは複数のファイルが直接ディスクにアップデートされた場合、変更されたファイルのリストが表示されます。リストの各 ラインは、ファイルのロケーションおよびファイルへのクリックすることのできるリンクを表示します。

6.5.2 Update オペレーションと構文

XQuery Update Facility では以下のオペレーションを使用することができます :

- 1つまたは複数のノードの削除 する
- 1つまたは複数のノードを指定されたノードの前、後、または内部に挿入する
- ノードの名前の変更をする

- アイテムのシーケンスのノードと置換える
- アイテムのシーケンスの文字列の値とノードの値を置換 える

これらのオペレーションのキーワード及び構文はこのセクションのサブセクションで説明されています。

ノードの削除

説明と構文
1つ以上のノードを削除します。

```
delete node nodeSequence
delete node nodeSequence
```

詳細

- *nodeSequence* 式は削除するノードのシーケンスを返します。選択されたノードは削除されるためマークされます。
- 単一 **ノード** または複数の **ノード** が使用されることができます。*nodeSequence* 内のアイテムの数に対するコレスポンデンスは必要ありません。

サンプル

```
for $i in /book/section return
delete nodes $i/@id
```

ノードの挿入

説明と構文
1つまたは複数のノードを指定されたターゲットノードの前、後、または内部に挿入する。

```
insert (node|nodes) items into targetNode
insert (node|nodes) items as first into targetNode
insert (node|nodes) items as last into targetNode
insert (node|nodes) items before targetNode
insert (node|nodes) items after targetNode
```

詳細

- *items* 式は、アイテムのシーケンスを返さなければなりません。キーボード **node|nodes** が使用されていますが、*items* は、非ノードアイテムのシーケンスである場合があります。
- *targetNode* 式は、単一のターゲットノードを指す必要があります。
- **into** キーボードが使用されていますが、*targetNode* は要素ノードまたはドキュメント要素ノードである必要があります。
- キーフレーズ **as first** または **as last** が使用されていますが、挿入は、最初または最後の子です。

- キーワード into は単独で仕様され、属性は既存の属性に追加され、要素は最初の子として挿入されます。
- キーワード before または after が使用されており、 *targetNode* はあらゆる型であることができます。
- 属性が挿入される場合、既存の属性の名前を重複してはなりません。

サンプル

```
for $i in /book/section return
insert nodes (attribute id { 'somevalue' }, <newelement>some content including the numbers
"{1 to 3}"</newelement>)
into $i
```

ノードの名前の変更

説明と構文
要素、属性、または指示ノードの名前を変更する。

```
rename node targetNode as name
```

詳細

- 要素、属性、または処理命令であることのできる *targetNode* 式は単一ターゲットノードを指す必要があります。
- *name* 式は、QName または文字列を評価する必要があります。
- QName が作成されると、必須名前空間がローカルで宣言されます。

詳細

```
rename node /book/title as 'header-1'

rename node /book/title as QName("http://www.altova.com/xquf","header-1")
```

ノードの置換

説明と構文
アイテムのシーケンスとノードを置き換えます。

```
replace node targetNode with items
```

詳細

- *targetNode* 式は、単一のターゲットノードを指す必要があります。
- *items* 式はアイテムのシーケンスを返す必要があります。このシーケンスはターゲットノードを置き換えます。

- 属性ノード以外、ターゲットノードはどの型のシーケンスとも置換えられることができます。
- 属性ノードは属性ノードと置き換えられます。下のサンプルを参照してください。

サンプル

```
replace node //hr with '<line/>'

for $i in //@height return
replace node $i with (attribute line-height{'12pt'})
```

ノードの値の置換

説明と構文
アイテムのシーケンスの文字列の値とノードの値を置き換えます。

```
replace value of node targetNode with items
```

詳細

- *targetNode* 式は、単一のターゲットノードを指す必要があります。
- *items* 式はアイテムのシーケンスを返す必要があります。
- ターゲットノードのコンテンツは *items* 式により返されたシーケンスの文字列の値と置き換えられます。これは、ターゲットノードはテキストノード1つのみを含むことを意味します。

サンプル

```
for $i in //title return
replace value of node $i with ('Draft Title')
```

fn:put 関数

fn.put 関数はXQuery 内蔵関数ライブラリの拡張子としてQuery Update Facility 1.0 により与えられています。(**fn:** このセクションの名前空間のプレフィックスは、以下の名前空間にバインドされているものと見なされます :`http://www.w3.org/2005/xpath-functions`.)

```
fn:put($node as node(), $uri as xs:string) as empty-sequence()
```

関数は、ドキュメントまたは要素を`$uri` により指定された場所に保管します。ファイルシステムまたはデータベースなどの外部ストレージシステムのリソースを作成するために呼び出されます。`fn:put` の外部効果は、XQuery のドメイン外で起こるため、実装定義済みです。目的は、`fn:put` がドキュメントノードにより呼び出され、エラーが指摘されなかった場合、次のクエリは、同じURI を持つ`fn:doc` を呼び出して保管されたドキュメントにアクセスすることができます。

詳細は 仕様 を参照してください。

6.6 XQuery と XML データベース

XQuery ドキュメントを使えば、XML データベース (XML DB) への問い合わせを行うことができます。この XQuery 機能は、現在 IBM DB2 データベースでしかサポートされておりません。XQuery を使った XML DB への問い合わせの仕組みは、(i) XQuery エンジンに対して DB 内部の XML へ問い合わせを行う通知と、(ii) DB 内部にある XML データへのアクセス、により成り立っています。

この仕組みを実装するには、以下のようなステップを踏む必要があります。各ステップの詳細を以下に記します：

1. XQuery ドキュメントの設定 を行い、XQUERY キーワードをドキュメントの最初に挿入することで XML DB の問い合わせを行うようにする。
2. アクティブな XQuery ドキュメントに対して、情報ウィンドウから DB のサポートを有効にして、クイック接続ダイアログを使って DB へ接続する。
3. XQuery ドキュメントにて、DB 固有の XQuery 拡張を使用することで、DB データにアクセスし、XQuery オペレーションも行えるようにする。
4. XQuery ドキュメントの実行を XMLSpy から行う。

XML DB の問い合わせを行うよう XQuery ドキュメントを設定する

XQuery ドキュメントを使って XML DB の問い合わせを行うには、XQuery ドキュメントを開き (または新しい XQuery ドキュメントを作成し)、XQUERY キーワードを、ドキュメントの最初に入力します。以下の例を参照ください。

```
XQUERY (:Retrieve details of all customers :)
declare default element namespace "http://www.altova.com/xquery/databases/db2";
<a> {db2-fn:xmlcolumn("CUSTOMER.INFO")}</a>
```

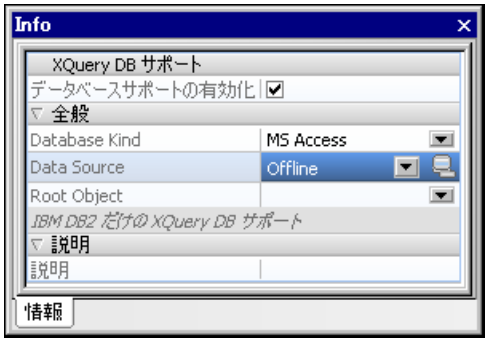
ドキュメントがオプションの xquery version 表現を使用する場合でも、XQUERY が必要です：

```
XQUERY xquery version "1.0"; (:Retrieve details of all customers :)
declare default element namespace "http://http://www.altova.com/xquery/databases/db2";
<a> {db2-fn:xmlcolumn("CUSTOMER.INFO")}</a>
```

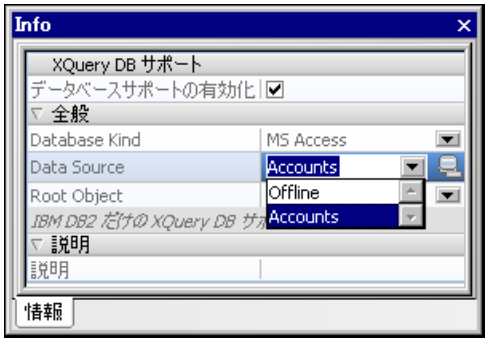
メモ: XMLSpy の内蔵 XQuery エンジンが XQUERY キーワードを読むことにより、XML DB がアクセスの対象になることが示唆されます。その結果、XQUERY キーワードを含む XQuery ドキュメントを、XML DB を含まない XML ドキュメントに対して実行してもエラーとなります。

DB サポートを有効化し、DB へ接続する。

下のスクリーンショットの様に、情報ウィンドウにあるデータベースサポートを有効化チェックボックスにチェックを入れることで、XQuery ドキュメントの DB サポートが有効になります。DB サポートは、各 XQuery ドキュメントに対して有効にする必要があり、XQuery ドキュメントを開き直したときにも再度設定する必要があることにご注意してください。



情報ウィンドウにて DB のサポートを有効にすると、接続ウィザードが起動され、データベースへの接続を行うことができます。現在のところ、IBM DB2 データベースのみがサポートされています。データベースへの接続方法については[データソースへ接続]、のセクションにて記述されています。データソースへの接続が既に存在する場合、これらの接続が（以下のスクリーンショットにある）情報ウィンドウの Data Source コンボボックスに表示され、データソースのリストに表示されているアイテムの1つを、アクティブな XQuery ドキュメントのデータソースとして選択することができます。情報ウィンドウでは、Root Object コンボボックスからルートオブジェクトを選択することもできます。



情報ウィンドウにある アイコンをクリックすることで、（DB への接続を行う）接続ウィザードにいつでもアクセスすることができます。

メモ: XQuery ドキュメントを閉じると、DB への接続も同様に切断されます。その後 XQuery ドキュメントを開く際に、DB への接続を再度行う必要が生じます。

IBM DB2 固有の XQuery 言語拡張

XQuery ドキュメントでは IBM DB2 データベースからデータを取得するために、2つの IBM DB2 固有の関数を使用することができます：

- `db2-fn:xmlcolumn` により、列の検索やフィルタリングを行うこと無く XML 列全体を取得することができます。
- `db2-fn:sqlquery` により、SQL `SELECT` ステートメントをベースにした値の取得を行ないます。

これらの関数を使用して取得された XML データは、標準的な XQuery コンストラクトにより開くことができます。以下の例を参照ください。

db2-fn:xmlcolumn：この関数の引数は大文字/小文字で区別される文字列リテラルで、テーブル内にある XML 列の参照に使用されます。文字列リテラル引数は XML タイプの修飾された列名でなければなりません。列内にある全ての XML データがシーケンスとして関数から返され、ここでは検索条件などは適用されません。以下の例では、`CUSTOMER` テーブル内の `INFO` (XML) 列のデータ全てが、トップレベルの `<newdocelement>` 要素内に戻されます：

```
XQUERY (:Retrieve details of all customers :)
declare default element namespace "http://www.altova.com/xquery/databases/db2";
<newdocelement> {db2-fn:xmlcolumn("CUSTOMER.INFO")}</newdocelement>
```

取得されたデータは XQuery コンストラクトを使って絞り込むことができます。以下の例では、`CUSTOMER` テーブルにある `INFO` (XML) 列から取得されたデータが、XQuery コンストラクトによりフィルタリングされ、トロント (Toronto) にある顧客データのみが取得されるようになっています。

```
XQUERY (:Retrieve details of Toronto customers :)
declare default element namespace "http://www.altova.com/xquery/databases/db2";
<newdocelement> {db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo[addr/city="Toronto"]}</
newdocelement>
```

メモ: 上の例では、各セルにある XML ファイルのドキュメント要素は `customerinfo` であり、`db2-`

fn:xmlcolumn により得られたXML シーケンスのルートノードはcustomerinfo ノードの上にあるabstract ノードと見なされます。

db2-fn:sqlquery:. この関数はSQL SELECT ステートメントを引数として受け取り、その結果をXML 値のシーケンスとして戻します。取得されたシーケンスはXQuery コンストラクトにより絞り込むこともできます。以下の例では、CUSTOMER テーブルにあるINFO 列のデータのうち、CID フィールドの値が1000 から1004 間にある行が取得されます。SQL ステートメントでは大文字/小文字の区別は行われませんが、XQuery では異なるものとして扱われることに注意してください。

```
XQUERY (:Retrieve details of customers by CID:)
declare default element namespace "http://www.altova.com/xquery/databases/db2";

<persons>
  {db2-fn:sqlquery("SELECT info FROM customer WHERE CID>1000 AND CID<1004")/
  <person>
    <id>{data(@CID)}</id>
    <name>{data(name)}</name>
  </person>}
</persons>
```

上のXQuery ドキュメントは以下のような出力を生成します：

```
<persons xmlns="http://www.altova.com/xquery/databases/db2">
  <person>
        <id>1001</id>
        <name>Kathy Smith</name>
  </person>
  <person>
        <id>1002</id>
        <name>Jim Jones</name>
  </person>
  <person>
        <id>1003</id>
        <name>Robert Shoemaker</name>
  </person>
</persons>
```

以下の点に注意してください：

- デフォルトの要素名前空間宣言（デフォルトのelement namespace）はXQuery 全体に対して適用され、XML ドキュメントや新たな要素の構築時のナビゲーションに利用されます。これは、XQuery セレクターname が、<default-element-namespace>:name に拡張され、構築された要素（例：persons）がデフォルトの名前空間にあることを意味します。
- SQL SELECT ステートメントでは大文字/小文字の区別が行われません。
- SELECT ステートメントのWHERE 句は、データベースアイテムを参照するもので、アクセスされているXML ファイルのノードを参照するものではありません。
- db2-fn:sqlquery 関数の後にある "/" は戻されたシーケンスの最初のアイテムを表わしており、このアイテムがコンテキストノードになります。

XQuery の実行

XQuery ドキュメントを実行するには、XSL/XQuery メニューにあるXQuery を実行コマンドを選択します。他にも、Alt+F10 を押下したり、XQuery を実行アイコン をクリックすることで、同様の操作を行えます。実行の結果は新しいドキュメント内に表示されます。

7 XSLT/ XQuery デバッガー とプロファイラー

XMLSpy には、ドキュメントを速く作成する手助けをするXSLT/XQuery デバッガー とXSLT/XQuery プロファイラー が搭載されています。

これらの 2 つの機能に関しては、このセクションのサブセクションで説明されています。

7.1 XSLT と XQuery デバッガー

Altova Web サイト: XSLT デバッガー、XQuery デバッガー

XSLT and XQuery Debuggerを使ってXSLT スタイルシートやXQuery ドキュメントの試験やデバッグを行うことができます。XSLT and XQuery Debuggerのインターフェースにはお互いに連動するXSLT/XQuery ドキュメント、結果ドキュメント、そしてソースXML ドキュメントのビューが搭載されており、XSLT/XQuery ドキュメントをステップ毎に実行することができます。対応する出力もステップごとに生成され、ソースXML ファイルが表示されていれば、XML ファイル内の対応する箇所が各ステップでハイライトされます。インターフェース内にあるウィンドウにはデバッグ情報が表示されます。

XSLT and XQuery Debuggerは常に**デバッガーセッション**内で起動されます。デバッガーセッションは以下のうちのどれかです：

- 内蔵のAltova XSLT 1.0 エンジンを使用するXSLT 1.0
- 内蔵のAltova XSLT 2.0 エンジンを使用するXSLT 2.0
- 内蔵のAltova XSLT 3.0 エンジンを使用するXSLT 3.0
- 内蔵のAltova XQuery 1.0 エンジンを使用するXQuery 1.0
- 内蔵のAltova XQuery 3.1 エンジンを使用するXQuery 3.1

メモ: XQuery Update Facility 1.0 とXQuery Update Facility 3.0 は現在 XSLT and XQuery Debugger でサポートされていません。

開かれるデバッガーセッションの種類は、デバッガーセッションが開かれたドキュメント (*アクティブドキュメント* または *アクティブファイル*) の種類により自動的に決定されます。XSLT デバッガーセッションはXSLT ファイルから開かれ (どのバージョンかは、XSLT スタイルシート(XSLT 1.0 には `"1.0"` 、XSLT 2.0 には`"2.0"`、XSLT 3.0 には `"3.0"`)) の `xsl:stylesheet` や`xsl:transform` 要素にある`version` 属性の値により変化します)、XQuery デバッガーセッションはXQuery ファイルから開かれます。*アクティブファイル*がXML ファイルであれば、XML ファイル上で動作させるのがXSLT かXQuery かにより、デバッガーセッションの種類が決定されます。前者の場合選択はスタイルシートがXSLT 1.0、XSLT 2.0、またはXSLT 3.0 スタイルシートであるかにより決定されます。

この情報は下のテーブルにまとめられています。

アクティブなファイル	関連したファイル	デバッグ セッション
XSLT 1.0	XML; (必須)	XSLT 1.0 (内蔵の Altova XSLT 1.0 エンジン使用)
XSLT 2.0	XML; (必須)	XSLT 2.0 (内蔵の Altova XSLT 2.0 エンジン使用)
XSLT 3.0	XML; (必須)	XSLT 3.0 (内蔵の Altova XSLT 3.0 エンジン使用)
XQuery 1.0	XML; (オプション)	XQuery 1.0 (内蔵の Altova XQuery 1.0 エンジン使用)
XQuery 3.1	XML; (オプション)	XQuery 3.1 (内蔵の Altova XQuery 3.1 エンジン使用)
XML	XSLT 1.0, 2.0, または 3.0,または XQuery 1.0 または 3.1; (必須)	XSLT 1.0、XSLT 2.0、 3.0または XQuery 1.0 または 3.1。XSLT スタイルシートの `xsl:stylesheet` または `xsl:transform`) 要素にある `version` 属性の値により、XSLT スタイルシートを決定。

3つのAltova エンジンに関する詳細は、付録内のXSLT とXQuery エンジン情報を参照してください。

Altova RaptorXML 2017 を使用した XSLT とXQuery タスクの自動化

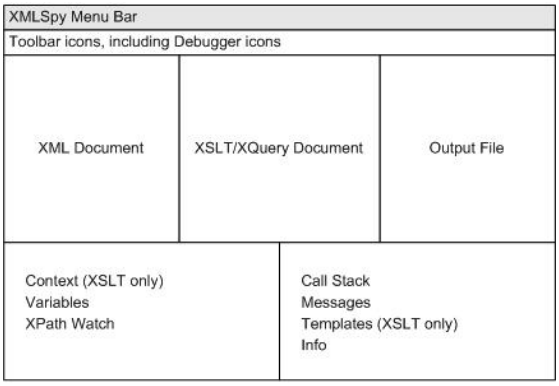
Altova RaptorXML Server は、Altova の XML バリデーター、XSLT 1.0、XSLT 2.0、XSLT 3.0、そして XQuery 1.0 と 3.1 エンジンを搭載した無料アプリケーションです。コマンドライン、COM インターフェース、Java プログラム、そして .NET アプリケーションから XML ドキュメントの検証を行ったり、XSLT スタイルシートを使った XML ドキュメントの変換や、XQuery ドキュメントの実行を行うことができます。

RaptorXML を使えば XSLT ならびに XQuery 処理を自動化することができます。例えば、RaptorXML を呼び出して、XML ドキュメントのセットを変換したり、XQuery ドキュメントのセットを実行するバッチファイルを作成することができます。詳細は RaptorXML Server ドキュメントを参照ください。

7.1.1 メカニズムとインターフェース

XSLT ならびに XQuery ファイルのデバッグにて使用される XSLT and XQuery Debugger の概要を以下に示します。(**「XSL/XQuery | デバッガーの開始 実行」**コマンドにより両方とも開始されるものですが) デバッグセッションとデバッガーは異なるものであることに注意してください。まずはデバッグセッションを開始して、デバッガーにより XSLT または XQuery ドキュメントをステップ実行している必要があります。

- **「XSL/XQuery | デバッガーの開始 実行」**コマンドによりデバッグセッションを開きます。アクティブファイルを元に適切なセッション (XSLT 1.0、XSLT 2.0、XSLT 3.0、XQuery 1.0 または XQuery 3.1) XSLT と XQuery デバッガーを参照ください)。XSLT and XQuery Debugger はテキストビューならびにグリッドビューでのみ動作します。アクティブドキュメントのビューがテキストビューでもグリッドビューでもない状態でデバッグセッションを開始した場合、XSLT and XQuery Debugger のデフォルトビューであるテキストビューにビューを変更する許可を求められます。デバッガー設定ダイアログにて、常に許可を行うよう設定することもできます。
- **「XSL/XQuery」**メニューにある **「ステップイン」**、**「ステップアウト」**、そして **「ステップオーバー」**コマンドを使用することで XSLT または XQuery ドキュメントをステップごとに実行することができます。ここでデバッガーの開始 実行コマンドをクリックすると、ブレークポイントに到達するまで変換または実行を行ないます。ブレークポイントが 1 つもセットされていない場合、デバッグの結果を表示すること無く、1 ステップで全ての変換や実行を行なうことになります。このセッションに XML ファイルが割り当てられている場合、XML ファイルの対応する箇所がハイライトされます。更に、対応するステップにより生成された結果は、結果ファイルに蓄えられ、ステップごとに結果ドキュメントが更新されます。この方法により、XSLT や XQuery ファイルが各ステートメントごとに行う動作を確認、分析することができます。



上で示されている 3 つのドキュメント (XML、XSLT/XQuery、出力) の代わりに、2 つのドキュメント (XSLT/XQuery、出力)、またはどれか 1 つのドキュメントビューを選択することもできます。

- デバッグセッションが開いている間、インターフェースの情報ウィンドウには変換 実行に関する様々な情報が表示されます (変数、XPath Watch、コールスタック メッセージなど)。
- デバッグセッションが開いている間、デバッガーを停止 (デバッグセッションは停止していないことに注意) して、ドキュメントを修正することもできます。XMLSpy で使用することのできる編集機能は、デバッグセッション中にファイルを編集する際にも全て使用することができます。デバッガーが停止している間でも XSLT and XQuery Debugger のインターフェースは開かれたままになり、情報ウィンドウに表示されている情報の参照を行うことができます。デバッグセッションにてデバッガーを停止させた後、同じデバッグセッションにて (XSLT/XQuery ドキュメントの初めから) デ

バッガーを再度開始することもできます。

- XSLT ファイルにブレークポイントをセットすることで、選択された箇所で割り込みを発生させることができます。この機能により、XSLT または XQuery ドキュメント内のステップを手動で個別に実行する必要が無くなり、デバッグセッションの効率を大幅に向上させることができます。
- XSLT にはトレースポイントをセットすることもできます。トレースポイントがセットされた命令にて処理が実行された場合、その命令の値が出力されます。
- デバッグセッションを停止します。この操作によりXSLT and XQuery Debugger インターフェースが終了し、元のXMLSpy 環境へ切り替わります。情報ウィンドウにある情報は利用できなくなります。セットされたブレークポイントとトレースポイントに関する情報は、そのファイルが閉じられるまで保持されたままになります（従って、新たなデバッグセッションが開始された場合、新たに表示されるデバッグセッションにもブレークポイントやトレースポイントは表示されます）。

メモ デバッガーアイコンのあるデバッガーツールバーは、デバッグセッションの開始時に自動的に表示されます。

7.1.2 コマンドとツールバーアイコン

XSL/XQuery メニューやツールバーアイコンからデバッグコマンドへアクセスすることができます。デバッガーアイコンは、デバッグセッションが開始されたときに、ツールバーに自動的に表示されます。以下にデバッガーアイコンを示します。

デバッガーの開始 実行 (Alt+F11)
XSLT/XQuery ドキュメントの処理を開始、または最後まで継続します。ブレークポイントがセットされている場合、そこで処理が停止します。デバッグセッションが開始されていない場合、このボタンによりデバッグセッションが開始され、最初に処理されるノードで停止します。セッションが動作中の場合、XSLT/XQuery ドキュメントはドキュメントの最後または次のブレークポイントまで処理されます。トレースポイントがセットされている場合、 処理命令がそこにたどり着いたときに 各トレースポイントにある命令の値がトレースウィンドウに表示されます。

アクティブなドキュメントのみを表示
デバッガーインターフェースにて現在アクティブなドキュメントのウィンドウを最大化します。

XSLT/XQuery と出力結果を表示
XSLT または XQuery ドキュメントウィンドウを表示し、XML ドキュメントを隠します。

XML、XSLT/XQuery、出力結果を表示
XML、XSLT/XQuery、そして出力ドキュメントを表示します。これは XML ドキュメントがデバッグセッションに割り当てられた時の標準のビューです。

デバッガーを停止
デバッガーを停止します。デバッグセッションの停止とは異なることに注意してください。デバッグを行っている最中にドキュメントの編集を行ったり、同じデバッグセッションにて別のファイルを使用したりするときに便利な機能になります。デバッガーを停止させた後は、XSLT/XQuery ドキュメントの最初からデバッガーを再度開始させることができます。

ステップイン (F11)
スタイルシートにある全てのノードや XPath 条件式の内部を処理します。このコマンドはデバッガーを再度開始するためにも使用することができます。

ステップオーバー (Ctrl+F11)
現在のノードから同じレベルの次のノードへ移るか、1つ上のレベルにある次のノードに移ります。このコマンドはデバッガーを再度開始するためにも使用することができます。

ステップアウト (Shift+F11)
現在のノードから、次の兄弟ノードに移動するか、1つ上のレベルにある次のノードに移ります。

現在の実行ノードを表示
XSLT/XQuery ドキュメントにある現在の実行ノードを表示、選択し、XML ドキュメント内の対応するノードをハイライトします。他のタブを選択し、XSLT スタイルシートまたは XML ファイルのコードを選択した後、以前作業していた場所を特定するのに便利な機能です。

デバッガーを再起動
出力ウィンドウに表示されている内容を破棄し、現在選択されているファイルでデバッグセッションを再度開始します。

ブレークポイントを挿入 削除 (F9)
現在のカーソル位置でブレークポイントを挿入、または削除します。インラインブレークポイントは XSLT/XQuery とXML の両方のノードに対して定義することができ、処理がどこで停止するかを決定することができます。赤色の点線がブレークポイントをセットしたノードの上の表示されます。終了 ノードにブレークポイントをセットすることはできず、XSLT ドキュメントの属性に対するブレークポイントは無視されます。このコマンドは右クリックして表示されるコンテキストメニューからも利用することができます。

トレースポイントを挿入 削除 (Shift+F9)
現在のカーソル位置でトレースポイントを挿入、または削除します。インライントレースポイントは XSLT ドキュメントのノードに対して定義することができます。デバッグ中にトレースポイントがセットされたステートメントにたどり着くと、そのステートメントの結果がトレースウィンドウに出力されます。青色の点線がトレースポイントをセットしたノードの上に表示されます。終了 ノードにトレースポイントをセットすることはできません。このコマンドは右クリックして表示されるコンテキストメニューからも利用することができます。

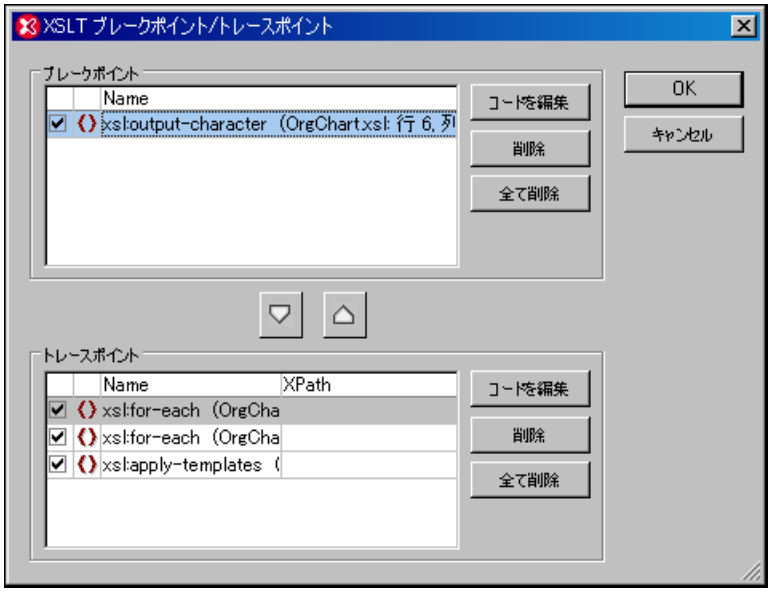
ブレークポイントの有効 無効化 (Ctrl+F9)
このコマンド (ツールバーアイコンにはありません) により、既に定義されたブレークポイントを有効 無効化することができます。ブレークポイントが無効化されると、赤色のブレークポイントが灰色で表示されます。デバッガーは無効化されたブレークポイントでは停止しません。ノード名までカーソルを移動させ、ブレークポイントの有効 無効化コマンドをクリックして、ブレークポイント

を有効 無効化することができます。このコマンドはブレークポイントのある位置で右クリックすることでも使用可能です。

トレースポイントの有効 無効化 (Shift+Ctrl+F9)
このコマンド (ツールバーアイコンにはありません) により、既に定義されたトレースポイントを有効 無効化することができます。トレースポイントが無効化されると、青色のトレースポイントが灰色で表示されます。無効化されたトレースポイントにおいて出力は行われません。ノード名までカーソルを移動させ、トレースポイントの有効 無効化コマンドをクリックして、トレースポイントを有効 無効化することができます。このコマンドはトレースポイントのある位置で右クリックすることでも使用可能です。

デバッグセッションの終了
デバッグセッションを終了し、デバッグセッションを開始する直前の、通常のXMLSpy ビューに戻ります。デバッグセッションで開かれた出力ドキュメントを開かれたままにするかは、XSLT/XQuery デバッガー設定ダイアログで設定することができます。

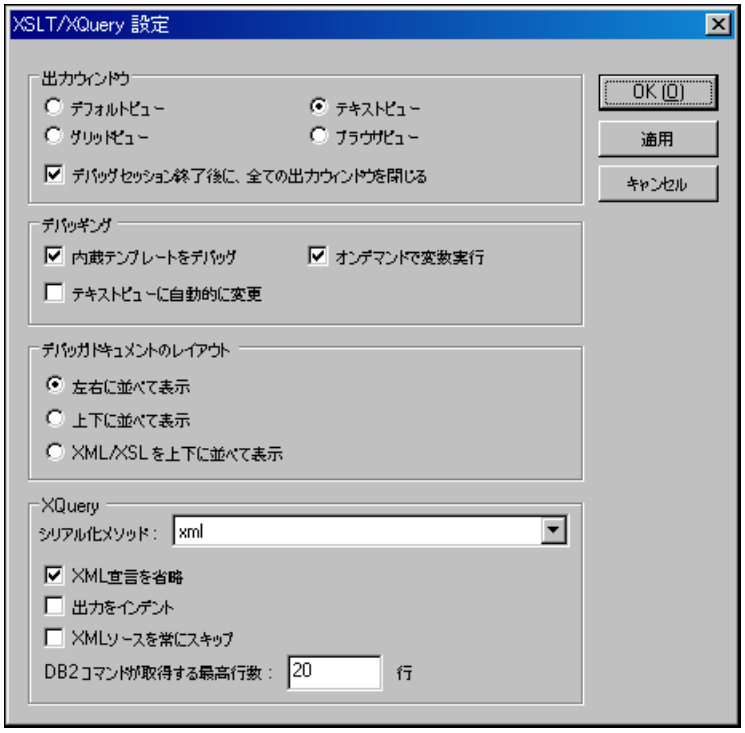
XSLT ブレークポイント/トレースポイントダイアログ
このコマンドによりXSLT/XQuery ブレークポイントならびにトレースポイントダイアログが開かれ、デバッグセッションにあるファイルで現在定義されている 無効化されているものも含めて) 全てのブレークポイント/トレースポイントが表示されます。



チェックボックスはブレークポイント/トレースポイントが有効になっているか (チェックされていれば有効) を表します。ブレークポイント/トレースポイントを選択し、**「削除」**ボタンを押すことで、ブレークポイント/トレースポイントの削除を行え、**「全て削除」**ボタンを押すことで、全てのブレークポイントまたはトレースポイントを削除することができます。**「コードを編集」**ボタンにより選択されたブレークポイント/トレースポイントへ移動します。

7.1.3 デバッガー設定

XSLT and XQuery Debugger 設定ダイアログでは、全てのデバッグセッションに適用されるデバッグならびに出力オプションを設定することができます。設定ダイアログを開くには、**「XSL/XQuery | XSLT/XQuery の設定」**を選択するか、ツールバーにある アイコンをクリックしてください。このセクションでは設定の詳細について記述します。



出力ウィンドウ

出力ドキュメントウィンドウを (デフォルト、テキスト、グリッド、またはブラウザービューから) 指定します。デフォルトビューとはオプションダイアログ (ツール| オプション) のファイルタイプタブにある (`xslt` や `.xq` といった) ファイルタイプにで選択されているビューのことです。XSLT 変換では、出力ファイルタイプはXSLT ファイルにで定義されます。XQuery の実行では、出力ファイルの種類は、このダイアログのXQuery 設定で指定するシリアル化メソッドにより決定されます (下記参照)。

全ての出力ウィンドウを閉じるオプションにより、デバッグセッション中に開かれていた出力ドキュメントを、デバッグセッションが終了した後にも開いたままにしておくか指定することができます。

デバッグ

内蔵テンプレートをデバッグ設定により、適切な箇所で内蔵テンプレート内のコードに、**「ステップイン」**することができるようになります。これはテンプレートタブの中にあるテンプレートエントリのクリックや、コールスタックにより**「表示」**される内蔵テンプレートファイルとは関係ありません。

XSLT はテキストビューまたはグリッドビューでのみ動作します。テキストビューに自動的に更新するオプションにチェックを入れておくと、デバッグを行うドキュメントがテキストビューやグリッドビューで表示されていない場合に、そのビューが自動的にテキストビューに切り替わります (XQuery デバッガーはテキストビューでのみ動作します)。オンデマンド変数実行チェックボックスがチェックされていれば、変数が呼び出されたときに変数の定義にステップインすることができるようになります。このオプションにチェックが入れられていない場合、変数の呼び出し場所に処理が到達しても、変数の定義へのステップインは行われず、次のステップへ移動します。

デバッガードキュメントのレイアウト

デバッガードキュメントとはデバッガーで開かれているドキュメントのことです。ここではこれらドキュメントを、左右、上下、または XML/XSLT ドキュメントを上下に並べて配置し、結果ドキュメントが左右になるような配置をすることができます。

7.1.4 デバッグセッションの開始

デバッグセッションを開始する最も単純な方法は、単一のXSLT、XQuery、またはXML ファイルから開始するというものです。必要な関連ファイル (関連するファイルのテーブルを参照) が既にアクティブなファイルに関連付けられている場合、デバッグセッションがすぐに開始されます。そうでない場合は必要な関連ファイルを選択するよう促されます。XQuery ファイルはXML ファイルの割り当てを含まず、必要ともしないので、XQuery ファイルからXQuery デバッグセッションを開始するたびにオプションのXML ファイル割り当てを促さないように設定することができます。

定義済みの割り当て

定義済みの割り当てはXSLT デバッグセッションでのみ有効になるもので、アクティブなファイルに対して既に割り当てファイルが存在する場合に参照されます。XML またはXSLT ファイルにて割り当てを行うには、以下の操作を行ってください：

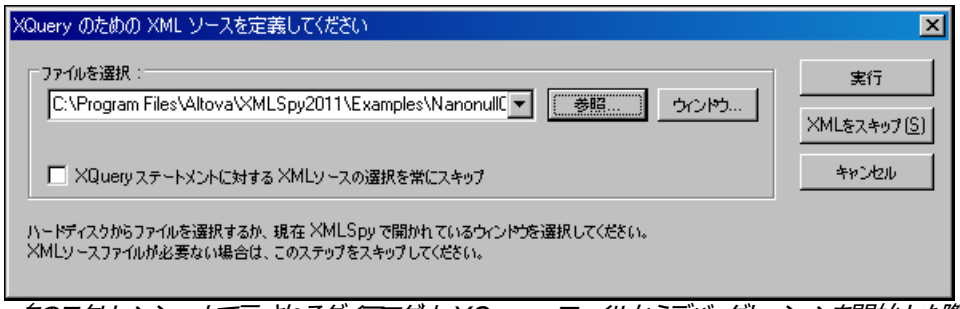
- XML ファイル：ファイルを開き、**「SL/XQuery | XSL の割り当て」**からXSLT ファイルを選択します。
- XSLT ファイル：ファイルを開き、**「SL/XQuery | サンプルXML ファイルの割り当て ..」**からXML ファイルを選択します。

「SL/XQuery | デバッガーの開始 実行」を選択すると、デバッグセッションが直接開始されます。アクティブなファイルに対してファイルの割り当てを行うよう促されることはありません。

直接の割り当て

アクティブなファイルに対して定義済みの割り当てが存在しない場合、割り当てを行うよう促されます。**「SL/XQuery | デバッガーの開始 実行」**を選択すると、以下のような動作が発生します：

- XML ファイルの場合：XSLT またはXQuery ファイルを選択するよう促されます。
- XSLT ファイルの場合：XML ファイルを選択するよう促されます。
- XQuery ファイルの場合：XML ファイルを選択するオプションが与えられます。スキップすることもできます。



(このスクリーンショットで示されるダイアログは、XQuery ファイルからデバッグセッションを開始した際に表示されるものです)

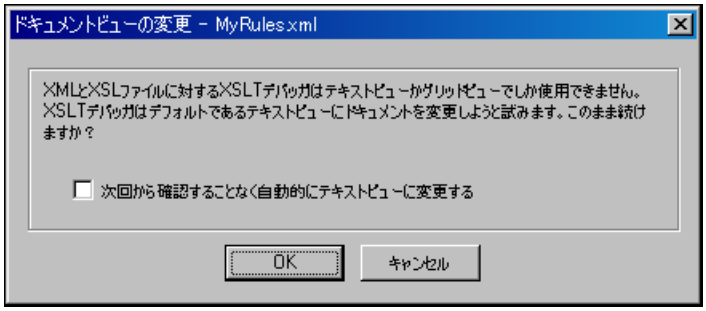
必要な割り当てファイルを選択するか、オプションの割り当てをスキップすると、デバッグセッションが開始されます。

その他のファイル割り当てを行う方法

プロジェクトプロパティダイアログでは、定義済みの割り当てを行うことができます。**「プロジェクト | プロパティ」**を選択し、**「使用するXSL / 使用するXML」**チェックボックスにて必要なファイルを割り当ててください。

デバッガービュー

XSLT and XQuery Debuggerはテキストビューおよび拡張グリッドビューでのみ動作します。お使いの XML またはXSLT ファイルが、テキストビューまたはグリッドビュー以外のビューで開かれている場合、またはSPS ファイルがXML ファイルに割り当てられている場合、デバッグセッションの起動時に以下のダイアログが表示されます。

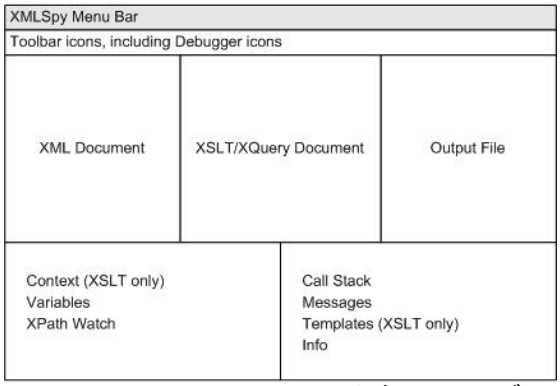


「OK」をクリックすることで、ドキュメントがテキストビューで開かれます。XQuery ファイルは常にテキストビューで表示されることに注意してください。

7.1.5 情報ウィンドウ

デバッグセッション中、XSLT and XQuery Debuggerのインターフェースに表示される情報ウィンドウにはXSLT 変換またはXQuery 実行に関する様々な情報が表示されます。この情報をXSLT やXQuery ファイルのデバッグを行う際に利用することができます。

XSLT デバッグセッションには8つの情報ウィンドウがあり、XQuery デバッグセッションには5つの情報ウィンドウが与えられます。これらウィンドウはデフォルトで2つのグループに分けられ、XSLT and XQuery Debuggerインターフェースの下部に配置されます（以下の図を参照）。このセクションでは、これらウィンドウと表示される情報に関する説明を行ないます。



XSLT and XQuery Debuggerインターフェースのデフォルトレイアウト。

情報ウィンドウにある最初のグループは、以下のタブとして単一のウィンドウに表示されます。

- コンテキスト（XSLT デバッグセッションのみ）
- 変数
- XPath-Watch

情報ウィンドウにある第2のグループは、以下のタブとして単一のウィンドウに表示されます。

- コールスタック
- メッセージ
- テンプレート（XSLT デバッグセッションのみ）
- Info
- トレース

デフォルトのレイアウトでは2つのウィンドウグループが存在し、各ウィンドウには様々なタブが含まれています。同時にアクティブにすることのできるタブは1つのみです。例えば、最初のウィンドウグループにて変数に関する情報を確認するには**[変数]** タ

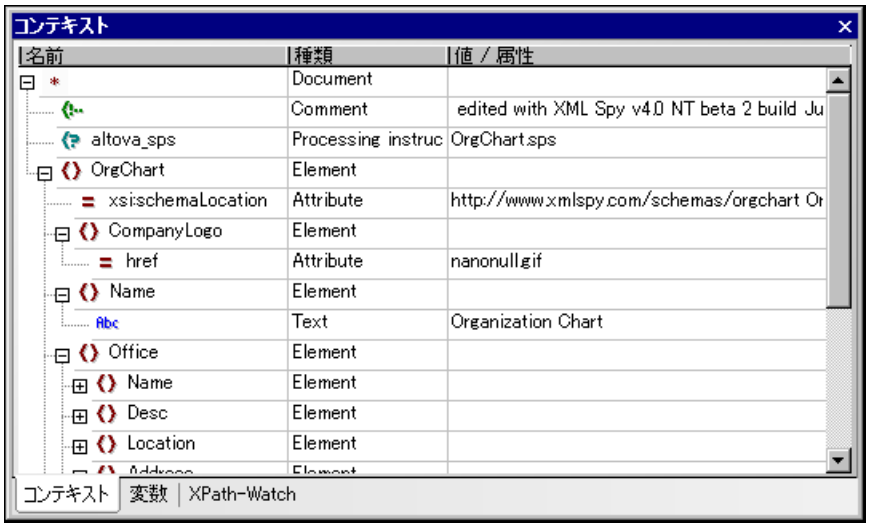
ブをクリックします。これにより変数情報ウィンドウが表示され、コンテキストならびにXPath-Watch 情報ウィンドウが隠されることになります。タブによってはナビゲーションを行うためのツールとしても使えることに注目してください。アイテムをクリックすることで、XML、XSLT、またはXQuery ファイル内の対応するアイテムに移動することができます。詳細は各情報ウィンドウのドキュメント (コンテキスト、コールスタック、テンプレート) を参照ください。

これら2つの情報ウィンドウグループは、ウィンドウの境界線をドラッグすることで大きさを変更することができます。タブ名をクリックして、ウィンドウの外にドラッグすることで、各情報ウィンドウを切り離すこともできます。ウィンドウのタイトルバーを、ウィンドウグループのタイトルバーにドラッグすることで、グループに加えることもできます。変更したレイアウトをデフォルトのレイアウトにリセットする方法は無いことに注意してください。

コンテキストウィンドウ

コンテキストウィンドウはXSLT デバッグセッションでしか利用できません。XQuery デバッグセッションでは表示されません。

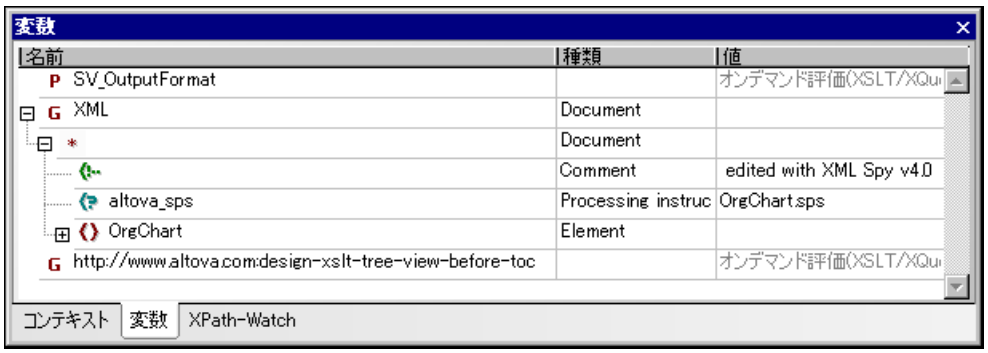
XSLT スタイルシートの処理中、プロセッサーのコンテキストは、常に (ノードやアトム値の) シーケンスにマッチするテンプレート内にあります。コンテキストウィンドウには、現在処理中のコンテキストである、ノード、シングルノード、または 文字列のような) アトム値のシーケンスが表示されます。コンテキストアイテムの種類に従い、値や属性が表示されます。例えばコンテキストアイテムが要素であれば、その要素の属性が表示され、属性またはテキストノードであれば、その値が表示されます。



コンテキストウィンドウのエントリをクリックすることで、XML ドキュメント内にあるそのアイテムが表示されます。XML ドキュメントがインターフェースに表示されていない場合、そのXML を表示するウィンドウが開かれます。

変数ウィンドウ

変数ウィンドウはXSLT ならびにXQuery デバッグセッションの両方で利用することができます。XSLT/XQuery ドキュメントのスコープ内で使用される変数やパラメーターが、値と共に表示されます。

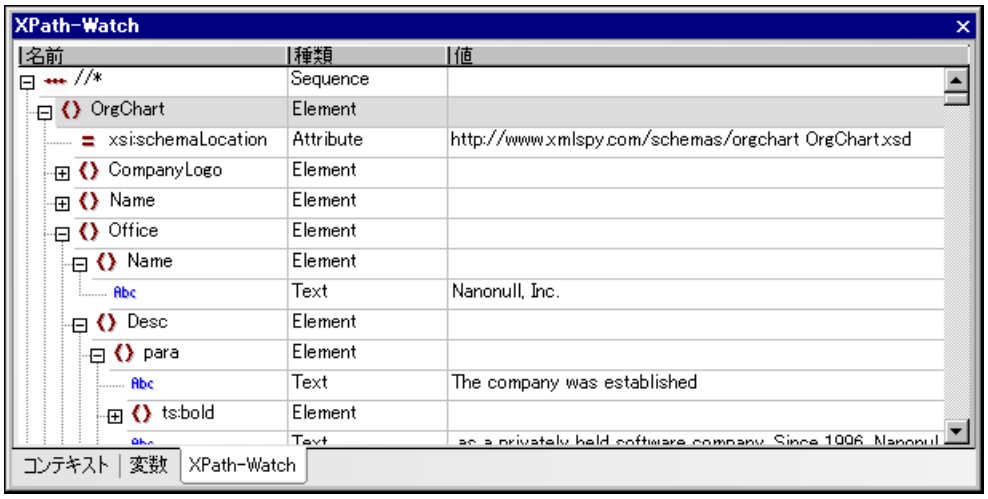


パラメーターは P と共に表示され、 (スタイルシートのトップレベルで宣言される) グローバル変数は G と共に表示されます。 (XSLT テンプレート内で宣言される) ローカル変数は L で表示されます。変数やパラメーターの値の種類は値フィールドのアイコンにより表示されます。以下の型が区別されます：ノードセット、ノードフラグメント、文字列、数値、Boolean。

XPath-Watch ウィンドウ

XPath-Watch ウィンドウはXSLT ならびにXQuery デバッグセッションにて利用可能です。

このウィンドウにより、コンテキスト上で評価を行ないたいXPath 条件式の入力を行うことができます。XSLT ドキュメントのステップを進めていくうちに、その時点のコンテキストでXPath 条件式が評価され、値列に結果が表示されます。



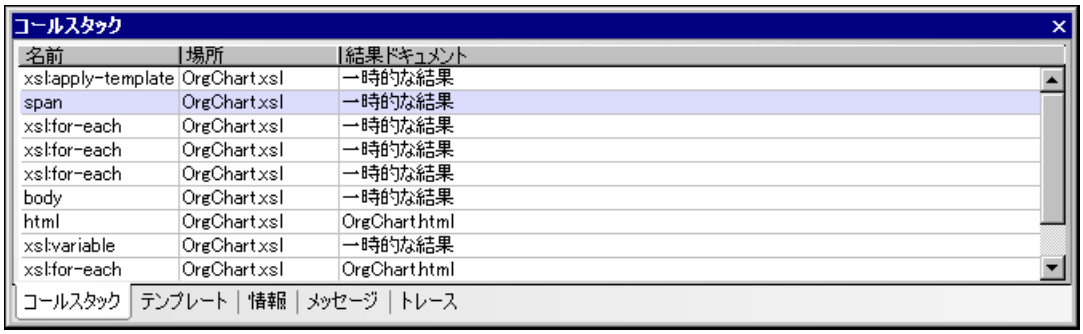
XPath 条件式を入力するには、名前列のテキストフィールドをダブルクリックして、XPath を入力します。他にも、ファイルからXPath 条件式をドラッグして、XPath-Watch ウィンドウにドロップすることもできます。XSLT スタイルシートのXSLT バージョンに合致するXPath のバージョンで使用される条件式を使用してください (XSLT 1.0 ではXPath 1.0 が使用され、XSLT 2.0 ではXPath 2.0 が使用され、XSLT 3.0 ではXPath 3.0 が使用されます)。

注意：XML ファイルまたはXSLT ファイルにて名前空間が使用されている場合、XPath 条件式でも、適切な名前空間プレフィックスを使用する必要があります。

コールスタックウィンドウ

コールスタックウィンドウはXSLT ならびにXQuery 両方のデバッグセッションにて利用可能です。

コールスタックウィンドウには、それまで処理したXSLT テンプレートと命令が、リストの最上部にある現在のテンプレートと共に表示されます。



ウィンドウのエントリをクリックすることで、選択されたXSLT テンプレートや命令がXSLT ドキュメントウィンドウ内に表示されます。内蔵のテンプレートを参照するテンプレートや命令をクリックすると、全ての内蔵関数を表示する新たなウィンドウが表示され、選択されたテンプレートがハイライトされます。

メッセージウィンドウ

メッセージウィンドウはXSLT とXQuery 両方のデバッグセッションで利用することができます。

XSLT 1.0、XSLT 2.0、およびXSLT 3.0

XSLT デバッグセッションのメッセージタブには、エラー、`xsl:message` 命令、またはデバッグ中に発生したエラーメッセージが表示されます。

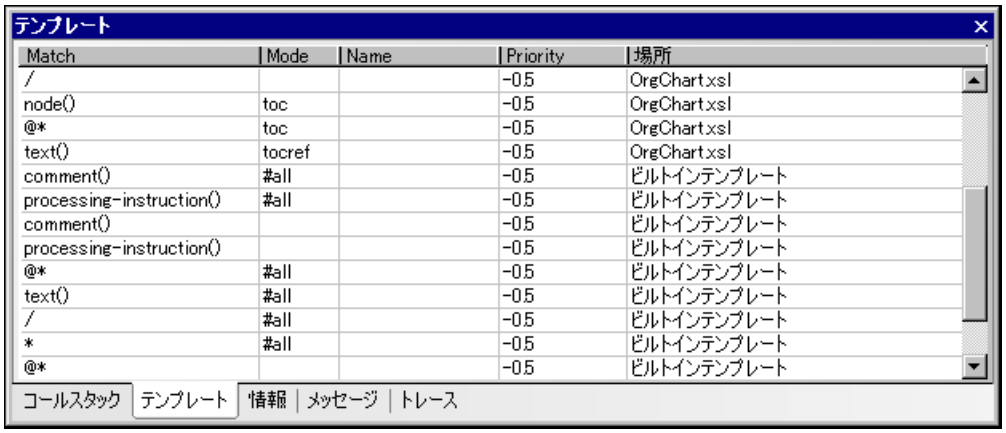
XQuery

XQuery デバッグセッションでは、メッセージウィンドウにエラーメッセージが表示されます。

テンプレートウィンドウ

(以下のスクリーンショットにある) テンプレートウィンドウはXSLT デバッグセッションでのみ表示され、XSLT デバッグセッションでは表示されません。

テンプレートウィンドウには、内蔵のテンプレートや名前付きテンプレートなどの、様々なXSLT スタイルシートが表示されます。マッチしたテンプレートは、ノードごとにまとめられて表示されます。名前付きテンプレートは名前順に表示されます。両方の種類のテンプレートにおいて、テンプレートの名前、モード、優先度、そして場所が表示されます。



上のスクリーンショットではXSLT スタイルシートにある3 つのテンプレートが表示されています：ドキュメントノード / にマッチするテンプレート、やテンプレート`n1:italic` と`n1:bold` ノードにマッチするテンプレートです。その他のテンプレートは

場所のフィールドにて示されている通り) 内蔵されている (ビルトインの) テンプレートです。

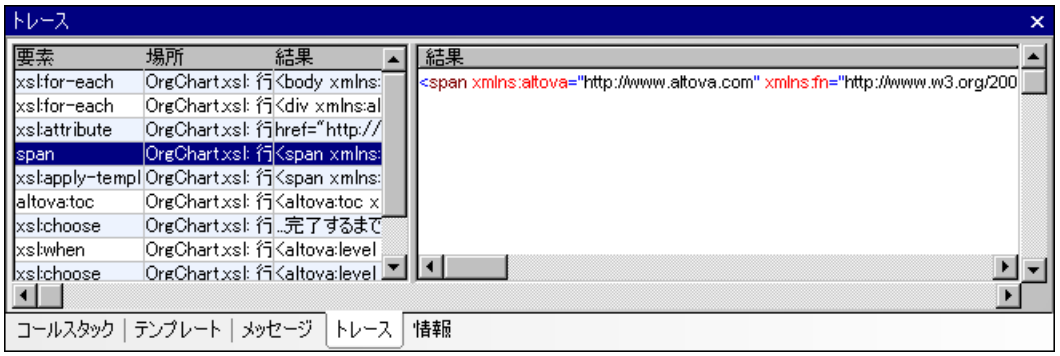
このウィンドウのエントリをクリックすることで、XSLT ドキュメントウィンドウにあるテンプレートがハイライトされます。内蔵のテンプレートをクリックすると、全ての内蔵テンプレートが表示されるウィンドウが新たに開かれ、選択したテンプレートがハイライトされます。

情報ウィンドウ

情報ウィンドウはXSLT とXQuery 両方のデバッグセッションで使用することができます。現在のデバッグセッションに関するメタ情報が表示されます。この情報の中には使用されているデバッガーや、ソースならびに出力ドキュメント、そしてデバッガーのステータスなどが表示されます。

トレースウィンドウ

トレースウィンドウはXSLT とXQuery 両方のデバッグセッションで利用することができます。



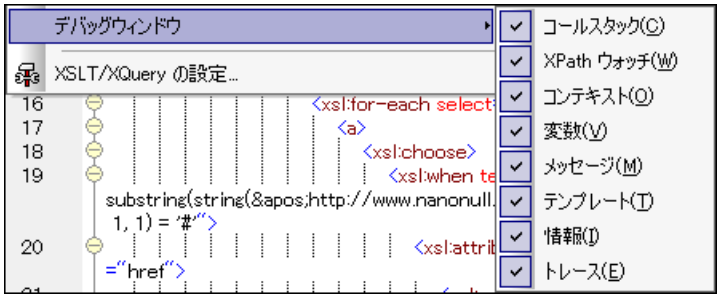
トレースウィンドウにはトレースポイントがセットされた要素、XSLT スタイルシート上の場所、そして要素が実行された際に生成される結果が表示されます。ウィンドウの左側にある行をクリックすることで、完全な結果が右側に表示されます。

情報ウィンドウの配置

情報ウィンドウをデバッガーインターフェースの内部に配置することもできます。ウィンドウはインターフェースにドッキングさせることも、固定させないでおくことも、更に別のウィンドウにて1つのペインとして配置することも可能です。以下のメカニズムを利用して、ウィンドウの配置を行うことができます。

メニュー

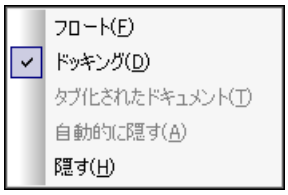
「**XSL/XQuery**」メニューにて、カーソルを「**デバッグウィンドウ**」に移動すると、情報ウィンドウのリストが表示されます。メニューアイテムをクリックすることでウィンドウの表示、非表示を切り替えることができます。



ウィンドウ表示のオン、オフを切り替えることができます。

コンテキストメニュー

ウィンドウのタブまたはタイトルバーを右クリックすることで、コンテキストメニューを表示することができます。



目的のオプションをクリックすることで、ウィンドウを切り離したり、ドッキングさせたりすることができます。

ドラッグアンドドロップ

タブまたはタイトルバーをドラッグアンドドロップすることで目的の場所に移動させることができます。

更に、ウィンドウを他のウィンドウにドッキングさせたり、ウィンドウをドラッグ中に表示される配置コントロールを使用することができます。

- ウィンドウを別のウィンドウにドラッグすると、(以下のスクリーンショットにある) 丸型の配置コントロールが表示されます。このコントロールは 5 つの部分に分かれており、これらのうちのどれかでマウスボタンを放すと、ドラッグしていたウィンドウが **「ターゲットとなるウィンドウ」** の対応する場所に配置されます。端にある 4 つのエリアにより、ターゲットとなるウィンドウの、対応する側への配置を行うことができます。中央にあるボタンにより、ドラッグしたウィンドウをターゲットとなるウィンドウのドッグとしてドッキングすることができます。ウィンドウを、別のウィンドウのタブバーまでドラッグアンドドロップすることで、タブとしてそのウィンドウを配置することもできます。

- ウィンドウをドラッグすると、4 つの矢印からなる配置コントロールが表れます。各矢印は **「デバッガーインターフェース」** の各側面に対応しており、これら矢印の上でウィンドウをドロップすると、ドラッグしたウィンドウをデバッガーの対応する位置にドッキングさせることができます。

ウィンドウのタイトルバーをダブルクリックすることでも、ウィンドウのドッキング、非ドッキング状態を切り替えることができます。

7.1.6 ブレークポイント

XSLT and XQuery Debugger では XSLT、XQuery、そして XML ドキュメントにてブレークポイントを定義することができます。ブレークポイントは (以下のスクリーンショットにあるように) 赤色の点線で表示されます。

メモ: 同一の場所にブレークポイントとトレースポイントの両方をセットすることも可能です。 (スクリーンショットにあるように) 青と赤による線として表示されます。

```
<!-- Local Variable reused -->
<xsl:variable name="OfficeName" select="n1:Name"/>
<!-- Display the company name if the varaible is true ********* -->
<xsl:if test="$Show_Company_Name">
    <h3>
        <xsl:value-of select="$OfficeName"/>
    </h3>
    <xsl:message>
        <xsl:text>Company Named Displayed</xsl:text>
    </xsl:message>
</xsl:if>
```

デバッグセッションでデバッガーを開始すると、各ブレークポイントにてデバッギングが停止します。この方法を使うと、デバッグを行う場所を限定し、XSLT、XQuery、そして XML ドキュメント内の特定の場所に注意を集中させることができます。セットできるブレークポイントの数に制限はありません。

注意: ドキュメント内にセットされたブレークポイントの情報は、そのドキュメントが閉じられるまで保持されたままになります。しかし (例えば XSD ドキュメントにて) スキーマビューへの切り替えが行われると、ブレークポイントは削除されます。その後テキストビューやグリッドビューに切り替え直しても、ブレークポイントは表示されません。

XML ドキュメント内のブレークポイント

XML ドキュメントでは任意のノードにブレークポイントをセットすることができます。ブレークポイントによる処理の停止は、そのノードの開始点で行われます。

XSLT ドキュメント内のブレークポイント

XSLT ドキュメントでは以下の箇所でブレークポイントをセットすることができます:

- テンプレートや (例えば `xsl:for-each` といった) テンプレート命令の開始箇所
- XPath 条件式 (XPath 1.0 または XPath 2.0)
- 文字列により構成される XML フラグメント。ブレークポイントによる処理の停止は、そのノードの開始点で行われます。

XQuery ドキュメント内のブレークポイント

XQuery ドキュメントでは、以下の箇所でブレークポイントをセットすることができます:

- XQuery ステートメントの開始箇所
- XQuery 表現内部
- 文字列により構成される XML フラグメント。ブレークポイントによる処理の停止は、そのノードの開始点で行われます。

ブレークポイントの挿入と削除

ブレークポイントを挿入するには:

1. ドキュメント内でブレークポイントを挿入したい箇所にカーソルを移動させます (上の記述を参照ください)。XSLT デバッグセッションでは、テキストビューならびにグリッドビューにてブレークポイントをセットすることができ、XQuery デバッグセッションでは、テキストビューでのみブレークポイントを使用することができます。
2. 以下のどれかの操作を行います:

- 「**XSL/XQuery | ブレークポイントの挿入 削除**」を選択
- **F9** を押下
- 右クリックして、「**ブレークポイントの挿入 削除**」を選択

ブレークポイントを削除するには：

1. ドキュメント内のブレークポイントが存在する箇所にカーソルを移動させます。
2. 以下のどれかの操作を行ないます：
   - 「**XSL/XQuery | ブレークポイントの挿入 削除**」を選択
   - 「**F9**」を押下
   - 右クリックして「**ブレークポイントの挿入 削除**」を選択

その他にも、ブレークポイント/トレースポイントダイアログを使用することでブレークポイントを削除することができます：

1. メニューオプションから「**XSL/XQuery | ブレークポイント/トレースポイント ..**」を選択
2. ダイアログボックスからブレークポイントを選択して、「**削除**」ボタンをクリック

「**全て削除**」ボタンにより、ダイアログボックス そしてXSLT スタイルシートにある) 全てのブレークポイントを削除することができます。

ブレークポイントの有効化 無効化

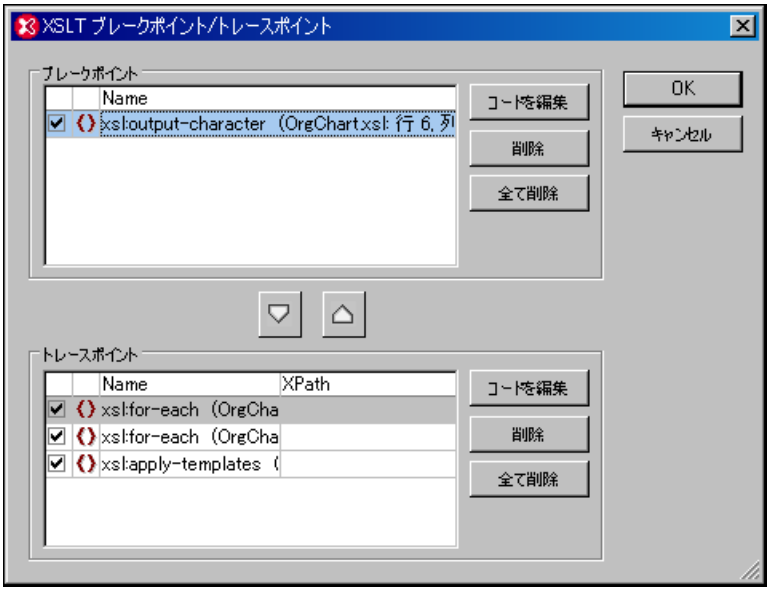
ブレークポイントを挿入した後に、そのブレークポイントを 削除すること無く スキップしたい場合は、ブレークポイントの無効化を行うことができます。必要に応じて、ブレークポイントは再度有効化することができます。

ブレークポイントを無効化するには：

1. ブレークポイントが含まれているノードまたは表現にカーソルを移動します。
2. 「**XSL/XQuery | ブレークポイントの有効化 無効化**」を選択します (または 「**Ctrl**」 + 「**F9**」を押下します)。ブレークポイントが赤色からグレーになり、無効化されます。

その他にも、ブレークポイント/トレースポイントダイアログを使用することでブレークポイントの無効化を行うことができます：

1. メニューオプションから 「**XSL/XQuery | ブレークポイント/トレースポイント ..**」を選択して、XSLT ブレークポイント/トレースポイントダイアログを表示します。ダイアログには開かれているXML ソースならびにXSLT スタイルシートにて現在定義されている全てのブレークポイントが表示されます。



2. 無効化したいブレークポイントのチェックマークを外し、「**OK**」ボタンにて確定します。ブレークポイントが赤色からグレ

ーになり無効化されます。

ブレークポイントを有効化するには：

1. ブレークポイントを含んだコードや表現にカーソルを移動します。
2. 「**SL/XQuery | ブレークポイントの有効化 無効化**」を選択します（または「**Ctrl**」+「**F9**」を押下します）。ブレークポイントがグレーから赤色になり、有効化されます。

特定のブレークポイントを見つける
特定のブレークポイントを見つけるには：

1. メニューオプションから「**SL/XQuery | ブレークポイント/トレースポイント ...**」を選択します。XSLT ブレークポイント/トレースポイントダイアログが表示されます。
2. ブレークポイントのリストから目的のブレークポイントをクリックします。
3. 「**コードを編集**」ボタンをクリックします。ブレークポイントダイアログが閉じられ、テキストビューにて選択したブレークポイントの直前にカーソルが配置されます。拡張グリッドビューの場合、ブレークポイントを含んだテーブルセルが赤色でハイライトされます。

ブレークポイント後の処理を継続
ブレークポイントの後にある処理を継続するには：

- 「**SL/XQuery | ステップイン**」または「**SL/XQuery | デバッガーの開始 実行**」コマンドを選択してください。

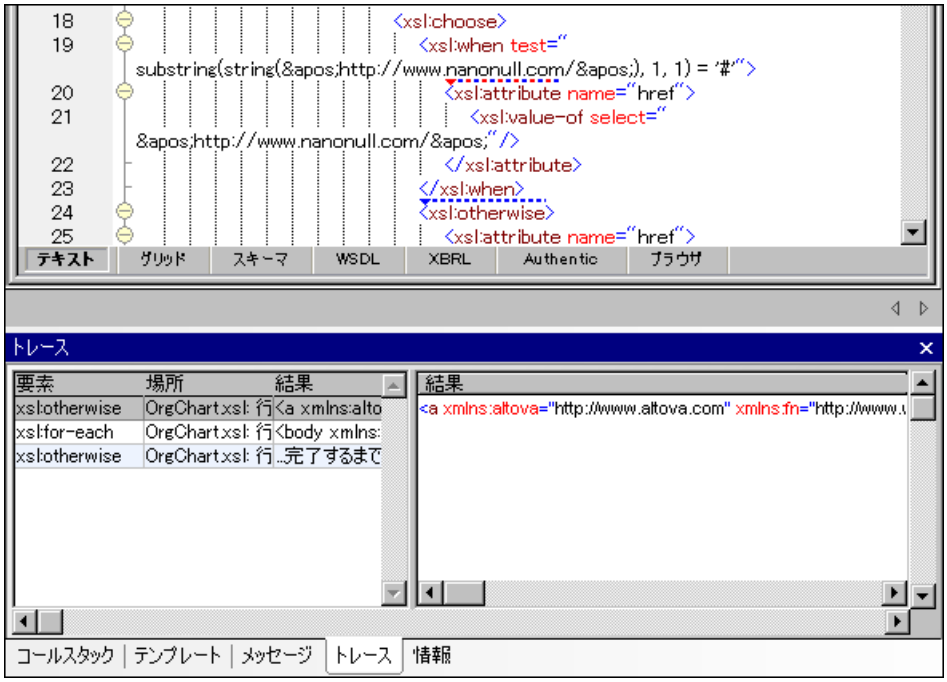
7.1.7 トレースポイント

XSLT and XQuery Debuggerでは XSLT ドキュメントのトレースポイントを定義することができます。

トレースポイントを使うことで、命令により生成されたコンテンツをトレースしたり、（xsl:message 要素を使ってメッセージを表示させるなどの）XSLT スタイルシートの編集を行うこと無く、トレースポイントがセットされた箇所における XPath 条件式の結果を確認することができます。

トレースポイントは（下のスクリーンショットにあるように）XSLT スタイルシートにて青色の点線で表示されます。

メモ：同一の場所にブレークポイントとトレースポイントの両方をセットすることも可能です。（スクリーンショットにあるように）青と赤による線として表示されます。



トレースポイントがセットされた箇所では、各命令により生成された結果が出力されます。この出力はトレースウィンドウで確認することができます。セットすることのできるトレースポイントの数に制限はありません。

注意：セットされたトレースポイントはドキュメントが閉じられるまで保持されます。

XSLT ドキュメントのトレースポイント

トレースポイントは XSL 命令や XSLT スタイルシートのリテラル結果にセットすることができます。

XML と XQuery ドキュメントのトレースポイント

トレースポイントを XML や XQuery ドキュメントにセットすることはできますが、これらのトレースポイントは効果を持ちません。

トレースポイントの挿入 削除

トレースポイントを挿入するには：

1. XSLT ドキュメント内でトレースポイントをセットしたい箇所にカーソルを移動させます。デバッグセッションでは、テキストビューならびにグリッドビューでトレースポイントをセットすることができます。
2. 以下のどれかの操作を行ないます：
   - 「**XSL/XQuery | トレースポイントの挿入 削除**」を選択
   - 「**Shift**」 + 「**F9**」を押下
   - 右クリックして「**トレースポイントの挿入 削除**」を選択

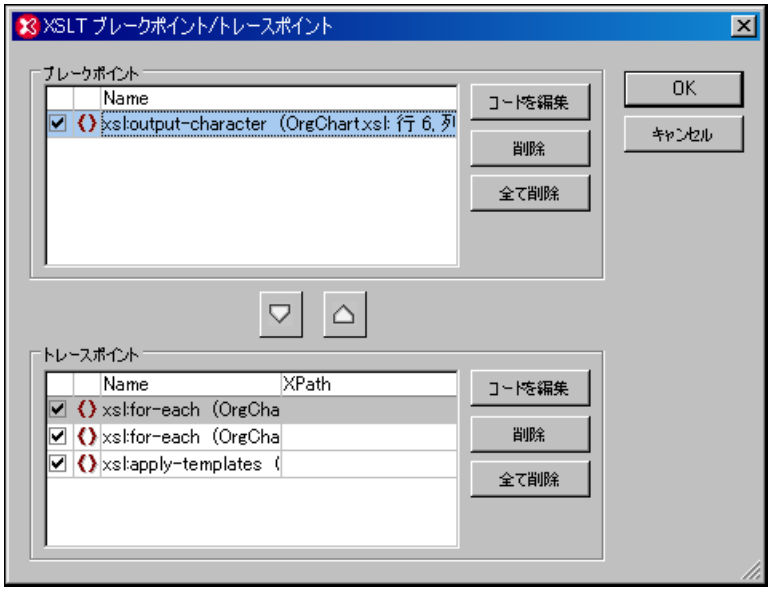
トレースポイントを削除するには：

1. XSLT ドキュメント内のトレースポイントがある場所にカーソルを移動させます。
2. 以下のどれかの操作を行ないます：
   - 「**XSL/XQuery | トレースポイントの挿入 削除**」を選択
   - 「**Shift**」 + 「**F9**」を押下
   - 右クリックして「**トレースポイントの挿入 削除**」を選択

他にも、XSLT ブレークポイント/トレースポイントダイアログからトレースポイントの削除を行うことができます：

1. メニューオプションから「**XSL/XQuery | ブレークポイント/トレースポイント ..**」を選択します。

2. 次のスクリーンショットのように、ダイアログボックスのトレースポイントをクリックし、**「削除」**をクリックします。



トレースポイントペインにある **「全て削除」**ボタンにより、ダイアログボックスに表示されている そしてXSLT スタイルシートにある) 全てのトレースポイントを削除します。

トレースポイントにXPath をセット

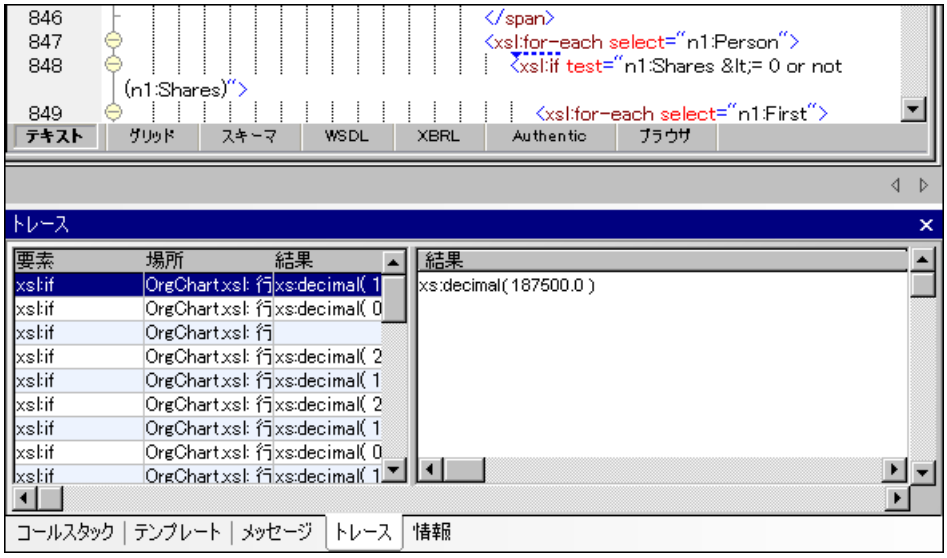
XPath に対してもトレースポイントをセットすることができます。XPath に対してトレースポイントをセットすると、ステートメントにより生成されるコンテンツの代わりに、XPath の評価結果がトレースウィンドウに表示されます。XPath の評価は、トレースポイントがセットされた箇所のコンテキストとは相対的に評価されます。

XPath にトレースポイントをセットするには：

1. メニューオプションから **「XSL/XQuery | ブレークポイント/トレースポイント ...」**を選びます。XSLT ブレークポイント/トレースポイントダイアログには、現在開かれている全てのXSLT スタイルシートドキュメント内で定義されているトレースポイントが表示されます。
2. 目的のトレースポイントの行の **「XPath」**フィールドにてXPath を入力します。

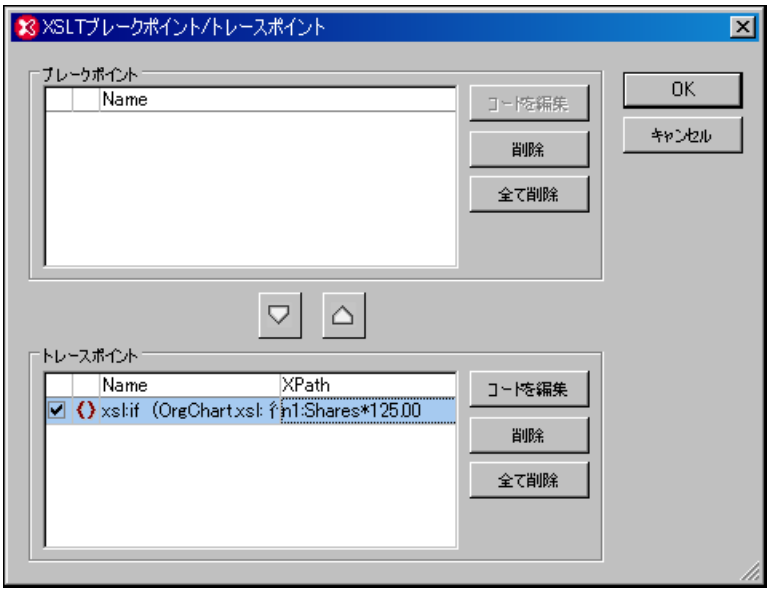
例:

以下のサンプルに示される通り、コンテキストノードが `Person` となるようにトレースポイントがセットされています。`Person` 要素には `Shares` 要素が含まれています。ここでは各 `Person` が持つ `Shares` の数を 125 でかけた値を表示します。



以下の操作を行ってください：

1. 555 行にてトレースポイントをセットします。
2. XSLT ブレークポイント/トレースポイントダイアログを開き、先程セットしたトレースポイントの XPath にて `n1:Shares*125.00` と入力します。

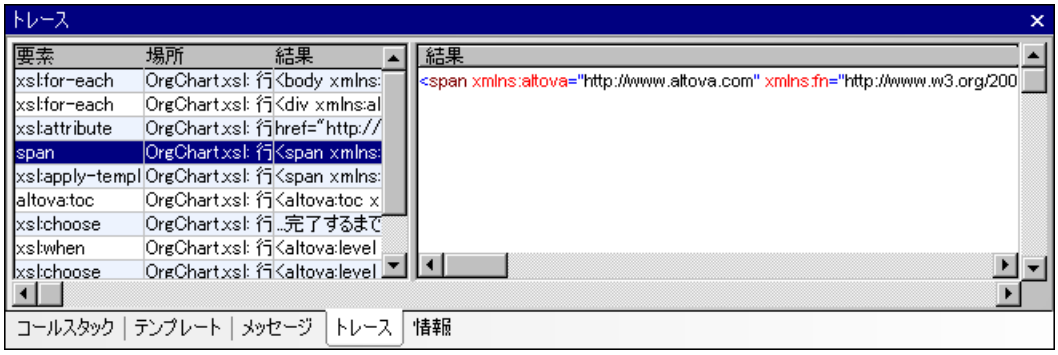


3. デバッガーを開始します。入力した XPath の出力がトレースポイントウィンドウに表示されます。

トレースウィンドウ

「XSL/XQuery | デバッガーの開始 実行」を選択して、デバッグをスタートさせます。トレースポイントがセットされた命令の出力がトレースウィンドウに表示されます（次のスクリーンショットを参照）。トレースウィンドウのアイテムをクリックすることで、その結果の詳細がウィンドウの右側に表示されます。

メモ：トレースウィンドウに表示される結果は、トレースされている命令が終了した後に表示されます。



トレースポイントの有効化 無効化

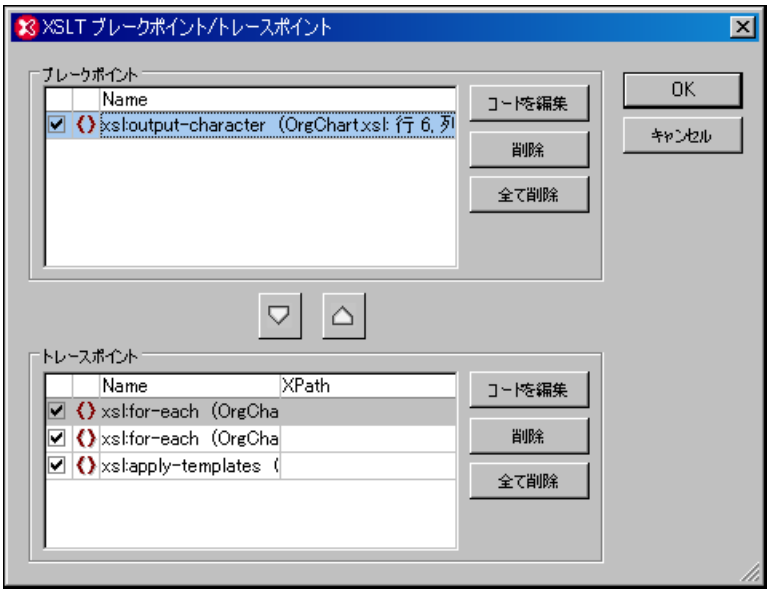
トレースポイントを挿入した後、それらを無効化することで、削除すること無くトレースポイントをスキップすることができます。必要になれば再度有効化することもできます。

トレースポイントを無効化するには：

1. XSLT スタイルシートにある無効化したいトレースポイントにカーソルを移動します。
2. 「**XSL/XQuery | トレースポイントの有効化 無効化**」を選択するか、「**Ctrl**」 + 「**Shift**」 + 「**F9**」を押下します。トレースポイントが青色からグレーに変わり、無効化されます。

その他にも、XSLT ブレークポイント/トレースポイントダイアログからトレースポイントの無効化を行うことができます：

1. メニューオプション「**XSL/XQuery | ブレークポイント/トレースポイント ..**」を選択します。XSLT ブレークポイント/トレースポイントダイアログボックスが開かれ、開かれているXSLT スタイルシートドキュメント内で現在定義されている全てのトレースポイントが表示されます。



2. 各トレースポイントにあるチェックを外すことで、トレースポイントを無効化して、「**OK**」で確定します。トレースポイントが青色からグレーに変わり、無効化されます。

トレースポイントを有効化するには：

1. XSLT スタイルシートにある有効化したいトレースポイントにカーソルを移動します。
2. 「**XSL/XQuery | トレースポイントの有効化 無効化**」を選択するか、「**Ctrl**」 + 「**Shift**」 + 「**F9**」を押下します。トレースポイントが青色からグレーに変わり、無効化されます。

特定のトレースポイントを見つける

特定のトレースポイントを見つけるには：

1. メニューオプションから『**XSL/XQuery | ブレークポイント/トレースポイント ..**』を選択します。XSLT ブレークポイント/トレースポイントダイアログが表示されます。
2. リストにある目的のトレースポイントをクリックします。
3. 『**コードを編集**』ボタンをクリックします。XSLT ブレークポイント/トレースポイントダイアログが閉じられ、テキストビューにて選択したトレースポイントの直前にカーソルが配置されます。拡張グリッドビューの場合、トレースポイントを含んだテーブルセルが青色でハイライトされます。

7.1.8 デバッガーショートカット

デフォルトのデバッガーショートカットは以下にリストされているとおりです

- *デバッガーコマンド*

F9	ブレークポイントの挿入/削除
F9 + Shift	トレースポイントの挿入/削除
F9 + CTRL	ブレークポイントのの有効化/無効化
F9 + Shift + CTRL	トレースポイントの有効化/無効化
F11	ステップイン
F11 + Shift	ステップアウト
F11 + CTRL	ステップオーバー
F11 + Alt	デバッガーを開始する

7.2 XSLT と XQuery プロファイラー

Altova ウェブサイト: XSLT プロファイラー、XQuery プロファイラー

XSLT/XQuery プロファイラーを利用することで、XMLSpy 内部からXSLT (1.0 および 2.0) スタイルシートや XQuery ドキュメントの実行時間を分析することができます。XSLT スタイルシートや XQuery ドキュメントにある各命令の実行に必要な時間が表示され、これらファイルの実行時間を最適化するための材料として使うことができます。

プロファイラーを使用することで、実行時間が最も長い命令を発見し、全体の最適化を行うことができます。
命令の合計実行時間が長くなる原因として、以下の2つが考えられます。

- 長い実行時間を必要とする命令である
- 命令の呼び出し回数が高い (高いヒットカウント)

ヒットカウントとコールグラフプロファイリング

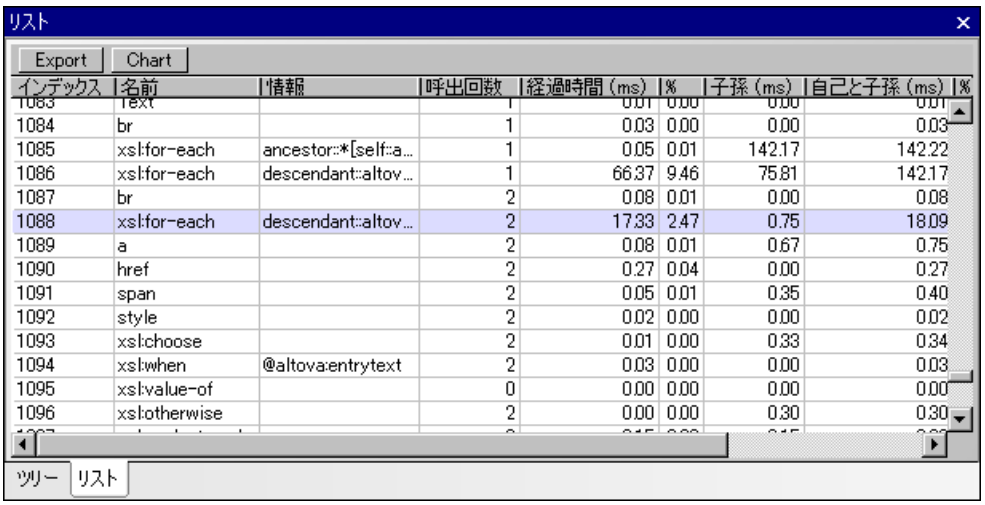
プロファイラーでは*ヒットカウント*と*コールグラフプロファイリング*からの選択を行うことができます。各命令の実行結果の表は両方のプロファイリングで表示されます。
最適化の為には、各命令を1行で表示するヒットカウントプロファイリングが通常使用されます。
コールグラフプロファイリングではXSLT 変換またはXQuery 実行全体の履歴が表示され、例えば変換において呼び出されたテンプレートや関数が表示されます。コールグラフプロファイリングでは、命令毎に1行使用するのではなく、命令の呼び出し毎の表示が行われます。

XSLT/XQuery プロファイラーを使用するには、XSLT プロファイリングとXQuery プロファイリングを参照ください。

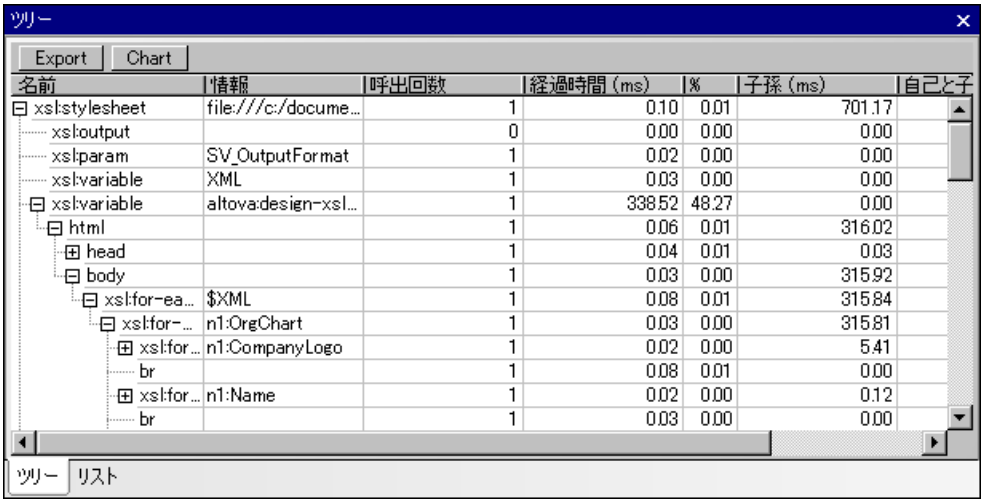
プロファイラービュー

分析の結果は、タブにより切り替えることのできる以下のビューのどれかで確認することができます：

- **リスト表示**：プロファイリングの結果がリスト形式で表示され、例えば命令の実行に要した経過時間により並べ替えを行うことができます。



- **ツリー表示**：結果がツリー構造に似た形式で表示されます。例えば関数の実行に要した時間を確認したり、その関数のツリーを展開して、関数内にある各命令が実行に要した時間と実行回数を参照することができます。



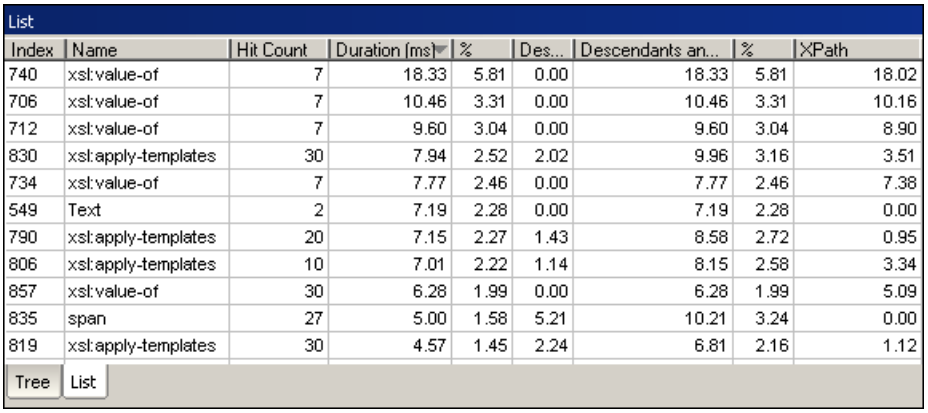
結果の並び替え

プロファイリングを行った後、各命令が実行に要した時間や、各命令が呼び出された回数により並べ替えを行うことができます。

プロファイラー内の情報を並べ替えるには：

1. **「リスト」**タブをクリックします。
2. 並べ替えを行いたい列のヘッダーをクリックします。例えば**呼び出し回数**では、命令が呼び出された回数で、**経過時間**では実行に要した時間により並べ替えが行われます。

このスクリーンショットでは命令の経過時間を降順で並べ替えた結果を示します。



XSLT スタイルシートとXQuery ドキュメントの最適化

XSLT スタイルシートならびにXQuery ドキュメントにある命令の最適化を行う際には、以下の点に注意してください：

- 変数が1度だけしか使用されないのであれば、命令内での変数の使用を避ける（変数の宣言には時間がかかるため）。

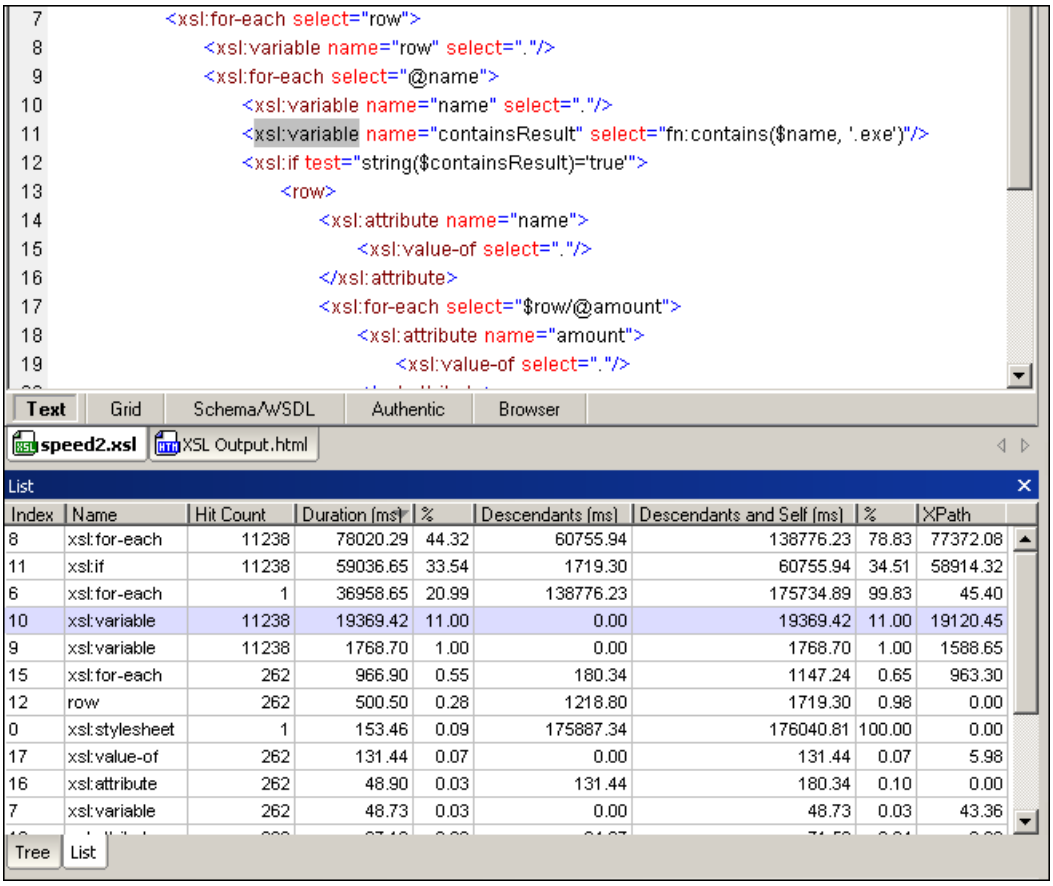
  以下の2つのXSLT コードフラグメントにて、必要ない変数を削除することでコードの最適化を図る例を示します。両者は同じことを行っていますが、2つめの例では変数 `name` と`containsResult` を使用すること無く同じ処理を行っています：

  コードフラグメント1：

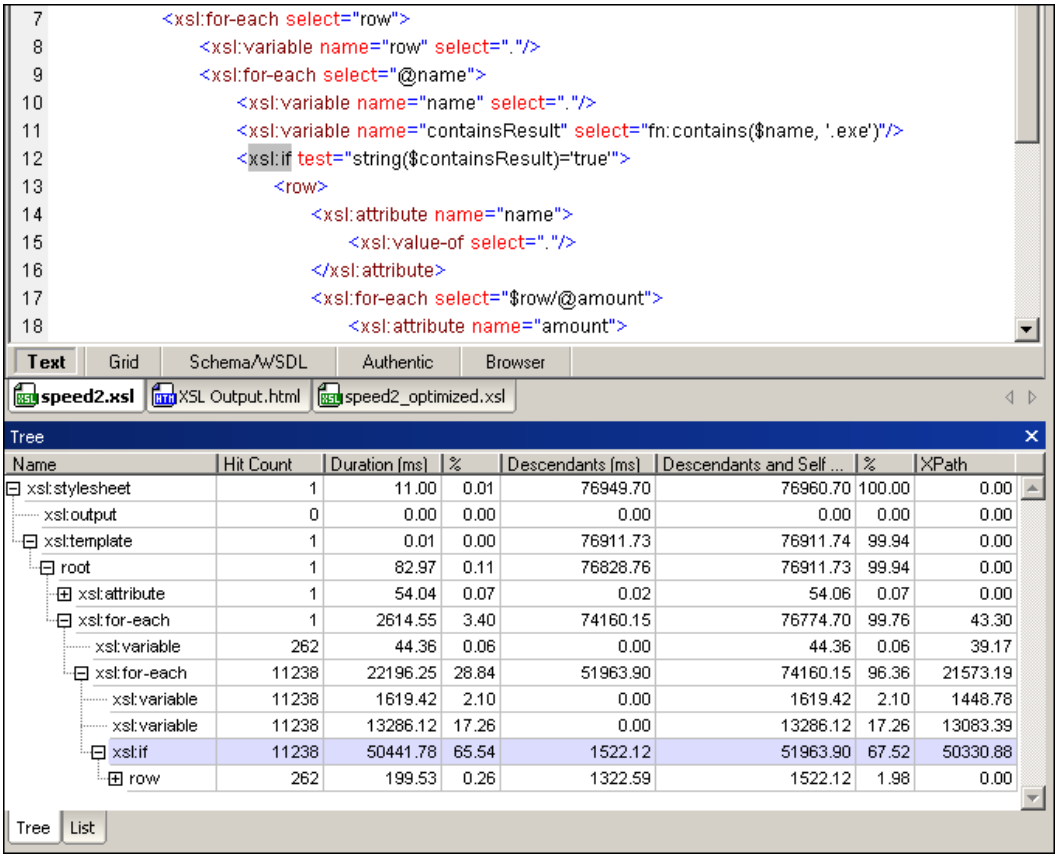
```
<xsl:for-each select="row">
       <xsl:variable name="row" select="."/>
       <xsl:for-each select="@name">
```

```
                <xsl:variable name="name" select="."/>
                <xsl:variable name="containsResult"
select="fn:contains($name, '.exe')"/>
                <xsl:if test="string($containsResult)='true'">

                ...

                </xsl:if>
        </xsl:for-each>
 </xsl:for-each>
```

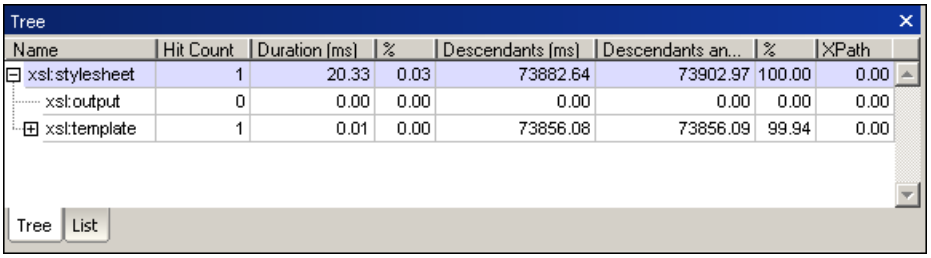
以下のスクリーンショットでは、このコードフラグメントを含むファイルの分析結果を命令の長さにより並べ替えた結果を示しています。変数 `containsResult` が初期化するための命令に、合計で 19秒の実行時間が必要になっています。



以下のスクリーンショットでは分析結果がツリー表示で示されています。ここでは、変数 containsResult を使用するIF ステートメントの実行に合計で約 50秒かかることが確認できます：



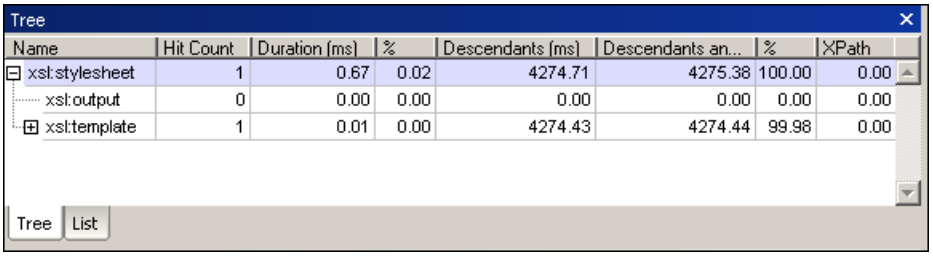
XSLT 変換には合計で約 74秒必要なことが確認できます：



コードフラグメント2：

```
<xsl:for-each select="row">
 <xsl:variable name="row" select="."/>
 <xsl:for-each select="@name">
       <xsl:if test="fn:contains(., '.exe')">

        ...

        </xsl:if>
 </xsl:for-each>
</xsl:for-each>
```

これらの変数を使用すること無くスタイルシートを書き直すと、合計実行時間は約 4.3秒まで短縮できることが確認できます。



- 値や条件式が繰り返し使用される場合には変数を使用する。
- 関数内ではローカル定数の作成を避ける。代わりにグローバル変数を使用する。
- 関数内では静的なツリーフラグメントの作成を避ける。代わりにグローバルで作成する。
- かっこを使ったフィルタリングの評価は各ノードに対して行われるため、かっこの使用を制限する。例えば、名前を使ったフィルタリングを行うことで、かっこの使用数を減らすことができます。以下の例では、*は2つのかっこと共に使用されています：

```
//*[node-name()=Book][author="Steve"]
```

等価の処理を行う以下のステートメントでは、Book という名前と1つのかっこが使用されています：

```
//Book[@Author="Steve"]
```

- 1度しか実行されなくても良い命令は1度しか呼び出されないように命令を分割する。グローバルな文脈でしか使用されない部分からは、グローバル変数を作成します。

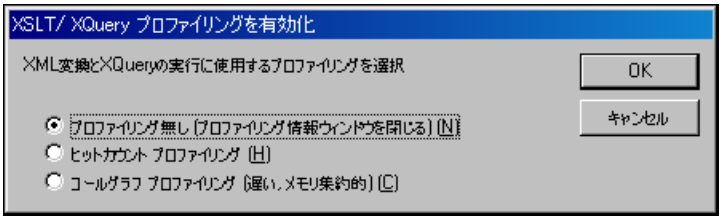
7.2.1 XSLT プロファイリング

プロファイラーの開始

注意：プロファイラーに表示される実行時間は、お使いのコンピューターで同時に動作している他のアプリケーションにより影響を受ける可能性もあります。プロファイラーを使ってファイルの分析を行うには、XMLSpy アプリケーションのみを動作させることが理想的です。

XSLT スタイルシートの分析を行うには：

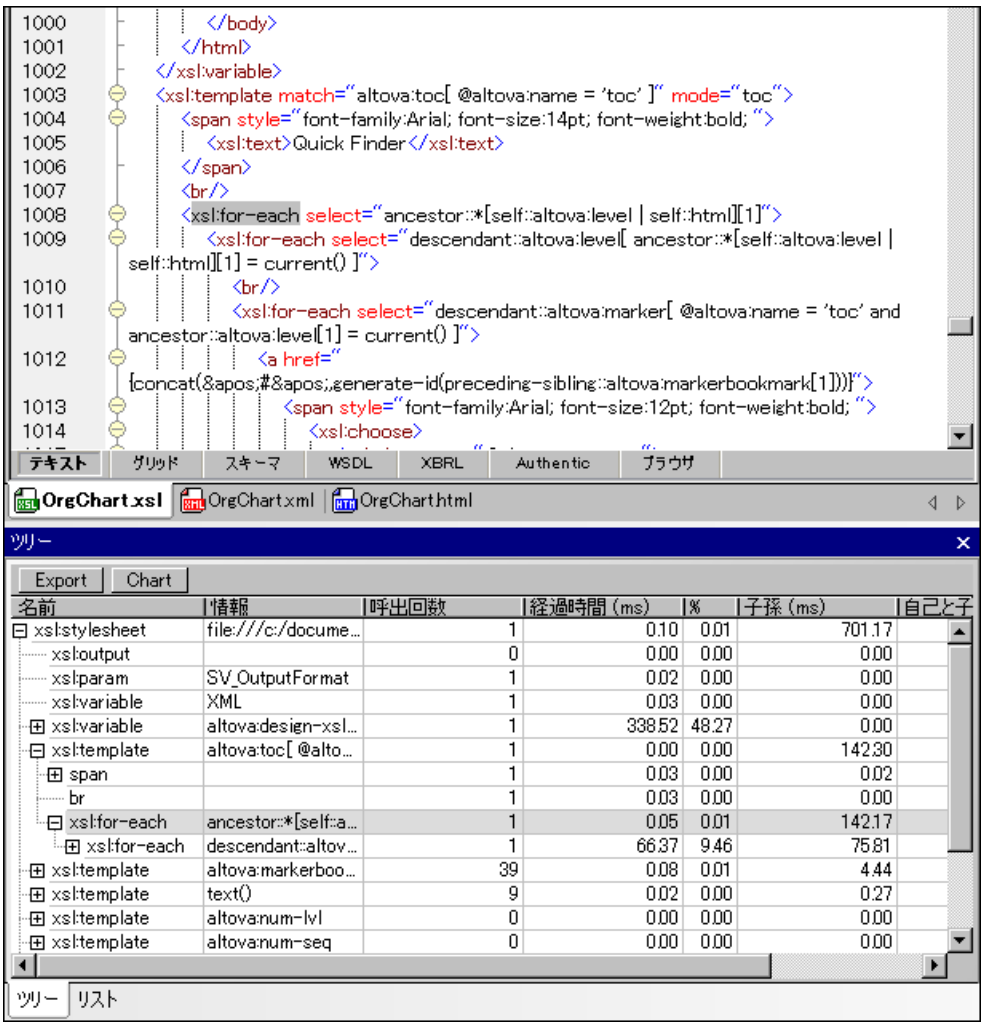
1. XMLSpy にて XSLT 変換の入力データとして使用される XML ファイルを開きます。
2. **「XSL/XQuery | XSLT/XQuery プロファイリングの有効化」**を選択します。以下のダイアログが開きます。



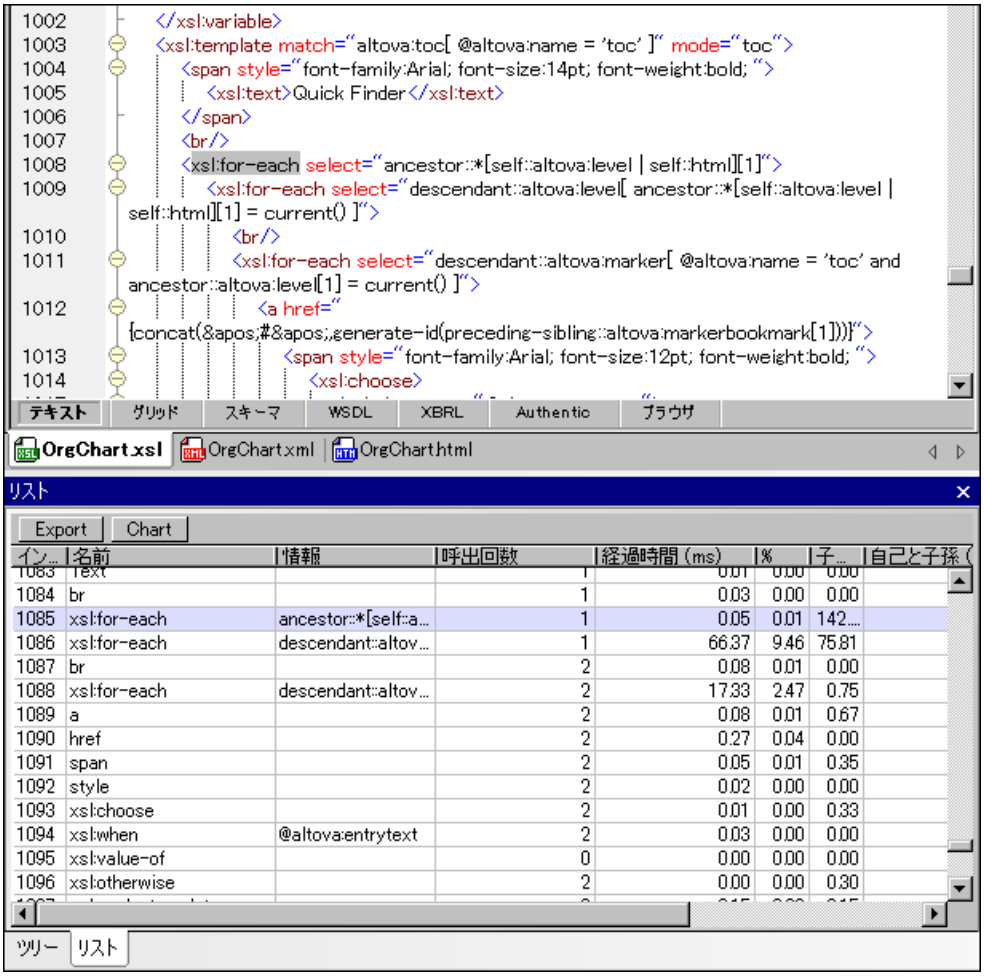
3. **「ヒットカウントプロファイリング」**または **「コールグラフプロファイリング」**を選択します。**「OK」**により確定すると、空のプロファイラーウィンドウが開きます。
4. XSL 変換を実行します（**「XSL/XQuery | XSL 変換」**）。分析を行いたい XSLT スタイルシートへのパスを選択するダイアログが表れます。変換の終了後、プロファイラーウィンドウに実行の結果が表示されます。
5. プロファイラーにある **"+"** アイコンをクリックして、XSLT スタイルシートの実行時間を表示します（以下のスクリーンショットを参照）。これらのスクリーンショットにはヒットカウントプロファイリングが使用されています。

プロファイラーにある行をクリックすると分析が行われたファイルの対応する命令がハイライトされます。

以下のスクリーンショットではプロファイラーのツリービューが示されます：



以下のスクリーンショットでは同じXSLT スタイルシートに対するプロファイラーのリストビューが示されます：



プロファイラーの情報を使用する
XSLT スタイルシートにある各命令に関する以下の情報がプロファイラーには表示されます：

- **インデックス**:各命令に割り当てられている番号。呼び出された順に番号は振り付けられます。
- **名前**:XSLT 命令の名前。
- **呼び出し回数 (ヒットカウント)** 変換においてその命令が呼び出された回数の合計。
- **経過時間 (ms)** と **%** : (子孫を計算に入れず) その命令を実行するのに要した時間をミリ秒単位で表し、全体に占める実行時間の割合が一緒に表示されます。
- **子孫 (ms)**:その命令の子孫を実行するのに要した時間をミリ秒単位で表します。
- **自己と子孫 (ms)** と **%** :その命令と、その命令の子孫を実行するのに要した時間がミリ秒単位で表示され、全体に占める実行時間の割合が一緒に表示されます。
- **XPath**:命令にXPath ステートメントが含まれている場合、そのステートメントを実行するのに要した時間が表示されます。

メモ:ヒットカウントプロファイリングを使用している場合、プロファイラーウィンドウに表示される時間は、その命令に対する全ての呼び出しに要した時間の合計になります。コールグラフプロファイリングを使用している場合、命令の各呼び出しが別々に表示されるため、プロファイラーウィンドウに表示される時間は各実行の個別に呼び出すための時間となります。

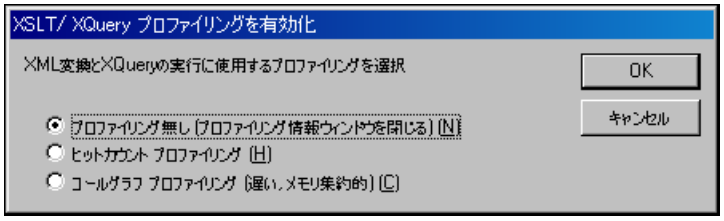
7.2.2 XQuery プロファイリング

プロファイラーの開始

注意：プロファイラーに表示される実行時間は、お使いのコンピューターで同時に動作している他のアプリケーションにより影響を受ける可能性もあります。プロファイラーを使ってファイルの分析を行うには、XMLSpy アプリケーションのみを動作させることが理想的です。

XQuery ドキュメントの分析を行うには：

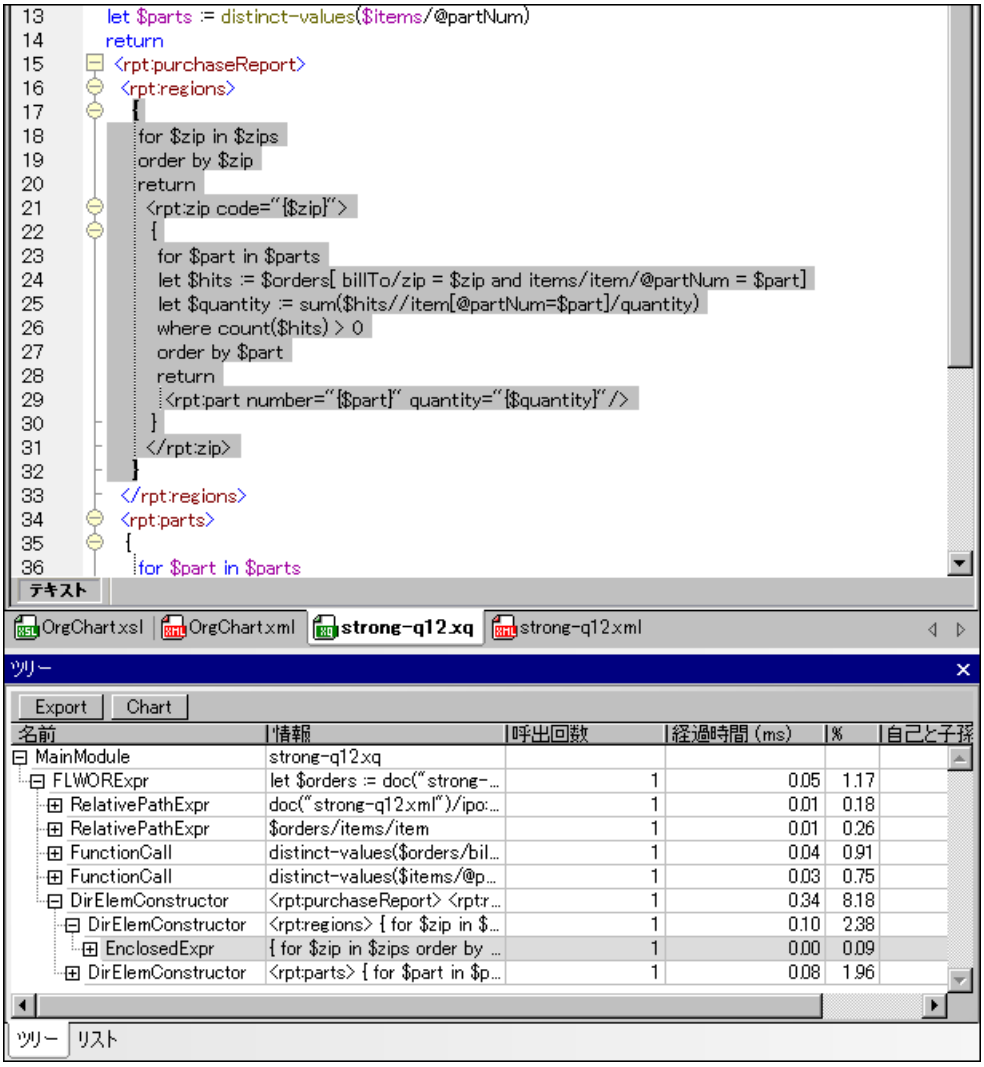
1. XMLSpy にて分析を行うXQuery ドキュメントを開きます。
2. **「SL/XQuery | XSLT/XQuery プロファイリングの有効化」**を選択します。以下のダイアログが開きます。



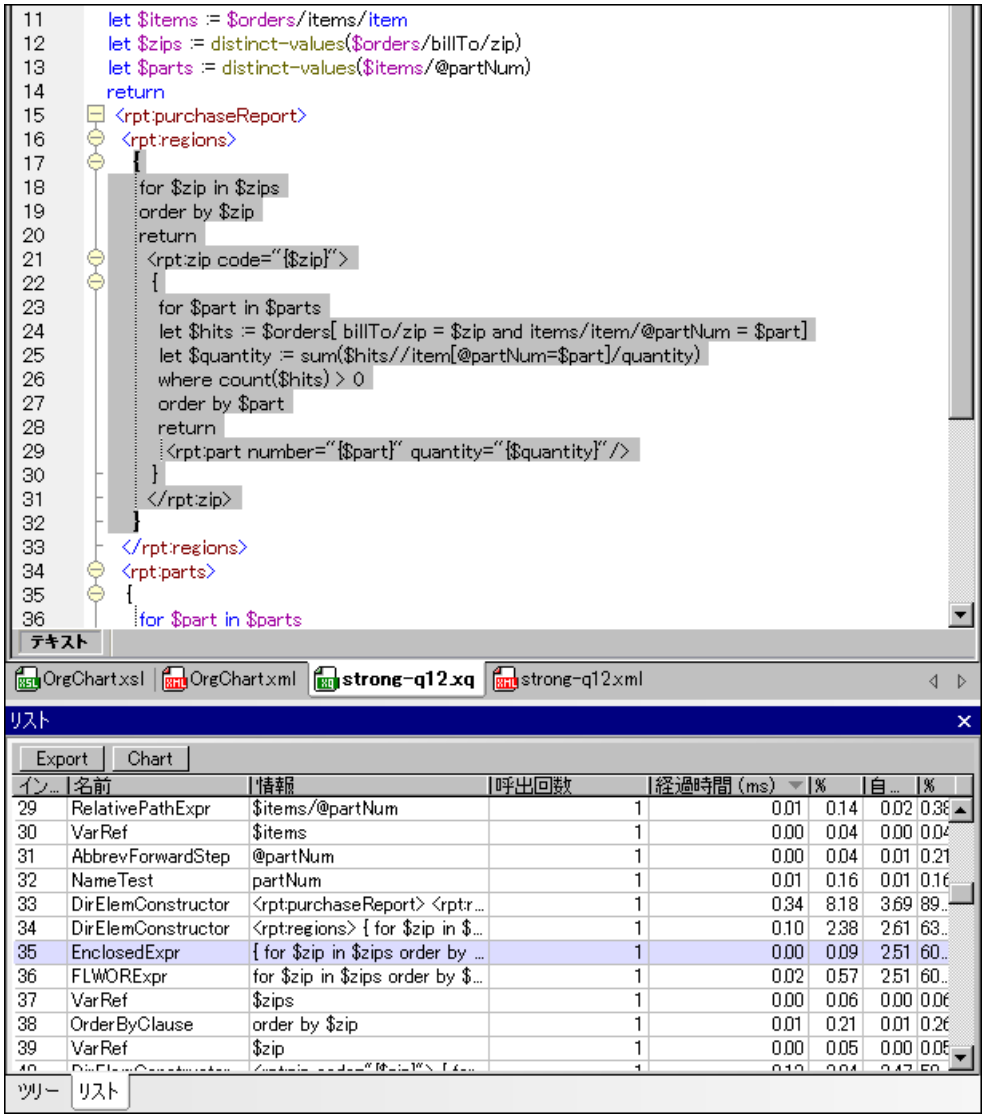
3. **「ヒットカウントプロファイリング」**または **「コールグラフプロファイリング」**を選択します。**「OK」**により確定すると 空のプロファイラーウィンドウが開きます。
4. XQuery を実行します（**「SL/XQuery | XQuery を実行」**）。実行が終了した後、プロファイラーウィンドウに実行の結果が表示されます。
5. プロファイラーにある "+" アイコンをクリックして、XQuery ドキュメントにある命令の実行時間を表示します（以下のスクリーンショットを参照）。これらのスクリーンショットには **「ヒットカウントプロファイリング」**が使用されています。

プロファイラーの行をクリックすると、分析されたファイルの対応する箇所がハイライトされます。

以下のスクリーンショットではプロファイラーのツリービューが示されます：


let $parts := distinct-values($items/@partNum)
return
<rpt:purchaseReport>
<rpt:regions>
{
for $zip in $zips
order by $zip
return
<rpt:zip code="{$zip}">
{
for $part in $parts
let $hits := $orders[billTo/zip = $zip and items/item/@partNum = $part]
let $quantity := sum($hits//item[@partNum=$part]/quantity)
where count($hits) > 0
order by $part
return
<rpt:part number="{$part}" quantity="{$quantity}" />
}
</rpt:zip>
}
</rpt:regions>
<rpt:parts>
{
for $part in $parts
テキスト
OrgChart.xsl
OrgChart.xml
strong-q12.xq
strong-q12.xml
ツリー
Export
Chart
名前
情報
呼出回数
経過時間 (ms)
%
自己と子孫
MainModule
strong-q12.xq
FLWORExpr
let $orders := doc("strong-...
1
0.05
1.17
RelativePathExpr
doc("strong-q12.xml")/po:...
1
0.01
0.18
RelativePathExpr
$orders/items/item
1
0.01
0.26
FunctionCall
distinct-values($orders/bil...
1
0.04
0.91
FunctionCall
distinct-values($items/@p...
1
0.03
0.75
DirElemConstructor
<rpt:purchaseReport> <rpt:r...
1
0.34
8.18
DirElemConstructor
<rpt:regions> { for $zip in $...
1
0.10
2.38
EnclosedExpr
{ for $zip in $zips order by ...
1
0.00
0.09
DirElemConstructor
<rpt:parts> { for $part in $p...
1
0.08
1.96
ツリー
リスト

以下のスクリーンショットでは同じXQuery ドキュメントに対するプロファイラーのリストビューが示されます：



プロファイラーの情報を使用する

XQuery ドキュメント内の各命令に関する以下の情報がプロファイラーには表示されます：

- **インデックス**:各命令に割り当てられている番号。呼び出された順に番号は振り付けられます。
- **名前**:XQuery 命令の名前。
- **情報**:命令に関する情報が表示されます。例えば、命令が変数の宣言であれば、この列にはその関数の名前と値が表示されます。命令が関数であれば、列にはその関数の名前とパラメーターが表示されます。
- **呼び出し回数（ヒットカウント）**:実行中にその命令が呼び出された合計回数を表示します。
- **経過時間（ms）と%**：子孫を計算に入れず）その命令を実行するのに要した時間をミリ秒単位で表し、全体に占める実行時間の割合が一緒に表示されます。
- **自己と子孫（ms）と%**：その命令と、その命令の子孫を実行するのに要した時間がミリ秒単位で表示され、全体に占める実行時間の割合が一緒に表示されます。

メモ:ヒットカウントプロファイリングを使用している場合、プロファイラーウィンドウに表示される時間は、その命令の全ての呼び出しに要した時間の合計になります。コールグラフプロファイリングを使用している場合、命令の各呼び出しが別々に表示さ

れるため、プロファイラーウィンドウに表示される時間は各実行の個別に呼び出すための時間となります。

7.2.3 プロファイリングの結果：エクスポートとチャート

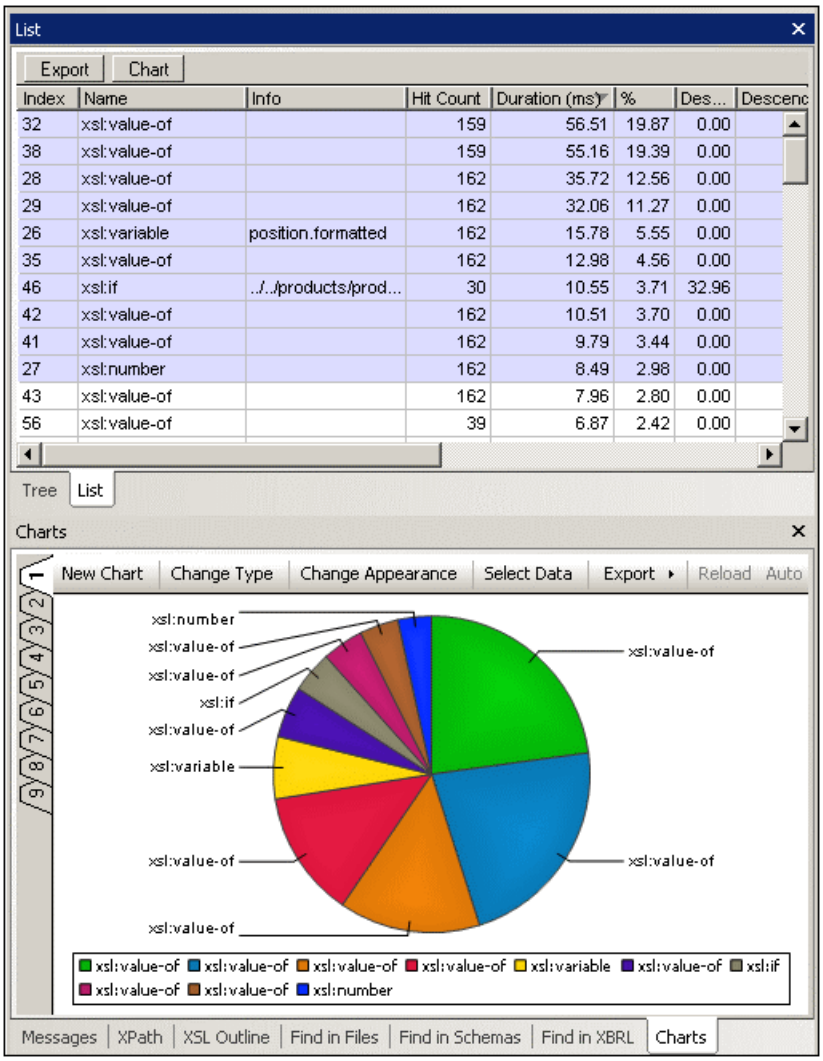
XSLT/XQuery プロファイラーを実行すると、その結果をXML へエクスポートしたり、作成されたチャートをイメージファイルとして保存することができます。

エクスポート

エクスポートボタンをクリックすると、プロファイリングの結果をエクスポートするXML ファイルの場所とファイル名を指定するよう促されます。グリッドビューを使用することで、XML ファイルの構造をよりクリアに理解することができます。例えばXSLT プロファイリングの結果をグリッドビューで標示すると、XML ドキュメントの構造が`node` 要素により分類される3つの階層レベルから構成されていることを確認することができます。最初の`node` 要素によりドキュメントルートが、2番目の`node` 要素により`xsl:stylesheet` 要素が、3番目の`node` 要素により（`xsl:output` や`xsl:template` といった）グローバル要素が表されます。プロファイリングのデータは、`node` 要素の各属性に保管されます。

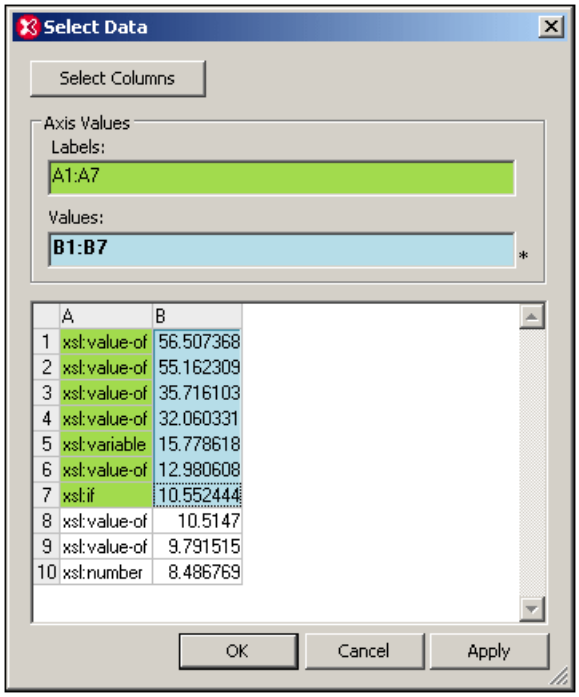
チャート

XSLT/XQuery プロファイラを実行後、実行結果やそのサブセットをグラフの形で生成することができます。以下のスクリーンショットにあるように、プロファイラウィンドウにて、**「チャート」**ボタンをクリックし、グラフをチャート出力ウィンドウに生成します。



以下の点に注意してください：

- プロファイラーウィンドウ（ツリーならびにリストビュー）にて、項目のマーキングをすることで、実行結果のサブセットを選択することができます。「Ctrl」や「Shift」キーを使って複数のアイテムを選択することができます。リストビューでは、列のヘッダーをクリックすることで、列にある値をベースに並べ替えを行うことができます。例えばこの機能を使うと、最も処理に時間を要するアイテム順に並べ替えを行った後、変換処理に最も時間を要しているアイテムだけを選択し、必要でないアイテムを結果の表示から取り除くことができます。上のスクリーンショットでは、ハイライトされたアイテムの結果だけが表示されています。
- チャートを作成した後は、チャート出力ウィンドウにある**種類の変更**ボタンをクリックして、そのチャートの種類（円グラス 棒グラス 折れ線グラフなど）を変更することができます。利用可能なチャートの種類に関する情報は、このドキュメントのチャートセクションを参照ください。
- チャート出力ウィンドウの「**データの選択**」ボタンをクリックすることで、データの選択ダイアログが表示されます（以下のスクリーンショットを参照）。このダイアログでは、列の選択プロセスで得られたデータテーブルから、X 軸ならびにY-軸にて使用されるデータの選択を行うことができます。軸のデータ選択を行うには、その軸のテキストフィールドをクリックし、テーブルの範囲（例：`A1:A7`）を入力するか、カーソルをドラッグすることで、範囲を指定します。



	A	B
1	xsl:value-of	56.507368
2	xsl:value-of	55.162309
3	xsl:value-of	35.716103
4	xsl:value-of	32.060331
5	xsl:variable	15.778618
6	xsl:value-of	12.980608
7	xsl:if	10.552444
8	xsl:value-of	10.5147
9	xsl:value-of	9.791515
10	xsl:number	8.486769

「**列の選択**」ボタンをクリックすることで、データテーブルに表示されるデータの選択を変更することができます。列の選択に関しては、ソースXPath、X-軸の選択、Y-軸の選択を参照ください。

チャートに関するより詳細な情報については、ドキュメントのチャートセクションを参照ください。

8 Authentic

(以下のスクリーンショットにある) Authentic View はお使いの XML ドキュメントをグラフィカルな形式で表示したものです。XML ドキュメントのマークアップの表示を隠し、入力フィールド、コンボボックス、ラジオボタンといったフォーマットやデータ入力機能を適切な形で表示します。Authentic View を使ってユーザーが入力したデータは XML ファイルにも反映されます。

Authentic View でデータの閲覧と入力を行うためには、Altova StyleVision により作成された **StyleVision Power Stylesheet (SPS)** に XML ドキュメントを関連付ける必要があります。SPS (sps ファイル) は、実質的には XSLT スタイルシートで、データ入力を行うことができる XML ファイルの出力フォーマットを指定することができます。Authentic View ユーザーは、XML ファイルや DB へのデータ入力をグラフィカルに行うことができます。SPS はスキーマをベースにしており、スキーマに特化したファイルです。Authentic View ビューで SPS を使った XML ファイルの編集を行うには、XML ファイルがベースにしているスキーマファイルを使用しなければなりません。

Authentic View を使う

- XML ファイルが開かれている場合、メインウィンドウの下部にある **Authentic** ボタンをクリックすることで Authentic View に切り替えることができます。SPS が XML ファイルにまだ割り当てられていない場合、割り当てを行うよう促されます。使用できる SPS は、XML ファイルがベースにしているスキーマをベースにしたもので無ければなりません。
- **「ファイル | 新規作成」** コマンドを選択し、[StyleVision スタイルシートを選択] ボタンを選択すると、新規 XML ファイルが作成され、Authentic View で表示されます。この新しいファイルは、選択した SPS ファイルに割り当てられたテンプレートファイルになります。このファイルは、既に表示されている様に様々な数の開始データを持つことができます。この開始データは (テンプレート) XML ファイルに含まれ、SPS ファイルと関連付けることもできます。XML ファイルの Authentic View が表示されたら、データの入力とファイルの保存を行うことができます。
- **「Authentic | 新規作成」** コマンドを使用して SPS を開くことができます。テンプレート XML ファイルが SPS に割り当てられている場合、テンプレート XML ファイル内のデータは、Authentic View により作成された XML ドキュメントテンプレートのデータを開始するために使用されます。

このセクションでは ...

このセクションは Authentic ビューのチュートリアルを含んでおり、Authentic View の使い方を説明した後、Authentic ビューでの編集や個々の編集機能の詳細について説明します。

Authentic View について更に詳しく

Authentic View に関する更に詳しい情報については、(i) Authentic View 編集ウィンドウについて解説された、このドキュメントのAuthentic ウィンドウインターフェイスを参照するか、(ii) このドキュメントのユーザーリファレンスにある Authentic メニューセクションを参照ください。

8.1 Authentic View チュートリアル

Authentic View では、Microsoft Word に代表されるワードプロセッサーアプリケーションと同様に、グラフィカルな WYSIWYG インターフェース (以下のスクリーンショットを参照) によりXML ドキュメントを編集することができます。フォーマットは XML ドキュメントの Authentic View を制御するスタイルシートにより既に定義されているため、ドキュメントのフォーマットについて心配する必要はありません。 StyleVision Power Stylesheet StyleVision Power Stylesheetという名前で、このチュートリアルでは StyleVision Power Stylesheet と省略される) スタイルシートは、Altova の StyleVision を使ったスタイルシートデザイナーにより作成されます。

Authentic View における XML ドキュメントの編集には2つのユーザーアクションがあります： (i) ドキュメントの構造を編集 (例えば、段落や見出しといったドキュメントのパーツを追加、削除)、そして (ii) データ (ドキュメントパーツのコンテンツ) を入力。

このチュートリアルでは、以下の項目について説明します：

- Authentic View にて XML ドキュメントを開く。Authentic View で編集を行うには SPS が割り当てられている必要があります。
- Authentic View インターフェースの外観と、編集メカニズムの大まかな説明。
- ノードの挿入や削除によりドキュメントの構造を編集。
- XML ドキュメントにデータを入力。
- (i) 属性入力ヘルパーにて属性値の入力、そして (ii) エンティティ値の入力
- ドキュメントの印刷

このチュートリアルは、Authentic View を使い始めるために作成されており、あえてシンプルにまとめられています。より詳細なリファレンスや機能に関する記述は Authentic View インターフェイス を参照ください。

チュートリアルに必要なもの

このチュートリアルで必要な全てのファイルは Altova アプリケーションフォルダーの Examples フォルダーに収められています。必要なファイルは以下のとおりです：

- NanonullOrg.xml (開くことになる XML ドキュメント)
- NanonullOrg.sps (XML ドキュメントがリンクしている StyleVision Power Stylesheet)
- NanonullOrg.xsd (XML ドキュメントならびに StyleVision Power Stylesheet がベースになり、リンクされ

ているXML スキーマファイル)
- nanonull.gif ならびに Altova_right_300.gif (チュートリアルでは2つのイメージファイルが使用されます)

メモ: チュートリアルの一部では、XML ドキュメントを (ドキュメントの Authentic View ではなく) XML テキストとして参照することになります。お使いの Altova 製品がテキストビューをサポートしない場合 (例:Authentic Desktop や Authentic Browser)、メモ帳やワードパットといったテキストエディターを使用して XML ドキュメントを参照してください。

注意: オリジナルの状態に戻る必要が生じたときのために、チュートリアルで使用する NanonullOrg.xml はオリジナルのコピーを使用することをお勧めします。

8.1.1 Authentic View にて XML ドキュメントを開く

Authentic View では既存の XML ドキュメントを編集したり、新たな XML ドキュメントを作成、編集することができます。このチュートリアルでは、既存の XML ドキュメントを <Authentic View で開き (このセクション)、その編集を行う方法を以降のセクションで学習します。更にこのセクションでは、Authentic View で編集することのできる XML ドキュメントの作成方法を紹介します。

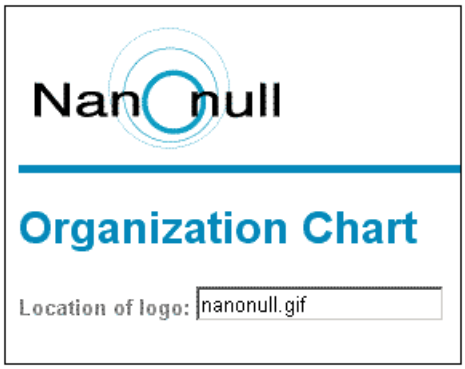
既存の XML ドキュメントを開く

Altova アプリケーションの Examples フォルダーにあるファイル NanonullOrg.xml を開いてください。以下の2種類の方法で NanonullOrg.xml を開くことができます:

- Altova 製品の **「ファイル | 開く」** をクリックし、表示されるダイアログにて NanonullOrg.xml を選択し、**開く** をクリックします。
- Windows のエクスプローラーを使用してファイルを選択、右クリックで表示されるメニューから起動するアプリケーションとして Altova 製品を選択します。

NanonullOrg.xml は直接 Authentic View で開かれます (以下のスクリーンショットを参照)。これは:

1. ファイルは既に SPS (SPS) が割り当てられており、更に
2. オプションダイアログ (**ツール | オプション**) のビュータブにて、SPS ファイルが割り当てられている場合、Authentic View で開くよう設定されているためです (そうでない場合、ファイルはテキストビューで開かれます)



メモ SPS により、Authentic View においてどのように XML ドキュメントが表示されるかが定義されます。SPS が存在しない場合、ドキュメントを Authentic View にて表示することはできません。

SPS をベースにした XML ドキュメントを新たに作成する

SPS をベースにした XML ドキュメントを新たに作成することもできます。**「ファイル | 新規作成」** メニューコマンドからの作成と、**「Authentic | 新規ドキュメント」** メニューコマンドを使う方法があります。両方の場合で SPS が選択されます。

「ファイル | 新規作成」

1. 「ファイル | 新規作成」を選択し、新規ドキュメントを作成ダイアログにて XML を選択します。

2. StyleVision スタイルシートを選択をクリックして、目的の SPS を選択します。

Authentic | 新規ドキュメント

1. **「ファイル | 新規作成」**を選択します。
2. **StyleVision スタイルシートを選択**をクリックして、目的のSPS を選択します。

テンプレートXML ファイルがSPS に割り当てられている場合、テンプレートXML ファイル内にあるデータがXML ドキュメントテンプレートの開始点として使用され、Authentic View にて作成されます。

8.1.2 Authentic View インターフェース

Authentic View 編集ビューは、ドキュメントのデータを入力ならびに編集するメインウィンドウと、3つの入力ヘルパーから成ります。ドキュメントの編集は簡単に行うことができ、ドキュメントのマークアップを参照するには、マークアップタグを切り替える必要があります。ドキュメントのコンテンツ編集はキーボードから行うことができ、ドキュメントの構造を変更するには、コンテキストメニューまたは要素入力ヘルパーを使用します。

XML ノードタグ（ドキュメントマークアップ）の表示

XML ドキュメントの根幹はノードから成る階層になります。以下に例を示します：

```
<DocumentRoot>
  <Person id="ABC001">
        <Name>Alpha Beta</Name>
        <Address>Some Address</Address>
        <Tel>1234567</Tel>
  </Person>
</DocumentRoot>
```

デフォルトではAuthentic View にノードタグは表示されません。メニューアイテムの **Authentic | 大きなマークアップを表示**」または ツールバーアイコン）を選択することでノードタグを表示することができます。大きなマークアップには、対応するノードの名前が表示されます。更に、小さなマークアップ（ノード名の表示なし）や混合マークアップを選択することもできます（混合マークアップではスタイルシートのデザイナーにより定義された通りにマークアップが表示されます。デフォルトではマークアップ無しが使用されます）。

Altova 製品のテキストビュー、またはテキストエディターにてXML ドキュメントのテキストを確認することができます。

入力ヘルパー

デフォルトでは、アプリケーションウィンドウの右側に3つの入力ヘルパーが配置されます（以下のスクリーンショットを参照）。それぞれ要素入力ヘルパー、属性入力ヘルパー、エンティティ入力ヘルパーと呼ばれます。

要素入力ヘルパー:要素入力ヘルパーには、メインウィンドウ内にある現在カーソルの位置や選択に対して、挿入または削除することのできる要素が表示されます。要素入力ヘルパーに表示されるコンテンツは現在カーソルの位置または選択により変化することに注目してください。要素入力ヘルパー内のXML ツリーにある別ノードが選択されると、そのノードに対応する要素が入力ヘルパーに表示されます。入力ヘルパーの上部にあるXML ツリーを表示チェックボックスを有効にすることで、要素入力ヘルパーをXML ツリーが表示されるよう展開することができます (上のスクリーンショットを参照)。XML ツリーには、トップレベルの要素ノードから、現在メインウィンドウで選択されているノードまでの階層が表示されます。

属性入力ヘルパー:属性入力ヘルパーにはメインウィンドウで選択されている要素の属性が表示されます。属性入力ヘルパーではこれら属性の値の入力ならびに編集を行うことができます。トップレベルの要素から現在選択されている要素における要素ノードが属性入力ヘルパーのコンボボックスに表示され、選択することができます。コンボボックスのドロップダウンリストから要素を選択することで、その要素の属性が入力ヘルパーに表示され、編集することができるようになります。

エンティティ入力ヘルパー:エンティティ入力ヘルパーの表示内容は状況に応じて変化せず、ドキュメントに対して宣言された全てのエンティティが表示されます。エンティティをダブルクリックすることで、そのエンティティがカーソル位置に挿入されます。ドキュメントに対してエンティティを追加する方法については、Authentic View インターフェースを参照ください。

コンテキストメニュー

Authentic View コンテキストメニューマークアップタグの表示入力ヘルパーに表示されているドキュメントを右クリックすることで、ノードの位置に関連した内容のコンテキストメニューが表示されます。コンテキストメニューに表示されるコマンドにより、以下のような操作を行うことができます:

- 選択されたノードの位置 (またはその前後)、に新たなノードを挿入する。サブメニューにはそれぞ

れの挿入位置で許されているノードのリストが表示されます。
- (スキーマにより許される場合) 選択されたノードを削除したり、削除可能な祖先要素を削除する。スキーマにより削除することが許されるノードがサブメニューに表示されます。
- エンティティならびに CDATA を挿入する。ドキュメントに対して宣言されたエンティティがサブメニューに表示されます。CDATA セクションはテキストの内部にしか挿入することができません。
- ドキュメントのコンテンツを切り取り、コピー、 (XML またはテキストとして) 貼り付けます。

メモ: インターフェースに関する詳細は、Authentic View インターフェースを参照ください。

8.1.3 ノードオペレーション

Authentic View XML ドキュメントにて表示されるノードには、**要素ノード**ならびに**属性ノード**という2種類のノードがあります。これらのノードは、切り替え可能なタグにより表されます。テキストノード (マークアップは表示されません) や、CDATA セクションノード (その他のテキストと区切るためのマークアップが表示されます) といったその他のノードもドキュメント内には含まれます。

このセクションで記述されるノードオペレーションは、要素ノードならびに属性ノードを前提としたものです。このセクションで記述されるオペレーションを実行する場合、大きなマークアップを有効にしておいたほうが良いでしょう。

メモ: 選択された要素の前後には、**同レベルまたはより高いレベルの要素**しか挿入できないことに注意してください。同レベルの要素とは**兄弟**位置にある要素のことです。段落要素の兄弟は他の段落や、リスト、テーブル、イメージなどとなります。兄弟要素は、要素の直前または直後に出現します。より高いレベルの要素とは**祖先**要素やその兄弟のことです。段落要素に対する祖先要素はセクション、章、文章などとなります。妥当な XML ファイル内の段落は既に祖先を持っており、Authentic View にて高レベルの要素を追加すると、対応する祖先の兄弟として新たな要素が作成されます。例えば、セクション要素が段落の後に挿入されると、新たなセクションが現在段落要素があるセクションの兄弟として作成され、兄弟セクション位置に配置されます。

ノードオペレーションの実行

ノードオペレーションはコンテキストメニューからコマンドを選択することで、または要素入力ヘルパーのノードオペレーションエントリーをクリックすることで実行できます。場合によっては、ドキュメントの Authentic View にあるノードリンクを追加 をクリックすることで要素や属性を追加することもできます。段落やリストアイテムとして定義された要素の場合、要素内で Enter キーを押下することで、同じ種類の兄弟要素が作成されます。このセクションでは要素の適用、ノードの削除 および要素のクリア といったメカニズムを使用したノードの作成や削除について記述します。

要素の挿入

以下の位置にて要素の挿入を行うことができます:

- 要素内にあるカーソル位置。その位置にて挿入可能な要素が、コンテキストメニューにて表示される**挿入**コマンドのサブメニューにて表示されます。カーソルがある位置で挿入することが可能な要素は、要素入力ヘルパーにて アイコンで示されます。NanonullOrg.xml ドキュメントにおいて、カーソルを para 要素内部に移動し、コンテキストメニューと要素入力ヘルパーを使って bold ならびに italic 要素を作成します。
- 選択された要素の前後、または (Ctrl キーにて許されている場合) その祖先。表示されるサブメニューから目的の要素を選択します。選択された要素の前または後に挿入することのできる要素は、要素入力ヘルパーにてそれぞれ と アイコンで示されます。要素入力ヘルパーからは、選択された要素の前後にしか要素を入力できないということに注意してください (祖先要素の前後に要素は挿入されません)。まず para 要素内部で、次に従業員が表示されているテーブル内部でこの操作を行ってください。

ノードリンクの追加

要素または属性がドキュメントデザインに加えられているがXML ドキュメント内に表示されていない場合、ドキュメント内のノードが指定された箇所に**ノードを追加**リンクが表示されます。このリンクを表示するには、*Location of logo* の隣にある CompanyLogo 要素内のhref 属性を削除します（削除は**Delete** キーを押下することで行えます）。@href を追加というリンクが、編集されたCompanyLogo 内に表れます（以下のスクリーンショットを参照）。リンクをクリックすることで@href ノードがXML ドキュメントに追加されます。@href ノードの追加方法がデザインにて指定されているため、@href タグ内にテキストボックスが表れます。@href ノードに対して値（またはコンテンツ）を入力しなければなりません。nanonull.gif というテキストを入力してください。

例えば子要素のシーケンスがどのような順序でも出現可能というように、要素のコンテンツモデルが曖昧な場合、追加 ... というリンクが表示されます。ノード名が指定されていないことに注目してください。リンクをクリックすることで、挿入可能な要素のリストが表示されます。

メモ: ノードを追加リンクはドキュメントテンプレートに直接表示されます。コンテキストメニューや要素入力ヘルパーに対応するエントリーは表示されません。

Enter キーにより新たな要素を作成

（スタイルシートデザイナーにより）要素が段落またはリストアイテムとして定義されているような場合では、そのようなノード内で Enter キーを押下することで、同じ種類の新しいノードが現在のノードの後に挿入されます。NanonullOrg.xml ドキュメントにあるpara ノードの最後（終了タグの手前）で**Enter** キーを押下することで、このメカニズムを確認することができます。

要素の適用

（テキストと子要素の両方を含む）混合コンテンツの要素によっては、テキストコンテンツを選択して、そのテキストに対して子要素を適用することができます。選択されたテキストが適用された要素のコンテンツとなります。要素を適用するには、コンテキストメニューから適用を選択し、許されている要素から目的の要素を選択します。選択されたテキストに対して適用できる要素がない場合、適用コマンドはコンテキストメニューに表示されません。選択されたテキストに対して適用可能な要素は、要素入力ヘルパーにて アイコンで示されます。NanonullOrg.xml ドキュメントにて、混合コンテンツのpara 要素内のテキストを選択し、bold ならびにitalic 要素を適用してみてください。

スタイルシートのデザイナーは要素を適用するアイコンをツールバーに作成することもできます。NanonullOrg.xml ドキュメントでは、アプリケーションのAuthentic ツールバーに表示されているbold ならびにitalic アイコンをクリックすることで、bold やitalic 要素を適用することができます。

ノードの削除

そのノードが削除されてもドキュメントの妥当性が保たれる場合、ノードを削除することができます。ノードを削除することで、そのノードと、そこにある全てのコンテンツが削除されます。ノードの削除はコンテキストメニューの削除コマンドにより行うことができます。削除コマンドがハイライトされると、削除可能なノードの一覧が、現在選択されているノードとトップレベルまでのノードがサブメニューに表示されます。削除するノードを選択するには、ノード内にカーソルを移動させるか、ノード（またはその一部）をハイライトします。削除することのできるノードは、要素入力ヘルパーにて アイコンにより示されます。ノードを選択し、**Delete** キーを押下することでもノードを削除することができます。NanonullOrg.xml ドキュメントにおいて、ここで記述された方法を使ったノードの削除を試してみてください。**Ctrl+Z** により行った操作を元に戻すことができます。

要素のクリア

ユーザーは (テキストと子要素からなる) 混合コンテンツの要素内にある子要素をクリアすることができます。ノードが選択されている時やカーソルがノードの内部に位置している場合、要素全体がクリアされます。要素内にあるテキストフラグメントをハイライトすることで、要素マークアップをクリアすることができます。選択が行われた状態で、コンテキストメニューの**クリア**からクリアする要素を選択します。選択された対象からクリアすることができる要素は、要素入力ヘルパーにて アイコン (挿入ポイント選択) または アイコン (範囲選択) で示されます。NanonullOrg.xml ドキュメントにて、para 要素 (混合コンテンツ) の bold ならびに italic 子要素をクリアしてみてください。

テーブルとテーブル構造

Authentic Viewテーブルには2つの種類があります：

- *SPS テーブル (静的ならびに動的)* : SPS テーブルの大まかな構造はスタイルシートデザイナーにより決定されます。既に定義されている大まかな構造の内部で編集できる箇所はコンテンツに関する部分のみで、例えば動的 SPS テーブルに新たな行を追加することができます。
- *XML テーブル* : 特定のノードにあるコンテンツ (例えば、人物に関する詳細など) をテーブルとして表示することができます。あるノードが XML テーブルとして作成できるようにとスタイルシートのデザイナーにより定義された場合、テーブルの構造を決定し、コンテンツの編集を行うことができます。XML テーブルに関する詳細は Authentic View におけるテーブル を参照ください。

8.1.4 Authentic View におけるデータの入力

データの入力はAuthentic View のメインウィンドウから直接行われます。属性に対しては更に、データ (属性の値) を 属性入力ヘルパーから入力 こともできます。データの入力は、(i) テキストとして直接、または (ii) 定義済みの値にマッピングされたデータ入力デバイスからオプションを選択することで行うことができます。

テキストコンテンツを追加する

ユーザーはAuthentic View のメインウィンドウから要素のコンテンツや属性の値を直接入力することができます。コンテンツを入力するには、テキストの挿入を行いたい箇所にカーソルを移動させ、入力を行います。クリップボードからドキュメントへテキストのコピーを行うこともできます。また、**CapsLock** キーや **Delete** キーといった一般的な編集メカニズムを使用してコンテンツの編集を行うこともできます。例えば、編集するテキストをハイライトして、**CapsLock** キーが有効になっている状態でテキストの置き換えを行うことができます。

例えば会社の名前を変更する場合、Office の Name フィールドにて Nanonull の直後にカーソルを移動させ、USA と入力して名前を Nanonull, Inc. から Nanonull USA, Inc. に変更することができます。

テキストが編集可能であれば、カーソルを移動させハイライトすることができます (それができなければ編集できないということです)。アドレス帳にある "Street" や "City"、または "State/Zip" といった項目の**フィールド名** (対応する値ではありません) を変更しようとしても、それらのテキストは XML コンテンツでは無いため編集することはできないはずです。これらの値は StyleVision Power Stylesheet により与えられます。

特殊な文字やエンティティを挿入する

以下に示される特殊な種類のコンテンツは特別な方法で入力されます：

- *XML マークアップに使用される特別な文字 (アンパサンド、アポストロフィ、大なり、小なり、引用符)*。これらの文字列は 内蔵のエンティティ として既に用意されており、エンティティのヘルパーに表示されているエンティティをダブルクリックすることで、対応するエンティティをドキュメントに入力することができます。例えばブログ

ラムコードのリストといった状況でこれらの文字が頻繁に使用される場合、CDATA セクション内でこれらの値を入力することができます。CDATA セクションを挿入するには、目的の場所で右クリックし、コンテキストメニューから **CDATA セクションを挿入** を選択します。CDATA セクション内の XML マークアップは XML プロセッサーにより処理されません。そのため CDATA セクション内で特別な文字を入力する際には、エンティティのリファレンスではなく直接その文字を入力するように注意しなければなりません。

- *キーボードから入力できない特殊な文字* は、お使いのシステムにある文字コード表から目的の場所に貼り付けることで入力することができます。
- *頻繁に使用されるテキストの文字列* は エンティティとして定義して、エンティティ入力ヘルパーに表示させることができます。目的の位置にカーソルを移動させ、エンティティ入力ヘルパーに表示されているエンティティのをダブルクリックすることで、エンティティの挿入 を行うことができます。この機能により、テキスト文字列の値が一箇所で管理されることになり、より良いメンテナンスを行うことができるようになります。変更が必要となったときに、エンティティの定義を変更することにより対応することができます。

メモ: Authentic View にてマークアップが非表示になっていると、空要素（コンテンツを持たない要素）を見落としてしまう事がしばしば起こります。空要素を見落とさないようにするには、大きなまたは小さなマークアップを有効にしてください。

ここで紹介された種類のテキストコンテンツ全てを使用してみて下さい。

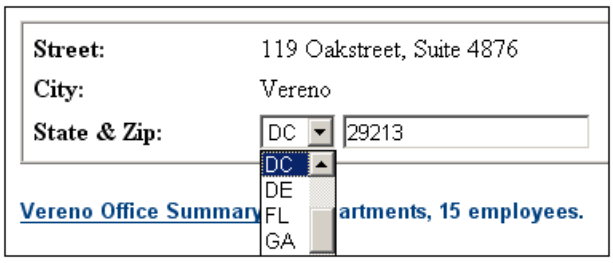
データ入力デバイスからコンテンツを追加する

上で紹介したコンテンツの編集では、テキストをコンテンツとして直接入力していました。Authentic View では、他にも**要素コンテンツ**（または属性の値）を入力する方法が用意されています：データ入力デバイスを使用します。

以下のリストでは Authentic View におけるデータ入力デバイスと、各デバイスにおいてデータがどのように XML ファイル内に格納されるのか示します。

データ入力デバイス	XML ファイル内のデータ
入力フィールド（テキストボックス）	ユーザーにより入力されたテキスト
複数行の入力フィールド	ユーザーにより入力されたテキスト
コンボボックス	値にマッピングされたユーザーの選択
チェックボックス	値にマッピングされたユーザーの選択
ラジオボタン	値にマッピングされたユーザーの選択
ボタン	値にマッピングされたユーザーの選択

以下に示されるアドレスフィールドを含む静的なテーブルでは、郵便番号を入力するフィールドと、米国の州を選択するコンボボックスという2つのデータ入力デバイスが使用されています。テキストフィールドに入力された値は、XML コンテンツとして対応する要素に直接入力されます。その他のデータ入力デバイスでは、予め定められた値に対してマッピングが行われます。



上の Authentic View で示されるデータは、以下のような XML テキストとなります：

```
<Address>
  <postreet>119 Oakstreet, Suite 4876</postreet>
```

```
    <ipo:city>Vereno</ipo:city>
    <ipo:state>DC</ipo:state>
    <ipo:zip>29213</ipo:zip>
  </Address>
```

コンボボックスにて選択されたDC が、DC という値でマッピングされていることに注目してください。Zip フィールドの値は、ipo:zip 要素のコンテンツとして直接入力されています。

8.1.5 属性値の入力

属性とは要素のプロパティで、要素内に含まれる属性の数に制限はありません。それぞれの属性には値が与えられます。属性の値としてXML のデータを入力されるよう求められることもあります。Authentic View では2つの方法で属性の入力を行うことができます：

- メインウィンドウのコンテンツとして（この方法で値の入力を行えるよう属性が作成された場合）
- 属性入力ヘルパーにて

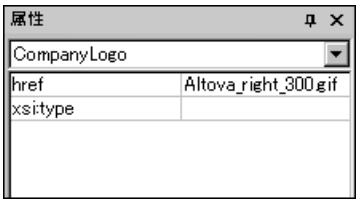
メインウィンドウ内の属性値

属性値の入力は、通常のテキスト入力フィールドのテキスト、またはXML の値にマッピングされている選択を通して行うことができます。属性の値は要素のコンテンツと同じような方法により入力することが可能です（Authentic View におけるデータの入力を参照ください）。この場合、要素コンテンツと属性の値の違いは、StyleVision Power Stylesheet により決定され、それに従いデータは処理されます。

属性入力ヘルパー内の属性値

属性入力ヘルパーを使って属性の値を入力または変更することもできます。属性のノードをAuthentic View で選択し、属性入力ヘルパーにて属性の値を入力または編集します。NanonullOrg.xml ドキュメントにおいてロゴイメージの場所はCompanyLogo 要素のhref 属性により指定されます。使用するロゴを変更するには：

1. CompanyLogo タグをクリックして、CompanyLogo 要素を選択します。CompanyLogo 要素の属性が属性入力ヘルパーに表示されます。
2. 属性入力ヘルパーにて、href 属性の値をnanonull.gif からAltova_right_300.gif（Examples フォルダー内にあるイメージ）に変更します。

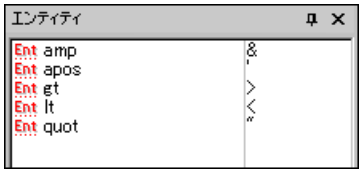


この操作によりNanonull ロゴが Altova ロゴに置換えられます。

メモ： 属性入力ヘルパーにてエンティティを入力することはできません。

8.1.6 エンティティの追加

通常 Authentic View 内のエンティティは、単一文字やテキスト文字列、またはXML ドキュメントのフラグメントといったXML データとなります。エンティティはイメージファイルのようなバイナリファイルであることもあります。ドキュメント内で使用することのできる全てのエンティティはエンティティ入力ヘルパーに表示されます（以下のスクリーンショット参照）。エンティティを挿入するには、ドキュメント内の挿入したい場所にカーソルを移動させ、エンティティ入力ヘルパーに表示されているエンティティをダブルクリックします。属性入力ヘルパーにはエンティティを入力できないことに注意してください。



アンパサンド文字 (&) は、アポストロフィ、大なり小なり、引用符と同様に、XML にて特別な意味を持ちます。XML に関連した文字を XML 上の意味を持たせない形で入力するためにもエンティティは使用されます。これらのエンティティは Authentic View にて使用することができます。

NanonullOrg.xml にて、「Marketing にある」Joe Martin の Marketing Manager Europe を、Marketing Manager Europe & Asia に変更しましょう。以下の操作を行って下さい：

1. アンパサンドが挿入されるべき箇所にカーソルを移動します。
2. "amp" と表示されているエンティティをダブルクリックします。これでアンパサンドが挿入されます（以下のスクリーンショットを参照）

メモ： エンティティ入力ヘルパーは状況に応じて変化しません。カーソルの位置に関わらず、利用可能な全てのエンティティがエンティティ入力ヘルパーに表示されます。しかしこれはドキュメント内のあらゆる箇所で全てのエンティティを入力できることを意味していません。カーソルがある位置へエンティティの挿入が許されるかどうか分からない場合、実際に挿入を行った後、「**XML | XML 検証 XML (F8)**」によりドキュメントの検証を行ってください。

独自のエンティティを定義する

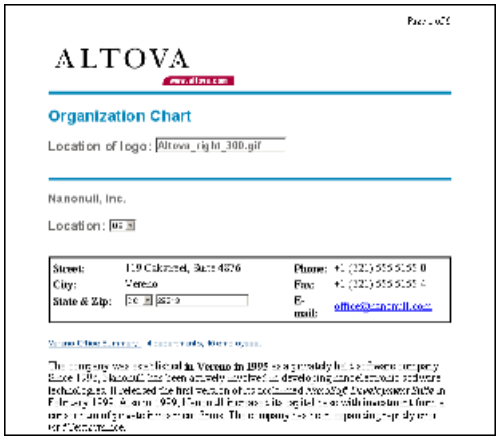
ドキュメントの編集者は独自のドキュメントエンティティを定義することもできます。エンティティの定義方法については、Authentic View において エンティティを定義する を参照ください。

8.1.7 ドキュメントの印刷

Authentic View から XML ドキュメントを印刷すると、Authentic View にて表示された表示書式が保たれた形で印刷が行われます。

NanonullOrg.xml を印刷するには：

1. マークアップを隠すモードに切り替えます。この操作によりマークアップの印刷が行われなくなります。
2. 「**ファイル | 印刷プレビュー**」を選択して、前ページのプレビューを確認します。以下は 50% に縮小された印刷プレビューの一部です。



ページのフォーマットがAuthentic View と同一であることにご注目してください。

3. To print the file, click **File | Print**.

マークアップを表示した状態でも印刷を行えることにご注意してください。Authentic View にてマークアップを表示するモードに切り替え、印刷を行います。

8.2 Authentic View インターフェイス

Authentic View は、アクティブなドキュメントのAuthentic にタブをクリックすることで有効になります。XML ドキュメントにSPS が割り当てられてる場合、割り当てを行うように促されます。SPS の割り当ては、**Authentic | SPS ファイルの割り当て**」コマンドにいつでも行うことができます。.

このセクションでは以下の事柄情報が提供されます：

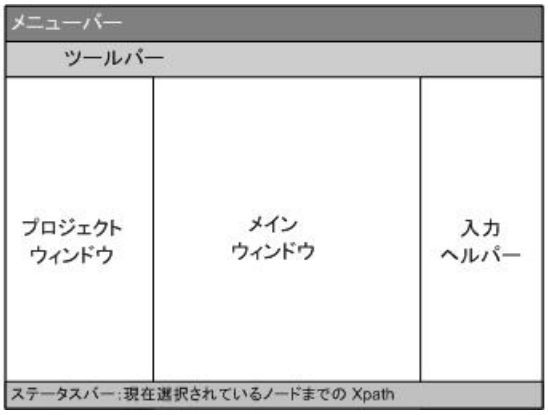
- インターフェースの概要
- Authentic View 独自のツールバーアイコン
- Authentic View ウィンドウにて利用可能な表示モード
- 入力ヘルパーならびにその使用方法
- XML ドキュメントのAuthentic View にて利用することができる様々なコンテキストメニュー

以下のリソースにてAuthentic View に関する更に詳しい情報を入手することができます：

- Authentic View インターフェースについて解説したAuthentic View チュートリアル。このチュートリアルは、Altova XMLSpy ならびにAltova Authentic Desktop 製品のドキュメンテーション (チュートリアルセクションを参照ください)、またはオンラインにてご利用になれます。
- For a detailed description of Authentic View 詳細については、製品のドキュメンテーションにあるユーザーリファレンスを参照ください。

8.2.1 GUI の概要

Authentic View ではメニューバーならびにツールバーがウィンドウの上部に配置されており、以下に示すように、プロジェクトウィンドウ、メインウィンドウ、入力ヘルパーウィンドウの3つの領域が他のインターフェース部分を占めています。



メニューバー

メニューバーから使用することのできるメニューに関する詳細は、製品ドキュメンテーションのユーザーリファレンスを参照ください。

ツールバー

ツールバーに表示されているシンボルやアイコンに関する情報はAuthentic View ツールバーアイコンを参照ください。

プロジェクトウィンドウ

ユーザーはXML XSL、XML スキーマ、エンティティファイルをプロジェクト内でまとめることができます。プロジェクトファイルのリストを作成、変更するには、プロジェクトメニュー以下にあるコマンドを使用します (詳細は製品ドキュメンテーションのユーザーリファレンスを参照ください)。プロジェクトウィンドウには、プロジェクトファイルのリストが表示されます。プロジェクトウィンドウに表

示されているファイルをダブルクリックすることで、そのファイルにアクセスすることができます。

メインウィンドウ
XML ドキュメントを表示、編集するウィンドウです。詳細は Authentic View メインウィンドウ を参照ください。

入力ヘルパー
この領域には、要素、属性、エンティティという3つの入力ヘルパーウィンドウが表示されます。これら 要素ならびに属性入力ヘルパー ウィンドウに表示されるアイテムは状況に応じて変化し、カーソルがある位置に応じて表示内容が変更されます。入力ヘルパーをダブルクリックすることで、要素やエンティティをドキュメント内に挿入することができます。属性の値は、属性入力ヘルパー内にある属性の値フィールドに入力されます。詳細は Authentic View 入力ヘルパー を参照ください。

ステータスバー
ステータスバーには現在選択されているノードへの XPath が表示されます。

コンテキストメニュー
メインウィンドウにて右クリックすることにより表示されるメニューのことです。表示されるコマンドは状況に応じて変化し、選択されたノードに関連する構造やコンテンツの編集を行うことができます。ノードの挿入、追加や削除、エンティティの追加、コンテンツの切り取りや削除といった操作を行うことができます。

8.2.2 Authentic View ツールバーアイコン

Authentic View ツールバーにあるアイコンはコマンドへのショートカットとなっています。いくつかのアイコンは他の Windows アプリケーションや Altova 製品で既に馴染みのあるものでしょう。このセクションでは Authentic View。独自のアイコンについて説明します。以下の説明では、関連するアイコンがグループとして紹介されます。

XML マークアップを表示 隠す
Authentic View では、XML の要素や属性を表すタグの表示を ,名前付き 大きなマークアップ または名前なし 小さなマークアップ から選択することができます。ツールバーには4つのマークアップアイコンが表示され、対応するコマンドも **Authentic** メニューから選択することができます。

マークアップを隠す。縮退されたものを除き、全てのタグが非表示になります。縮退されたタグをダブルクリックすることで、ノードのコンテンツが表示され、タグが非表示になります。

小さなマークアップを表示。XML の要素と属性のタグが名前なしで表示されます。

大きなマークアップを表示。XML 要素 属性タグが名前とともに表示されます。

混合マークアップを表示。StyleVision Power Stylesheet では、XML の要素や属性を個別に大きなマークアップや小さなマークアップで表示、または非表示するよう指定することができます。表示するマークアップや非表示にするマークアップを、各要素に対して指定することができるため、混合マークアップと呼ばれています。そのため、混合マークアップモードにおいて Authentic View ユーザーは、カスタマイズされたマークアップを閲覧することになります。カスタマイズは StyleVision Power Stylesheet をデザインしたユーザーにより作成され、Authentic View ユーザーが定義することはできません。

動的なテーブル構造の編集
データ構造は**動的な SPS テーブル**の行として表現されます。それぞれの要素は行として表示されます。従って、各行には XML のサブ構造と同様の情報が表示されます。

動的なテーブルの編集コマンドにより、要素の出現順序やその数を変更といった、動的 SPS テーブルの操作を行うことがで

きます。しかしながら、各出現要素のサブ構造を変更することになるため、動的 SPS テーブルのカラム編集を行うことはできません。

動的テーブル編集コマンドのアイコンはツールバーに表示されるとともに、**Authentic** メニューからもアクセスすることができます。

テーブルへ行を追加

テーブルへ行を挿入

現在の行を複製（セルのコンテンツが複製されます）

現在の行を1つ上に移動

現在の行を1つ下に移動

現在の行を削除

注釈：これらのコマンドは**動的 SPS テーブル**に対してのみ適用され、静的 SPS テーブル内部で使用されることはありません。Authentic View で使用される様々な種類のテーブルに関しては、このドキュメンテーションの Authentic View におけるテーブル使用 にて記されます。

XML テーブルの作成と編集

お使いのデータをテーブルとして表示したい場合、独自のテーブルを挿入することができます。テーブルは XML テーブルとして挿入され、そのテーブルの構造やフォーマットを変更することができます。XML テーブルの作成や編集を行うためのコマンドへは、以下に示されるようにツールバーのアイコンからアクセスすることができます。これらコマンドの詳細に関しては、XML テーブル編集アイコンを参照ください。

これらのアイコンに対応するコマンドは、**メニューアイテムからは利用できません。**更に、XML テーブルを使用するには、StyleVision Power Stylesheet にてこの機能を有効にし、正しく設定する必要があります。

Authentic View にて使用することのできるテーブルの種類や、XML テーブルの作成ならびに編集方法に関する詳細については、Authentic View におけるテーブルの使用 参照ください。

テキストフォーマットアイコン

Authentic View におけるフォーマットは、フォーマット情報を持った XML 要素や属性をテキストに対して適用することで行われます。そのようなフォーマットが定義されると、StyleVision Power Stylesheet のデザイナーは Authentic View ツールバーのアイコンからフォーマットの適用を行うことができます。テキストフォーマットアイコンによりテキストフォーマットを適用するには、フォーマットしたいテキストをハイライトして、適切なアイコンをクリックします。

DB 行ナビゲーションアイコン

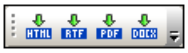
左から順に矢印アイコンは、DB 内の最初の行へ、前の行へ、レコードへ移動ダイアログを開く、次のレコードへ移動、最後のレコードへ移動、となります。

このアイコンにより、クエリの入力を行うデータベースクエリの編集ダイアログを開きます。Authentic View には問い合わせ結果の行が表示されます。

XML データベース編集
XML データ付きの新規行を選択して編集コマンドにより、IBM DB2 といったXML DB 内の関連するテーブルから、新たな行を選択することができます。 Authentic View に表示されるこの行は、編集後、DB 内に保存しなおすことができます。

Portable XML Form (PXF) ツールバーボタン

XMLSpy およびAuthentic Desktop のAuthentic View では以下の PXF ツールバーボタンを利用することができます：



個々のボタンをクリックすることで、HTML、RTF、PDF、DocX 出力が生成されます。.

これらのボタンは PXF ファイルが Authentic View にて開かれているときに有効になります。個々の出力フォーマットに対応した XSLT スタイルシートが PXF ファイルに含まれている場合、対応するボタンが有効になります。例えば、HTML ならびに RTF に対して対応した XSLT スタイルシートだけが PXF ファイルに含まれている場合、HTML ならびに RTF 出力に対応したツールバーボタンだけが有効になり、PDF ならびに DocX (Word 2007+) 出力に対応したボタンが無効になります。

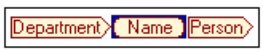
8.2.3 Authentic View メインウィンドウ

Authentic View には大きなマークアップ、小さなマークアップ、混合マークアップ、そしてマークアップを隠すという4つの異なる表示モードがあり、これらのモードを切り替えることで、異なるレベルのマークアップ情報を表示することができます。モード間の切り替えを行うには、**Authentic** メニューにあるコマンドを使用するか、ツールバーにあるアイコンをクリックしてください（前セクションのAuthentic View ツールバーアイコン）を参照ください）。

大きなマークアップ

このオプションでは、要素の開始ならびに終了タグが、要素名 /属性名を持った属性とともに表示されます。

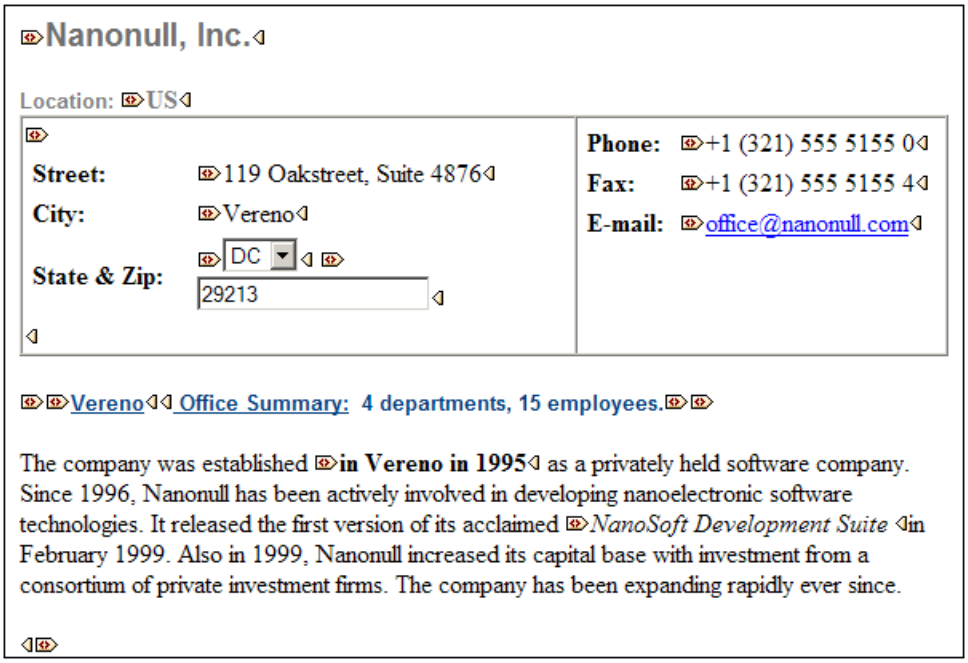
上のイメージにあるName という要素は**展開**されたもので、開始ならびに終了タグが、要素のコンテンツとともに表示されます。開始または終了タグをダブルクリックすることで、要素 /属性を**縮退**することができます。そのタグをダブルクリックすることで、縮退された要素 /属性を展開することができます。



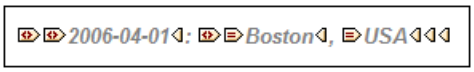
大きなマークアップにおいて、属性は開始ならびに終了タグの等号記号（=）により表されます。

小さなマークアップ

このオプションでは、要素 /属性の開始ならびに終了タグが名前なしで表示されます。



開始タグにはシンボルが表示されていますが、終了タグにはそれが無いことに注目してください。また、以下のスクリーンショットで示されるように、要素タグには不等号記号（<>）が表示されているとともに、属性タグには等号記号が付随します。



要素/属性を縮退または展開するには、対応するタグをダブルクリックします。以下の例では（青色でハイライトされている）縮退された要素が示されます。縮退された要素のタグと、その隣にある（展開されている）開始タグの形が違うことに注目してください。

混合マークアップ

混合マークアップでは、カスタマイズされた個々のマークアップを表示します。StyleVision Power Stylesheet をデザインする人は、ドキュメント内の各要素/属性に対して、大きなマークアップ、小さなマークアップ、またはマークアップを表示しないよう指定することができます。Authentic View によりドキュメントの閲覧をしているユーザーは、このモードにてマスタマイズされた個々のマークアップを表示することになります。

全てのマークアップを隠す

全てのXML マークアップが非表示になります。Authentic View にて表示されるフォーマットは、印刷時のドキュメントフォーマットになるため、このモードではWYSIWYG なドキュメントの表示が行われます。

コンテンツの表示

Authentic View において、コンテンツの表示は2種類の方法で行われます：

- プレインテキスト：テキストの入力を行うと、そのテキストが要素のコンテンツまたは属性の値になります。

- データ入力デバイス：入力フィールド（テキストボックス）や複数行の入力フィールド、コンボボックス、チェックボックス、ラジオボタンが該当します。入力フィールドや、複数行の入力フィールドによっては、その中に入力したテキストが、XML 要素のコンテンツまたは属性の値となります。

その他の入力デバイスの場合、行われた選択に応じて、StyleVision Power Stylesheet で指定されたXML の値が入力されます。例えばコンボボックスの場合、（コンボボックスのドロップダウンリストから選択可能な）「承認」という選択肢 "1"や "approved" というXML の値にマッピングしたり、「未承認」という選択肢を "0"や "hot approved" という値にマッピングすることができます。

オプションノード
要素や属性が（参照されるスキーマにより）オプションとして定義されている場合、**[要素 属性]を追加**が表示されます：

この表示をクリックすることで要素が追加され、カーソルがデータ入力のために配置されます。複数のオプションノードが存在する場合、**追加 ...**が表示され、クリックすることでオプションノードのメニューが表示されます。

8.2.4 Authentic View 入力ヘルパー

Authentic View には要素、属性、エンティティに対応した3つの入力ヘルパーがあり、ウィンドウとしてAuthentic View インターフェースの右側に表示されます（以下のスクリーンショットを参照）。

要素ならびに属性入力ヘルパーの表示内容は、ドキュメント内におけるカーソルの現在位置に応じて変化します。エンティティ入力ヘルパーに表示されるエンティティは、カーソルの位置に関わらず、ドキュメント内で許されている全てのエンティティが表示されます。

以下に各入力ヘルパーの説明を記します。

要素入力ヘルパー

要素入力ヘルパーは2つの部分から構成されています：

- 上部には XML ツリーが表示されており、**XML ツリーを表示**チェックボックスにより、その表示を切り替えることができます。XML ツリーには、現在の要素からドキュメントのルート要素までの祖先が表示されます。XML ツリーの要素をクリックすると、対応する要素が 次の記述で説明されるように）要素入力ヘルパーの下部に表示されます。
- 下部にはノードのリストが表示され、Authentic View にて選択された要素やテキスト領域の前や内部、後にノードを挿入、または削除したりといった操作を行うことができます。入力ヘルパー内に表示されている要素に対して行える操作は、要素名の隣にあるアイコンにより示されます。要素入力ヘルパーに表示されるアイコンとその説明を以下に記します。

入力ヘルパーに表示されているノードを使用するには、そのアイコンをクリックしてください。

要素の後に挿入
入力ヘルパー内の要素が、選択されている要素の後に挿入されます。ノードは適切な階層レベルにて追加されます。例えば、カーソルが/ / sect1/ para 要素内部にあり、sect1 要素を追加した場合、sect1 要

素は/ / sect1/ para の兄弟としてではなく、para 要素の親であるsect1 要素の後に来る兄弟として追加されます。.

要素の前に挿入
入力ヘルパー内の要素が、選択された要素の前に挿入されます。要素の後に挿入するのと同様に、ノードは適切な階層レベルに挿入されます。

要素を削除
要素とそのコンテンツを削除します。

要素を挿入
入力ヘルパーにある要素が、要素内に挿入されます。カーソルが要素の内部に位置している場合、その要素内にて許されている子要素を挿入することができます。挿入が許されている要素は要素のみのコンテンツや（テキストと子要素から成る）混合コンテンツモデルの一部でもあり得るということに注意してください。

テキスト領域が選択されている時か、カーソルがテキストの挿入ポイントに位置している時に許されている要素を挿入することができます。

- テキスト領域が選択されている状態で要素が挿入されると、選択されているテキストが挿入された要素のコンテンツになります。
- 挿入ポイントに要素が挿入された場合、そのポイントに要素が挿入されます。

要素が挿入された後、これらインライン要素に対して（要素入力ヘルパーに）表示される要素のクリアアイコンをクリックすることで、要素のクリアを行うことができます。2種類あるアイコンのどちらが表示されるかはテキスト領域を選択するか、カーソルをテキスト内に配置するかにより変化します（以下のスクリーンショットを参照）。

要素を適用
ドキュメント内の要素を（開始または終了タグをクリックすることで）選択すると、その要素を他の要素に置き換えることができます。例えば、para のような混合コンテンツ要素において、bold 要素の代わりにitalic 要素を置き換えることができます。入力ヘルパーにある要素を、選択された（オリジナルの）要素に対して適用できることをこのアイコンは示しています。**要素を選択**コマンドは要素内または混合コンテンツ内のテキストに対しても適応することができ、適用された要素のコンテンツとしてテキスト領域は作成されます。

- 適用される要素にオリジナルの要素にあるものと**同名の子要素**があり、この子要素のインスタンスがオリジナルの要素内に存在する場合、オリジナルの子要素が要素のコンテンツ内に保持されます。
- 適用される要素に、オリジナルの要素にてインスタンス化された子要素と**同名の子要素が存在しない**場合、オリジナルの要素にてインスタンス化された子は、新たな要素内にある子の兄弟として追加されます。
- 適用される要素に、**オリジナル要素のコンテンツモデルには存在しない子要素がある**場合、この子要素は直接作成されず、Authentic View により挿入オプションが表示されます。

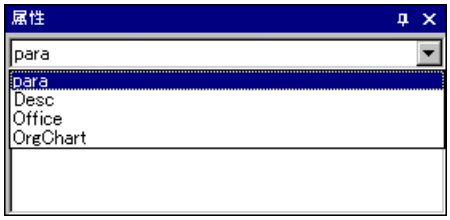
要素ではなく、テキスト領域が選択された状態で要素が適用されると、選択されたテキスト領域をコンテンツとして要素が適用されます。カーソルが挿入ポイントにある場合、要素の適用は許可されません。

領域選択時に）要素をクリア
混合コンテンツ要素内にあるテキストが選択されている時にこのアイコンが表示され、アイコンをクリックすることで選択されたテキスト領域の周りにある要素がクリアされます。

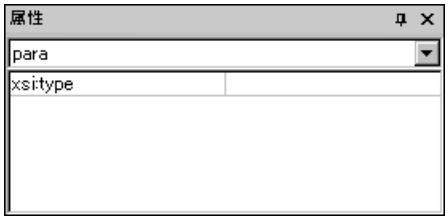
挿入ポイント選択時に）要素をクリア
混合コンテンツ要素の子要素内にカーソルが位置しているときにこのアイコンが表示されます。アイコンをクリックすることで、インライン要素がクリアされます。

属性入力ヘルパー
属性入力ヘルパーはコンボボックスと属性のリストから成り立ち、（開始または終了タグをクリック、または要素内のコンテンツをクリックすることで）選択した要素がコンボボックスに表示されます。

以下のスクリーンショットにて表示される属性入力ヘルパーでは、コンボボックス内にpara 要素が表示されます。コンボボックスの矢印をクリックすることで、para 要素から（この場合はOrgChart である）**ドキュメントのルート要素までの祖先**が表示されます。



コンボボックスの下には、その要素（この例ではpara）に対して妥当な属性のリストが表示されます。その要素に対して属性が必須の場合、その属性が太字で表示されます。以下の例では、ビルトインの属性であるxsi:type 以外に必須属性はありません。



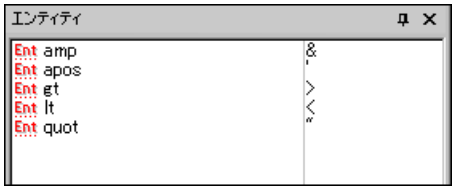
属性の値を入力するには、その属性の値フィールドをクリックし、値を入力します。この操作により、その属性が値とともにXML ドキュメント内に作成されます。

nillable な要素が選択された時に属性入力ヘルパーに表示されるxsi:nil 属性の場合、xsi:nil 属性の値はドロップダウンリストから許されている値（true または false）を選択することで入力されます。

xsi:type 属性は、属性の値フィールドをクリックして、表示されるドロップダウンリストから選択を行うことで変更することができます。表示される値はAuthentic 表示ドキュメントがベースとしているXML スキーマ内で定義されているabstract 型となります。

エンティティ入力ヘルパー

エンティティ入力ヘルパーを使うと、ドキュメント内にエンティティを挿入することができます。エンティティは（会社名のように）ドキュメント内で頻繁に表示されるテキストや特別な文字を挿入するために使用されます。エンティティを挿入するには、テキスト内のエンティティを挿入したい箇所にカーソルを移動させ、エンティティ入力ヘルパー内のエンティティをダブルクリックします。



メモ: DTD 内で定義されているエンティティを内部エンティティと言い、外部 XML ファイルのような外部ソースに値が含まれているエンティティを外部エンティティと言います。内部ならびに外部エンティティの両方がエンティティ入力ヘルパーには表示されます。エンティティを挿入すると、それが外部エンティティであれ内部エンティティであれ、（エンティティの値ではなく）エンティティが XML テキストに挿入されます。エンティティが内部エンティティの場合、Authentic View には**そのエンティティの値**が表示されます。エンティティが外部エンティティの場合、（値ではなく）エンティティが表示されます。従って、例えば外部エンティティである XML ファイルがエンティティとして Authentic View に表示され、そのコンテンツは Authentic View にて表示されません。

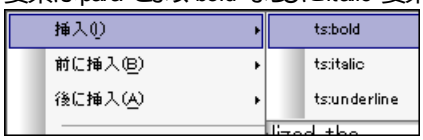
Authentic View にて**独自のエンティティを定義**して、入力ヘルパーにて表示させることも可能です。Authentic View における編集セクション内にあるエンティティの定義を参照ください。

8.2.5 Authentic View コンテキストメニュー

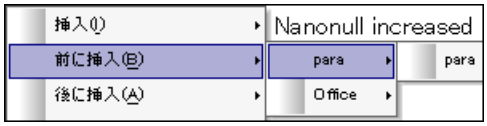
選択されたドキュメントのコンテンツやノードを右クリックすることで、その選択やカーソル位置に関連するコマンドがコンテキストメニューにより表示されます。

要素の挿入

以下のスクリーンショットには**挿入**サブメニューが示されており、現在のカーソル位置において挿入することのできる全ての要素が表示されます。**前に挿入**サブメニューでは、現在の要素の前に挿入することのできる全ての要素が表示されます。**後に挿入**サブメニューでは、現在の要素の後に挿入することのできる全ての要素が表示されます。以下のイメージにおいて、現在の要素は para となり、bold ならびに italic 要素を内部に挿入することができます。



以下に示されるように、現在カーソルがある para 要素の前には para ならびに Office 要素を挿入することができます。



コンテキストメニューにて利用可能な挿入、置換え（**適用**）、マークアップの削除（**クリア**）といったコマンドは Authentic View 入力ヘルパー でも利用可能で、詳細な情報は入力ヘルパーのセクションに記述されます。

エンティティの挿入

エンティティを挿入コマンドにカーソルを移動させることで、宣言されているエンティティを表示するサブメニューが開かれます。エンティティをクリックすることで、その選択が挿入されます。ドキュメントに対してエンティティを宣言する方法に関しては、エンティティの定義 を参照ください。

CDATA セクションの挿入

テキスト内にカーソルが位置している時に、このコマンドが有効になります。クリックすることで CDATA セクションがカーソルの挿入ポイントに挿入されます。CDATA セクションは開始ならびに終了タグにより分離されており、これらのタグを表示するには大きなマークアップまたは小さなマークアップモードに切り替える必要があります。CDATA セクション内部では XML のマークアップや構文が無視されます。XML マークアップ文字（アンパサンド、アポストロフィー、不等号、引用句）はマークアップとしてではなく、文字列（テラル）として扱われます。CDATA セクションは、XML マークアップ文字を含むプログラムコードの様なテキストを挿入する際に便利な機能となります。

ノードの削除

マウスカーソルを**削除**に移動させると、選択されたノードと（ドキュメントを妥当な状態に保ったまま）削除可能な要素の祖先から成るメニューが表示されます。クリックすることで、その要素が削除されます。この機能により要素や、その祖先などを素早く削除することができます。祖先要素をクリックすることで、その子孫（選択中の要素を含む）も全て削除されることに注意してください。

クリア

クリアコマンドにより、選択箇所の周りにあるマークアップが削除されます。全ノードが選択されている場合、そのノード全体の要素マークアップが削除されます。テキストのセグメントが選択されている場合、そのテキストセグメントの周りからのみ要素マークアップが削除されます。

適用

適用コマンドでは、選択された要素をメインウィンドウ内の選択箇所に適用します。詳細については Authentic View 入力ヘルパーを参照ください。

コピー、切り取り、貼付け

これらは標準的な Windows コマンドです。しかし、**貼付け**コマンドにおいてコピーされたテキストは、スタイルシートのデザイナーが $PS 全体に対して指定した内容に従い XML またはテキストとしてコピーされることに注意してください。**XML としてコピー**ならびに**テキストとしてコピー**コマンドに関する詳細は、以下の**貼付けの形式**を参照ください。

貼付けの形式

貼付けの形式コマンドでは、(クリップボードにコピーされた) Authentic View XML フラグメントを XML またはテキストとしてコピーします。コピーされたフラグメントが XML として貼り付けられる場合、XML マークアップとともに貼付けが行われます。テキストとして貼り付けられる場合、コピーされたフラグメントのテキストコンテンツのみが貼り付けられ、XML マークアップが存在していたとしても無視されます。以下のケースが考えられます：

- Authentic View において**全てのノードをマークアップタブとともにハイライト**し、クリップボードにコピーします。(i) ドキュメントを妥当に保ったまま挿入できる位置に、ノードを XML として貼り付けることができます。妥当性を失う位置に貼り付けられることはありません。(ii) ノードがテキストとして貼り付けられる場合、マークアップ以外のテキストコンテンツだけが貼り付けられます。テキストコンテンツは、テキストを貼り付けることのできる箇所なら、ドキュメント内にある任意の場所に貼り付けることができます。
- **テキストフラグメント**を Authentic View にてハイライトし、クリップボードにコピーします。(i) このフラグメントが XML として貼り付けられると、このテキストの XML マークアップタグが (テキストフラグメントとともに明示的にコピーされていなかったとしても) テキストとともに貼り付けられます (ただし、フラグメントが貼り付けられる箇所において XML ノードが妥当となる場合だけ)。(ii) フラグメントをテキストとして貼りつける場合、テキストとして貼り付けることのできる XML ドキュメント内の任意の場所に貼り付けることができます。

メモ: テキストは、許されている箇所にだけ貼り付けることができるため、コピーされたテキストがドキュメントの妥当性を損なわないか確認する必要があります。コピーされたテキストは：

(i) 新しい位置において文字の種類として妥当であり (例えば、数字だけが認められているノードに対して、それ以外の文字を使用しない)
(ii) その他にもノードの妥当性を保つ (例えば、3文字までの使用が許されているノードで4文字以上使用しない) 必要があります。

貼り付けられたテキストが、何らかの形でドキュメントの妥当性を損なう場合、テキストは赤色で示されます。

削除

削除コマンドは選択されたノードとそのコンテンツを削除します。カーソルを特定のノード内に位置させるか、開始または終了タグをクリックすることで、ここでは選択したと見なされます。

8.3 Authentic View における編集

このセクションでは Authentic View の重要な機能の詳細を解説します。このセクションで紹介される機能は頻繁に使用するものか、機能のメカニズムやコンセプトが説明を要するものです。

このセクションでは以下のことが解説されます：

- Authentic View では 3つの異なる種類のテーブルが使用されます。Authentic View においてテーブルを使用 セクションでは、これら3種類のテーブル（静的 SPS、動的 SPS、そして XML）と、それらをいつどのように使用するのか説明します。まず大まかな範囲をコンセプトとともに説明し、詳細な使用方法を記述します。
- 日付の選択はクリックされた日付を正しい XML フォーマットで入力する、グラフィカルなカレンダーです。詳細については 日付の選択を使用を参照ください。
- エンティティは特殊な文字やテキスト文字列を簡単にまとめたものです。エンティティを定義し、対応するエンティティを挿入することで、これら特殊な文字やテキスト文字列を挿入することができるようになります。詳細は エンティティの定義 を参照下さい。
- Enterprise または Professional エディションの Altova 製品において、Authentic View ユーザーは XML ドキュメントを、デジタル XML 署名 により署名したり、その検証を行うことができます。
- Authentic View にて イメージフォーマットを表示。

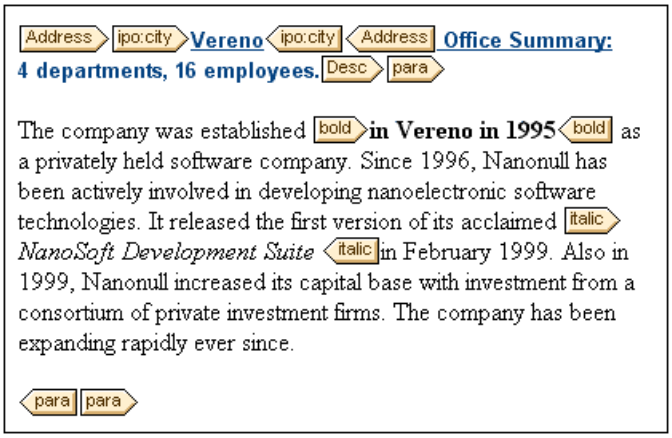
8.3.1 基本的な編集

Authentic View における編集作業は、そのまま XML ドキュメントの編集を意味します。Authentic View を使用することで、ドキュメントにおける XML マークアップを隠し、ドキュメントのコンテンツのみを表示することができます（最初のスクリーンショットを参照）。この機能により XML の技術的な側面に触れること無く、通常のテキストドキュメントとして XML ドキュメントの編集を行うことができます。マークアップは必要に応じて表示させることもできます（2番目のスクリーンショットを参照）。

Vereno Office Summary: 4 departments, 16 employees.

The company was established **in Vereno in 1995** as a privately held software company. Since 1996, Nanonull has been actively involved in developing nanoelectronic software technologies. It released the first version of its acclaimed *NanoSoft Development Suite* in February 1999. Also in 1999, Nanonull increased its capital base with investment from a consortium of private investment firms. The company has been expanding rapidly ever since.

Authentic View にて表示された XML マークアップ非表示状態のドキュメント

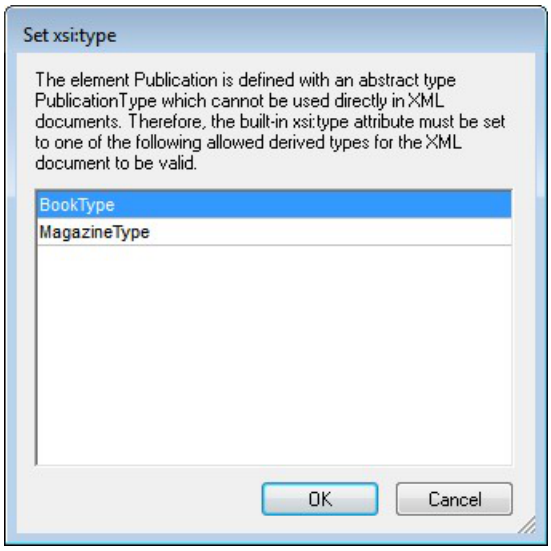


XML マークアップが Authentic View にて表示された状態

ノードの挿入

ユーザーはしばしば Authentic XML ドキュメントに新たなノードを追加する必要に迫られます。例えば、アドレス帳の役割を果たすドキュメントに、新しい Person 要素を追加しなければならなくなるといった具合です。このような場合、XML スキーマにより新たな要素の追加が許されていなければなりません。Authentic ビューにて表示されているドキュメントにおいて新たなノードを挿入する直前（または直後）のノードを右クリックし、コンテキストメニューから **前に挿入**」または **後に挿入**」を選択します。ドキュメント内でクリックされた位置に挿入可能なノードのリストがサブメニューに表示されます。目的のノードをクリックして挿入を行って下さい。必須となっている子孫ノードも同時に挿入されます。子孫ノードがオプションとなっている場合、ノード名を追加 というオプションが表示され、クリックすることでそのノードを追加することができます。

追加されたノードが abstract 型の要素である場合、（以下のスクリーンショットにあるような）ダイアログが、XML スキーマで利用可能な派生型とともに表示されます。



Publication 要素が追加されたところで上のスクリーンショットに示されるダイアログが表示されます。Publication 要素が abstract な複合型の PublicationType 型により定義されています。abstract 型の PublicationType からは BookType と MagazineType という 2 種類の複合型が派生しており、Publication 要素が XML ドキュメントへ追加されると、Publication の abstract 型から派生したこれら 2 つの型のうちいずれかを指定する必要があります。新たな Publication 要素には xsi:type 属性が追加されます：

```
<Publication xsi:type="BookType"> ... </Publication>
<Publication xsi:type="MagazineType"> ... </Publication>
```

```
...
<Publication xsi:type="MagazineType"> ... </Publication>
```

表示されている派生型の1つを選択して**OK** をクリックすると以下のことが行われます：

- 選択された派生型を、要素の xsi:type 属性の値としてセットします。
- コンテンツモデルで定義されている選択された派生型の子孫ノードとともに要素が挿入されます。

選択された派生型は、要素のxsi:type 属性を属性入力ヘルパーで編集することにより変更できます。この方法で要素の型が変更されると、以前の型のコンテンツモデルにあるノードが全て削除され、新たな型のコンテンツモデルにあるノードが挿入されます。

テキストの編集

Authentic View ドキュメントはテキストとイメージから成り立っています。ドキュメント内のテキストを編集するには、テキストを挿入する場所にカーソルを移動させ入力を行います。テキストは、キーボードやマウスのドラッグアンドドロップといった通常使用される方法でコピー、移動、削除を行うことができます。唯一の例外は**Enter** キーです。Authentic View ドキュメントは既にフォーマットされているため、アイテムの間に余分な余白を加えることはできません。Authentic View において**Enter** キーは、現在編集されている要素の新たなインスタンスを作成するためだけに使用されます。

XML またはテキストとしてコピー

テキストはそのままテキストとして、またはXML としてコピーすることができます。

- テキストを XML として貼りつけると、ノードのテキストコンテンツとともに XML マークアップも貼り付けられます。ノードコンテンツの一部しかコピーされない様な場合でも XML マークアップは貼り付けられます。マークアップを貼り付けるには、貼り付けを行う場所において、そのマークアップがスキーマにより許されていなければなりません。
- テキストをテキストとして貼りつけると、XML マークアップは無視されます。

XML またはテキストとして貼り付けるには、テキストをコピー（**Ctrl+C**）し、貼付けを行う場所を右クリックし、コンテキストメニューから**貼付けの形式 | XML として貼付け**」または**貼付けの形式 | テキストとして貼付け**」を選択します。貼付けのショートカット**Ctrl+V** が使用された場合、SPS のデフォルト貼り付けモードに従い貼付けが行われます。詳細については、コンテキストメニューセクションを参照ください。

他にも、ハイライトしたテキストをドラッグして貼り付けることができます。テキストがドロップされると、テキストをテキストとして貼り付けるのか、XML として貼り付けるのかの決定を促すポップアップが表示されます。必要なオプションを選択してください。

テキストのフォーマット

XML ドキュメントシステムの原理は、コンテンツとその表現形式（書式）を分離するというものです。XML ドキュメントにはコンテンツが蓄えられ、スタイルシートには表現形式（フォーマット）が保存されます。Authentic View では、スタイルシートにより XML が表示されており、これはAuthentic View にて表示される全てのフォーマットはスタイルシートにより生成されることを意味します。ドキュメント内の太字テキストは、スタイルシートにより太字フォーマットが与えられており、リストやテーブルがドキュメント内にある場合、そのリストやテーブルのフォーマットもスタイルシートにて与えられてます。Authentic View にて編集するXML ドキュメントにはコンテンツしか含まれておらず、フォーマットに関する情報は全てスタイルシートに収められています。これはAuthentic View ユーザーは編集するテキストのフォーマットを変更することができない（する必要も無い）ことを意味しています。コンテンツに対して自動的に適用されるフォーマットは、編集中のデータが持つ意味またはドキュメント構造にリンクしています。例えば、（意味を持ったユニットであると見なされる）E メールアドレスは、自動的に特定の形式にフォーマットされます。同様に、見出しも（意味的ならびにドキュメントの構造という観点から）ドキュメントの特定位置に表示され、スタイルシートデザイナーが見出しに対して指定したフォーマットに従い表示されます。E メールアドレスや見出しのフォーマットを変更することはできません。行えるのはE メールアドレスや見出しのコンテンツを変更することだけです。

例えば、テキストを太字で表示するなど、特別な方法でコンテンツの表示を行う必要が出てくる場合もあります。そのような場合、ドキュメント内に記述される要素に実際の表示を括りつけることで対応します。太字で表示するテキストを例にすると、目的のテキストを他のコンテンツから分離し、スタイルシートデザイナーが太字とデザインしたマークアップで囲む必要があります。Authentic View ユーザーが太字テキストを必要とする場合、テキスト文字列を適切な要素マークアップで囲まなけれ

ばなりません。そのような操作の詳細については、このドキュメンテーションの 要素入力ヘルパーセクションにある要素の挿入コマンドを参照ください。

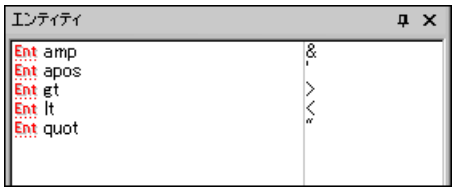
Authentic View にてリッチエディットを使用する

Authentic View にて、リッチエディットコンポーネントとして作成された要素内にカーソルが配置されている場合、リッチエディットツールバーのボタンが有効になります (以下のスクリーンショットを参照)。そうでない場合、これらのボタンはグレーアウトされます。

スタイルを指定するテキストを選択し、リッチエディットツールバーのボタンやコントロールにより、目的のスタイルを指定します。リッチエディットにより、テキストにおける文字の太さ、フォントスタイル、装飾、サイズ、色、背景色や配置をAuthentic View ユーザーが指定できるようになります。スタイルが指定されたテキストはスタイル要素により囲まれます。

エンティティの挿入

XML ドキュメントにおいて、一部の文字はマークアップのために使用され、通常のテキストとして使用することができなくなっています。つまりアンパサンド (&)、アポストロフィー (')、小なり (<)、大なり (>)、引用符 (")文字が使用できないテキストとなります。データ内でこれらの文字を使用するには、 エンティティ入力ヘルパー からエンティティ参照を挿入する必要があります (以下のスクリーンショットを参照)。



XML では独自のエンティティを作成することも可能です。エンティティでは (i) キーボードでは入力することができない特殊な文字、(i) ドキュメントコンテンツにて再利用したいテキスト、(ii) XML データフラグメント、または (v) イメージのようなその他のリソースを使用することができます。Authentic View アプリケーション内部で 独自のエンティティを定義 することもできます。一度定義すると、これらのエンティティが エンティティ入力ヘルパー に表示され、ドキュメント内に挿入することができるようになります。

CDATA セクションの挿入

CDATA セクションは XML プロセッサーが XML データとして認識しない XML ドキュメント内にあるセクションの事です。この機能はエンティティを使用するのが現実的ではないような長いテキストをエスケープする際に使用され、例えばプログラムコードやマークアップタグが混入している XML フラグメントを挿入するために使用することができます。CDATA セクションは <![CDATA[と]]> をそれぞれ開始ならびに終了の区切りとして、要素内に定義されます。また、]]> というテキストは CDATA セクションの終了と見なされてしまうため、CDATA 内部で使用することはできません。CDATA セクション内部で]]> を表現するには、最後の> をエンティティ参照 (>) によりエスケープします。CDATA セクションを要素内部に挿入するには、目的の場所にカーソルを移動し、右クリックして表示されるコンテキストメニューから**CDATA セクションを挿入**を選択します。Authentic View にて CDATA セクションのタグを閲覧するには、 マークアップを表示します。他にも、CDATA セクションにより囲まれるべきテキストをハイライトした後、**CDATA セクションを挿入**コマンドを選択することもできます。

メモ: CDATA セクションを入力フィールド (テキストボックスならびに複数行のテキストボックス) に挿入することはできません。CDATA セクションは、テキストコンテンツコンポーネントとして Authentic View に表示されている要素内部に対してのみ入力することができます。

リンクの編集とリンク先への移動

ハイパーリンクはリンクのテキストと、そのリンクのターゲットという2つのパーツから構成されます。リンクのテキストは、テキストをクリックすることで編集することができますが、リンクのターゲットを編集することはできません (リンクのターゲットは 静的なターゲッ

トアドレスか、XML ドキュメントから得られたデータからアドレスを生成することで（スタイルシートのデザイナーによりセットされます）。Authentic View では、**Ctrl** キーを押しながらリンクテキストをクリックすることで、リンク先へ移動することができます（単にリンクテキストをクリックするだけでは、テキストの編集しか行えないことを思い出して下さい）。

8.3.2 Authentic View におけるテーブル

3つあるテーブルの種類は、SPS テーブル（静的ならびに動的）とCALS/HTML テーブルという2つのカテゴリーに分類されます。

SPS テーブル：静的と動的という2つの種類から構成されます。SPS テーブルは、XML ドキュメントがリンクしている StyleVision Power Stylesheet のデザイナーにより定義されます。ユーザーは SPS テーブルを XML ドキュメントに挿入することはできませんが、SPS テーブルフィールドにデータを入力し、動的 SPS テーブルの行を追加、削除することはできます。SPS テーブルのセクションではこれらテーブルの機能を紹介します。

CALS/HTML テーブル：Authentic View のユーザーにより挿入されます。ドキュメント内の許可された箇所ならどこにでもテーブルを挿入することができます。CALS/HTML テーブルの編集機能ならびにCALS/HTML テーブル編集アイコンについて解説します。

SPS テーブル

Authentic View では静的テーブルと動的テーブルという2つのSPS テーブルが使用されます。

静的テーブル：セルの構造とコンテンツ型は固定されています。Authentic View のユーザーはテーブルセルにデータを入力することはできますが、これらテーブルの構造を変更（つまり行やカラムを追加）したり、セルのコンテンツ型を変更することはできません。テキストを入力することで、またはチェックボックスやラジオボタンなどから既に定義されたオプションから値を選択することでデータの入力を行います。データの入力後、そのデータを編集することはできます。

Nanonull, Inc.

Street:	119 Oakstreet, Suite 4876	**Phone:**	+1 (321) 555 5155
City:	Vereno	**Fax:**	+1 (321) 555 5155 - 9
State & Zip:	DC 29213	**E-mail:**	office@nanonull.com

メモ：動的テーブルを編集するためのコマンドを静的テーブルの編集に**使用してはいけません。**

動的テーブル：各行に同一のデータ構造が与えられているようなテーブルです（静的テーブルではそうなりません）。そのため行の追加や挿入、移動、削除といった行に対するオペレーションを行うことができます。これらのコマンドは**Authentic** メニューや、以下に示されるアイコンにて利用可能です。

これらのコマンドを使用するには、カーソルを適切な行に移動し、必要なコマンドを選択してください。

Administration (3)

First	Last	Title	Ext	EMail	Shares	Leave		
						Total	Used	Left
Vernon	**Callaby**	Office Manager	582	v.callaby@nanonull.com	1500	25	4	21
Frank	Further	Accounts Receivable	471	f.further@nanonull.com	0	22	2	20
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com	Sharesを追加	25	7	18
Employees: 3 (20% of Office, 8% of Company)					Shares: 1500 (12% of Office, 4% of Company)			
Non-Shareholders: Frank Further, Loby Matise.								

テーブル内にあるセル間を移動するには、矢印キーを使用してください。あるセルから次のセルに移動するには、**Tab** キーを使用してください。行の最後にあるセルで**Tab** キーを押下することで新たな行が作成されます。

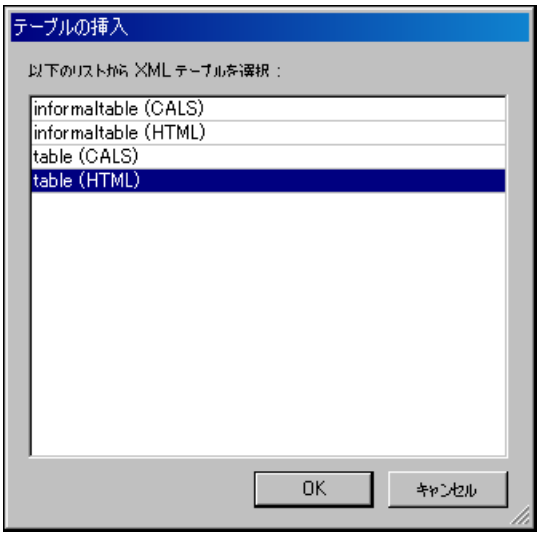
CALS/ HTML テーブル

Authentic View ユーザーは、テーブルフォーマットを表示すると指定されたXML データ構造に対してCALS/HTML テーブルを挿入することができます。CALS/HTML テーブルの挿入には、テーブルの挿入、形式の決定、そしてデータの入力という3つのステップが必要になります。CALS/HTML テーブルに関する処理を行うためのコマンドはツールバーに用意されています（CALS/ HTML テーブル編集アイコンを参照ください）。

テーブルの挿入

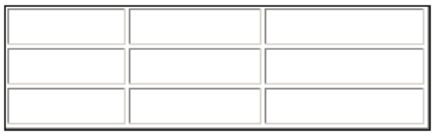
CALS/HTML テーブルを挿入するには、以下の操作を行います：

1. テーブルを挿入したい場所にカーソルを移動し、 アイコンをクリックします（テーブルを挿入することのできる箇所はスキーマにより指定されることに注意してください）。テーブルの挿入ダイアログが表示されます（以下のスクリーンショットを参照）。テーブル構造が定義された全てのXML 要素データ構造がこのダイアログには表示されます。例えば、以下のスクリーンショットでは、informaltable とtable 要素がCALS ならびにHTML テーブルの両方に対して定義されています。



2. エントリーを含む要素とテーブルモデルを選択し、**OK** をクリックします。
3. 次のダイアログ（以下のスクリーンショット）では、カラムと行の数、ヘッダーやフッターをテーブルに追加するか、利用可能な幅全てを使用するかなどを指定します。指定後に**OK** をクリックしてください。

上のダイアログボックスで指定した内容により、以下のようなテーブルが作成されます。



テーブル メニューコマンドを使用することで、カラムの追加、削除を行うことができ、行やカラムの結合や、分割を行うことができます。
まずは大まかな構造を作成しましょう。

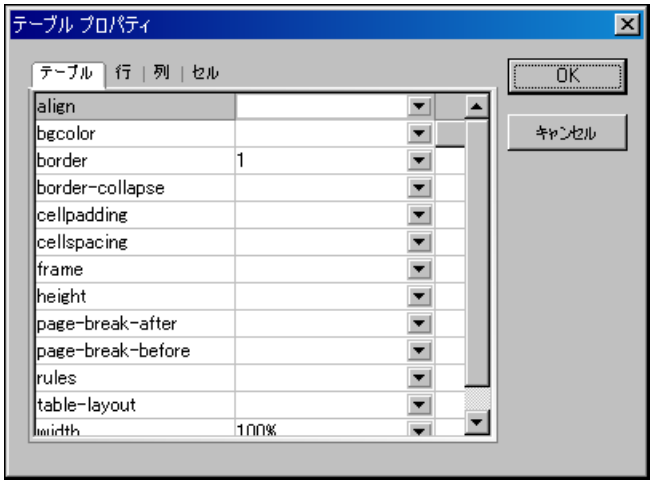
テーブルの書式とデータの入力

テーブルの書式は既にドキュメントデザインにて割り当てられていますが、以下のような条件が整っている場合によってはテーブルのフォーマットを修正できることもあります：

- 様々なテーブル構造要素に対応する要素において、CALS または HTML テーブルプロパティに対応するプロパティが（XML スキーマにて）属性として定義されている。ユーザーはこれら定義された属性をフォーマットに利用することができます。デザイン内にてこれらの属性に対して値が割り振られている場合、Authentic Viewにてこれらの値を上書きすることができます。
- デザイン内にて CSS スタイルを持つ style 属性がセットされていない。CSS スタイルを含む style 属性が要素に指定されていると、style 属性の値が全てのフォーマット属性に優先することになり、Authentic View にて指定されたフォーマットが全て上書きされてしまいます。

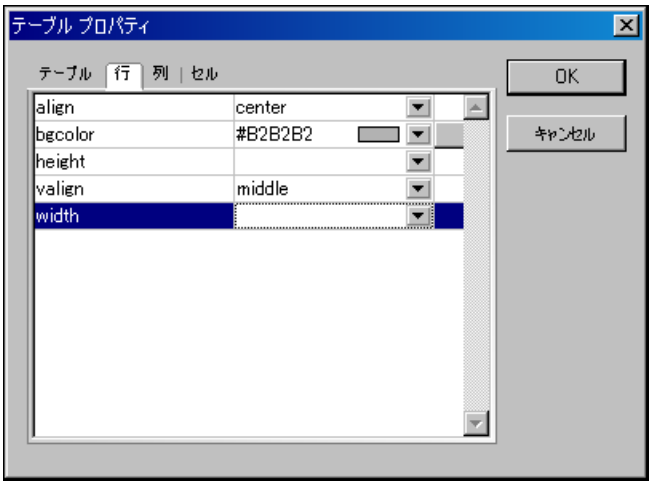
テーブル、行、カラム、またはセルのフォーマットを行うには、以下の操作を行います：

1. カーソルをテーブルに移動させ、テーブルプロパティアイコン をクリックします。この操作によりテーブルプロパティダイアログが表示され、テーブル、行、カラム、もしくはセルの書式を指定することができます（以下のスクリーンショットを参照）。



2. cellpadding とcellspacing の値を "0" に指定します。テーブルの外観は以下のようになります：

3. カーソルを書式設定する最初の行に移動し、テーブルプロパティアイコン をクリックし、**行**タブを選択します。



最初の行はヘッダーとなるため、背景色を異なるのもに変更して、他の行との違いを際立たせましょう。上のスクリーンショットで指定されている内容に注目してください。カラムヘッダーのテキストを入力すると、テーブルが以下のように表示されます：

Name	Telephone	Email

文字の配置が指定したとおり、中央寄せされていることに注目してください。

4. さて、ここで "Telephone" カラムを "Office" と "Home" というサブカラムに分割してみましょう。"Telephone" カラムを水平方向で2つのカラムに分割する必要があります。まずは最初にヘッダーセルからサブヘッダーの行を作成するため、カーソルを "Telephone" セルまで移動して、垂直に分割アイコン をクリックします。テーブルが以下のように表示されます：

Name	Telephone	Email

5. カーソルを "Telephone" セルのすぐ下にあるセルに移動させ、水平に分割アイコン をクリックします。その後 "Office" ならびに "Home" というテキストをそれぞれのセルに入力します。テーブルが以下のように表示されます：

Name	Telephone		Email
	Office	Home	

ここから、"Telephone" カラムにある各セルを水平に分割していくことになります。

テーブル編集アイコンを使用することで、カラムや行の追加や削除、セルのコンテンツを縦方向に揃えることができます。CALS/HTML テーブル編集アイコンについては、CALS/HTML テーブル編集アイコンにて記述されます。

テーブル内にあるセル間の移動

CALS/HTML テーブル内部のセル間を移動するには、矢印キーを使用します。

セル内にデータを入力

セル内にデータを入力するには、セル内にカーソルを移動させ、データを入力します。

テキストの書式

CALS/HTML テーブルにあるテキストは、XML ドキュメント内のテキストと同様に、XML 要素や属性により書式を指定します。要素を追加するには、テキストをハイライトして、要素入力ヘルパーに表示されている目的の要素をダブルクリックします。属性の値を指定するには、テキストフラグメント内にカーソルを移動し、必要な属性の値を属性入力ヘルパーから入力します。ヘッダーテキストを太字に指定すると、テーブルが以下のように表示されます。

Name	Telephone		Email
	Office	Home	

上に表示されるテキストは、テキストをハイライトし、太字の指定を行うグローバルテンプレートに存在するstrong という要素をダブルクリックします。テキストの書式が直ちに反映されます。

メモ Authentic Viewにて表示されているテキストに対して書式を指定するには、必要なテキスト書式が記述されたグローバルテンプレートをStyleVision にて作成する必要があります。

CALS/ HTML テーブル編集アイコン

CALS/HTML テーブルを編集するために必要なコマンドは、以下に記すツールバーのアイコンからアクセスすることができます。これらのアイコンに対応するコマンドはメニューコマンドには存在しないことに注意してください。

CALS/HTML テーブルがいつどのように使用されるかの詳細については、CALS/HTML テーブルを参照ください。

テーブルの挿入

「テーブルの挿入」コマンドにより、現在カーソルがある位置に**CALS/HTML テーブル**が挿入されます。

テーブルの削除

「テーブルの削除」コマンドにより、現在アクティブなテーブルを削除します。

行の追加

テーブルの削除」コマンドにより、現在アクティブなテーブルを削除します。

行の追加

「行の追加」コマンドにより、現在アクティブなテーブルの最後に新たな行を追加します。

カラムの追加

「カラムの追加」コマンドにより、現在アクティブなテーブルの最後に新たなカラムを追加します。

行の挿入

「カラムの挿入」コマンドにより、現在アクティブなテーブル内にあるカーソルの左に新たなカラムを挿入します。

左のセルと結合

「左のセルと結合」コマンドにより、現在のセル（現在カーソルが位置しているセル）を左のセルに結合します。両方のセルにあるタグは新たなセルに引き継がれ、カラムのヘッダーは変更されず、連結されます。

右のセルと結合

「右のセルと結合」コマンドにより、現在のセル（現在カーソルが位置しているセル）を右のセルに結合します。両方のセルにあるコンテンツは新たなセルにて連結されます。

下のセルと結合

「下のセルと結合」コマンドにより、現在のセル（現在カーソルが位置しているセル）を下のセルに結合します。両方のセルにあるコンテンツは新たなセルにて連結されます。

上のセルと結合

「上のセルと結合」コマンドにより、現在のセル（現在カーソルが位置しているセル）を上のセルに結合します。両方のセルにあるコンテンツは新たなセルにて連結されます。

セルを水平方向に分割

「セルを水平方向に分割」コマンドにより、現在アクティブなセルの右に新たなセルを作成します。セルのサイズは元のセルのサイズと同じです。

セルを垂直方向に分割

「セルを垂直方向に分割」コマンドにより、現在アクティブなセルの下に新たなセルを作成します。

上揃え

このコマンドにより、セルのコンテンツを上揃えにします。

中央揃え

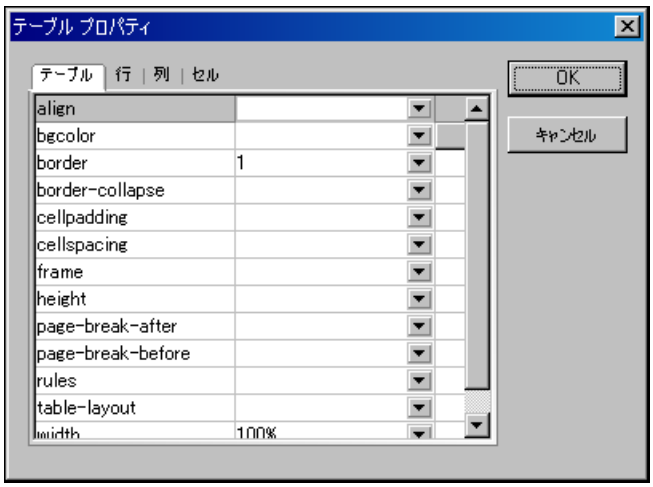
このコマンドにより、セルのコンテンツを中央揃えにします。

下揃え

このコマンドにより、セルのコンテンツを下揃えにします。

テーブルプロパティ

「テーブルプロパティ」コマンドにより、テーブルプロパティダイアログボックスが表示されます。このアイコンはHTML テーブルに対してのみ有効になり、CALS テーブルに対して使用することはできません。



8.3.3 DB の編集

Authentic Viewではデータベース (DB) のテーブルを編集し、DB 内にデータを保存することができます。このセクションではDB テーブルを編集する際に使用する事のできるインターフェースの記述を行います。以下の点を覚えておいてください：

- Authentic Viewに表示される DB テーブルのレコード数は、StyleVision Power Stylesheet のデザイナーにより、(デザインをよりコンパクトにまとめる為) あえて制限されていることがあります。このような場合、制限された数のレコードしか開始時には Authentic Viewにロードされません。DB テーブル行ナビゲーションアイコン (DB テーブルのナビゲーションを参照) を使用することで、DB テーブルにある他のレコードも表示することができます。
- DB の問い合わせ を行い、特定の行を表示することができます。
- DB の行を追加、修正、削除して、それらの変更をDB に保存することができます。DB テーブルの修正を参照ください。

DB をベースにしたStyleVision Power Stylesheet をAuthentic Viewで開くには：

- 「**Authentic | データベースデータの編集**」をクリックし、目的のStyleVision Power Stylesheet を選択します。

DB テーブルのナビゲーション

DB テーブル行のナビゲーションを行うコマンドはAuthentic View ドキュメント内で使用できるボタンとして使用できます。通常、4ないし5個のボタンにより構成されるナビゲーションパネルが各 DB のテーブルに対応します。

左から順に、DB テーブルの最初の行へ、1つ前の行へ、レコードへ移動ダイアログ (以下のスクリーンショット) を開く、次の行へ、最後の行へ移動します。

目的のボタンを押下することでDB テーブルのナビゲーションを行います。

XML データベース

IBM DB2 といったXML DB では、1つのセル (またはレコード) にXML ドキュメントを含めることができ、そのレコードを Authentic View にロードすることができます。他の行にあるXML ドキュメントをロードするには、「**Authentic | XML データの新たな行を選択し編集**」メニューコマンドを使用します。

DB クエリ

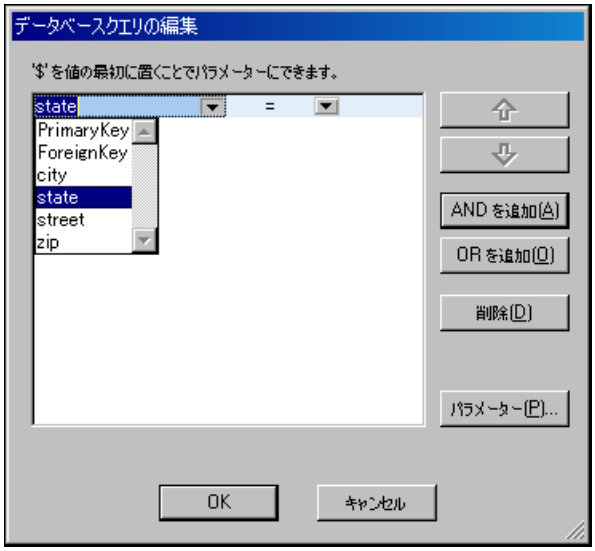
DB クエリを使用することで、Authentic View に表示されているテーブル行の問い合わせをすることができます。クエリは各テーブルに対して作成され、各テーブルに対して複数のクエリを作成することはできません。クエリの作成は編集中いつでも行うことができます。クエリの送信時にAuthentic View ドキュメントにて保存されていない変更がある場合、ドキュメント内にある**全ての**変更点を保存するか、または**全ての**変更点を破棄するか選択するよう促されます。他のテーブルにて行った変更も破棄されてしまうことに注意してください。クエリを送信すると、クエリ条件を使ってテーブルが再ロードされます。

メモ: 開いているテーブルの数が多すぎる旨のメッセージが表示された場合、テーブルをフィルタリングするクエリを使用することで、開かれているテーブルの数を減らすことができます。

クエリを作成し送信するには：

1. 目的のテーブルにあるクエリボタン をクリックし、データベースクエリの編集ダイアログ (以下のスクリーンショットを参照) を開きます。このボタンは各テーブルのトップまたは下部に表示されます。どのテーブルに対してもクエリボタンが表示されていない場合、StyleVision Power Stylesheet のデザイナーは、そのテーブルに対してデータベースクエリ機能を有効にしていないということになります。

2. **AND を追加**または**OR を追加**ボタンをクリックします。この操作によりクエリに対する空の条件が追加されます (以下のスクリーンショットを参照)。



4. 条件に使用する条件式を入力します。条件式は、(i) フィールド名（コンボボックスから選択できます）、(ii) 演算子（コンボボックスから選択できます）、そして (iii) 値（直接入力します）から構成されます。条件式を生成する詳細については、条件式内の表現を参照ください。
5. 他の条件を追加したい場合には、2つの条件を結びつける論理演算子（AND または OR）により、**AND を追加**または**OR を追加**ボタンをクリックし、その後条件を追加します。論理演算子に関する詳しい情報はDB クエリの条件を並び替えるを参照ください。

条件式内の表現

DB クエリ条件はフィールド名、演算子、そして値から構成されます。**使用することのできるフィールド名**は選択されたトップレベルデータテーブルの子要素で、これらテーブルの名前はコンボボックスに表示されます（上のスクリーンショットを参照）。以下に使用することのできる**演算子**を記します：

=	等しい
<>	等しくない
<	未満
<=	以下
>	を超えて
>=	以上
LIKE	文字列が含まれる
NOT LIKE	文字列が含まれない
IS NULL	空である
NOT NULL	空ではない

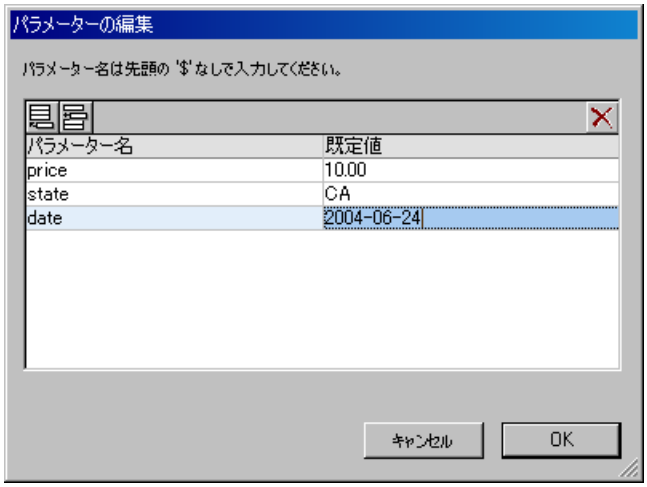
IS NULL や NOT NULL が選択されると、値フィールドが無効になります。値は引用符（または区切り文字）無しで入力しなければなりません。更に、値は対応するDB フィールドと同一の書式である必要があり、そうでない場合は条件式が FALSE と評価されてしまいます。例えば、MS Access DB におけるdate データ型のフィールドに対する条件が StartDate=25/ 05/ 2004 という式であった場合、date データ型の書式が違うということで、式は FALSE と評価されてしまいます（date データ型の書式は YYYY- MM- DD）。

DB クエリにパラメーターを使用する

クエリの作成を行う際には、**パラメーター**の名前を条件の値として入力することができます。パラメーターとはクエリ内でリテラル値の代わりに使用することのできる変数です。条件にパラメーター名を入力すると、対応する値が条件内で使用されます。利用可能なパラメーターは、SPS デザイナーによりSPS 内部で定義されており、パラメーターの表示ダイアログで確認することができます（以下のスクリーンショットを参照）。デフォルトで、パラメーターはSPS により定義された値に割り当てられていますが、コマンドラインによりパラメーターをオーバーライドすることができます（この操作はコマンドラインからドキュメントのコンパイ

ルを行うときに使用することができます)。

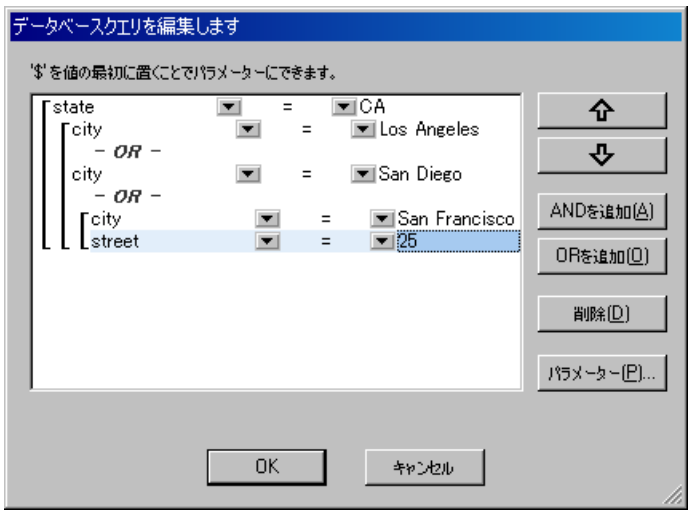
SPS ファイルで定義されたパラメーターを確認するには、**データベースクエリの編集**ダイアログから**パラメーター**ボタンを押下します (以下のスクリーンショットを参照)。



パラメーターの表示ダイアログには、SPS 内部のスタイルシートで定義された**全ての**パラメーターが表示されます。パラメーターの編集は、スタイルシートのデザインにて行う必要があります。

DB クエリの条件を並び替える

DB クエリの論理構造ならびに2つ以上の条件間におけるリレーションシップはグラフィカルに表示されます。論理構造の各レベルは角括弧の形をしたくくりにより示されます。隣り合っている条件や条件のセットがAND 演算子を表し、2つの条件間にOR がある場合、これら2つの条件はOR 論理演算子により関連付けられています。条件の集合は、論理構造を理解しやすい形で表示されます。



上のスクリーンショットに表示されているDB クエリは以下のような意味を持ちます:

state=CA AND (city=Los Angeles OR city=San Diego OR (city=San Francisco AND street=25))

条件や条件のセットを選択し、上下に移動することで、DB クエリの並び替えを行うことができます。条件や条件のセットを移動するには、以下の操作を行ってください:

1. クリックして条件を選択するか、レベルを表す角括弧をクリックしてそのレベル全体を選択する
2. ダイアログの上矢印または下矢印ボタンをクリックする

以下の点に注意してください：

- 同レベルで2つの条件が隣接している場合、条件が移動することで条件の位置が入れ替わります。
- 条件のセット（角括弧内の条件）は同レベル内において位置の変更を行うことはできますが、レベルの変更は行われません。
- 各条件の位置は同じレベルにおいて変更されます。隣接する条件が外側（内側にある場合（つまり同じレベルに無い場合）、選択された条件は**1つのレベルずつ**外側（内側を移動します

DB クエリ内の条件を削除するには、条件を選択した後**削除**をクリックしてください。

DB クエリの修正

DB クエリを修正するには：

1. クエリボタン をクリックします。データベースクエリの編集ダイアログボックスが表示されます。表示されている条件内にある式を編集したり、新たな条件を追加、並び替えや削除を行うことができます。
2. **OK** をクリックします。DB から得られたデータが自動的にAuthentic View に再ロードされ、DB クエリの変更結果が反映されます。

DB テーブルの修正

レコードの追加

To add a record to a DB table:

1. カーソルをDB テーブルの行に移動させ、 （レコードの追加）アイコンまたは （レコードの挿入）アイコンをクリックします。これで新たなレコードが作業用 XML ファイルに作成されます。
2. **「ファイル | 保存」**コマンドをクリックし、新たなレコードをDB 内に追加します。Authentic Viewにおいて、新たなレコードに対応する行がXML テーブルに追加されます。A （追加済み）がこのレコードのAltovaRowStatus にセットされます。

新たなレコードに対するデータを入力すると、太字に下線で入力されます。この表示により、追加されたレコードと既にあるレコード（既にあるレコードがテキストフォーマットプロパティによりフォーマットされていなければ）との区別行うことができます。データ型エラーは赤色で表示されます。

「ファイル | 保存」をクリックすることで、新たなレコードがDB に追加されます。新たなレコードがDB に保存されると、レコードのAltovaRowStatus が初期状態（--）に変更され、通常のレコードとしてAuthentic View に表示されます。

レコードの修正

レコードの修正を行うには、DB テーブル内の必要な箇所にカーソルを移動させ、レコードの修正を行います。表示することのできるレコードの数が制限されている場合、必要なレコードへのナビゲーションを行う必要がある場合もあります（DB テーブルのナビゲーションセクションを参照）。

レコードの修正を行うと、レコード内にある全フィールドのエントリーに下線がひかれ、このレコードにあるプライマリインスタンスのAltovaRowStatus にU （更新済み）がセットされます。このレコードにある全セカンダリインスタンスのAltovaRowStatus にはu （小文字）がセットされます。レコードのプライマリならびにセカンダリインスタンスはDB の構造（ならびにそこから生成されたXML スキーマ）により定義されます。例えば、Address テーブルがCustomer テーブルに含まれている場合、Address テーブルそのもののインスタンスとCustomer テーブルのインスタンス内部にあるインスタンスという、2種類のインスタンスをデザインドキュメントでは扱うことができます。これら2種類のうちどちらであろうと、修正されたほうはプライマリとして修正されます。その他の型はセカンダリ型となります。データ型のエラーは赤色で表示されます。

「ファイル | 保存」をクリックすることで変更がDB に保存されます。修正されたレコードがDB 保存されると、レコードのAltovaRowStatus が初期状態（--）に変更され、通常のレコードとしてAuthentic View に表示されます。

注意：

- Authentic View にてコードのフィールドが1つでも変更された場合、DB にデータが保存される際にはレコード全体が更新されます。.
- データの値 0001-01-01 は DB によっては NULL として定義されており、エラーメッセージを引き起こす場合があります。

レコードの削除

レコードの削除を行うには

1. 削除するレコードにカーソルを移動させ、 アイコンをクリックします。削除されるレコードが取り消し線でマークされます。AltovaRowStatus に以下のような値がセットされます：レコードのプライマリインスタンスは D、セカンダリインスタンスは d、間接的に削除されるレコードは X。間接的に削除されるレコードとは、削除されるレコードの中で、別のテーブルに収められているフィールドのことです。例えば、Customer テーブルに Address テーブルが加えられている状況を考えましょう。Customer のレコードが削除される場合、対応する Address のレコードも間接的に削除されます。Customer テーブルの Address テーブルが削除される場合、Customer テーブル内の Address レコードはプライマリとして削除され、Address テーブル内にあるレコードは（それがインスタンス化されている場合）セカンダリとして削除されます。
2. **「ファイル | 保存」**をクリックすることで、変更を DB に保存します。

注意：データを DB に保存すると、元に戻すコマンドがリセットされ、保存する前に行われた操作に対して元に戻すコマンドが使用できなくなります。

8.3.4 日付の入力

Authentic View では日付の変更を行うために2種類の方法が用意されています：

- 日付の選択を使用することで日付の入力や編集を行う
- 値を入力することで日付の入力や修正を行う

Authentic View のユーザーが使用する方法は SPS により定義されます。このセクションのサブセクションでは両方の方法について記述します。

日付の書式に関するメモ

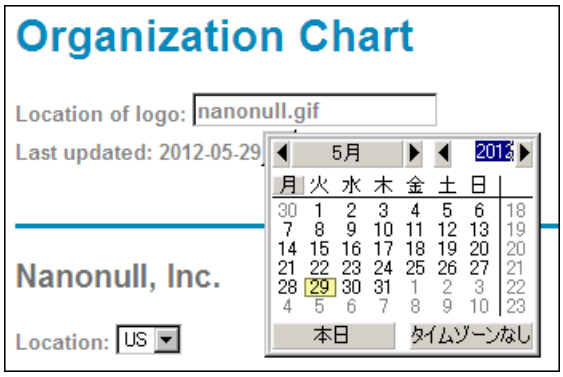
XML ドキュメントでは、複数ある日付のデータ型の1つにより日付情報を保管します。これらのデータ型では値が特定の書式で保管されることで妥当な XML となります。例えば、xs:date データ型では YYYY-MM-DD の書式が求められます。xs:date ノードのデータがこの書式以外で記述された場合、XML ドキュメントは妥当と見なされません。

日付が正しい書式で入力されることを保証するため、SPS デザイナーはデザイン内にグラフィカルな日付の選択を含めることができます。日付の選択を使用することで、選択された日付が正しい書式で入力されることを保証することができます。日付の選択がない場合、Authentic View により入力された日付が正しい書式であることを保証しなければなりません。XML ドキュメントを検証することで必要な書式について有用な情報が提供されます。

日付の選択

日付の選択はグラフィカルなカレンダーで、標準的な書式で日付データを XML ドキュメントに入力するために使用されます。ドキュメント内のデータを処理する際には標準的な書式が重要になります。日付の選択アイコンは編集対象となる日付の隣に配置されます（以下のスクリーンショットを参照）。

日付の選択を表示するには、日付の選択アイコンをクリックします。

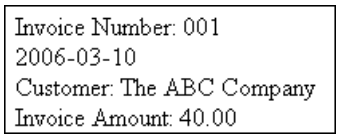


日付の選択を行うには、目的の日、月、そして年をクリックします。日付がXML ドキュメントに入力され、表示される日付もそれに従い変更されます。必要な場合は、タイムゾーンを入力することもできます。

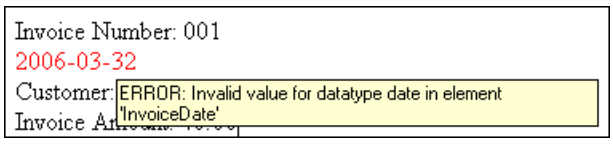
テキストの入力

日付の選択を含まない日付フィールドでは、新たに値を入力することで直接日付を編集することができます。

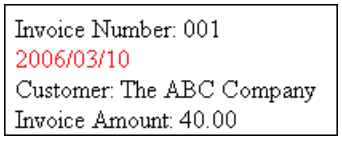
メモ:日付の編集時に書式を変更することはできません。



妥当な範囲を超えた日付を入力すると、日付が赤色になり、エラーであることが表示されます。マウスカーソルを不正な日付に移動させると、エラーメッセージが表示されます（以下のスクリーンショットを参照）。



日付の書式を変更しようとすると、日付が赤色になり、エラーであることが表示されます（以下のスクリーンショットを参照）。



8.3.5 エンティティの定義

ドキュメントがDTD かXML スキーマをベースに作成されているかに関わらず、ユーザーはAuthentic View にて使用することのできるエンティティを定義することができます。一度エンティティを定義すると、これらエンティティがエンティティ入力ヘルパー

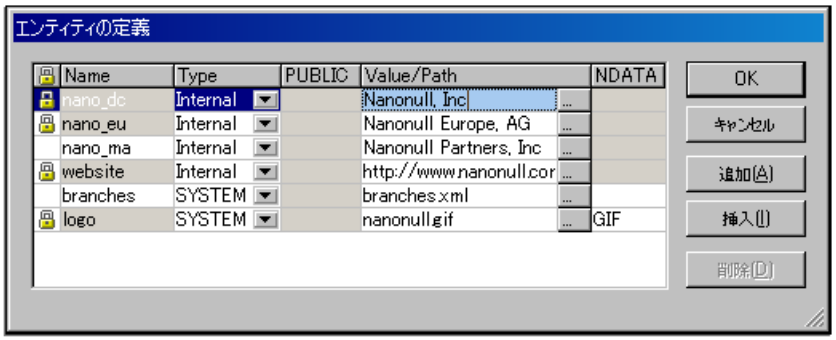
と、コンテキストメニューの**エンティティを挿入**サブメニューに表示されます。エンティティ入力ヘルパーに表示されているエンティティをダブルクリックすると、そのエンティティがカーソル挿入位置に挿入されます。

テキストや XML フラグメント、またはその他の外部リソースをドキュメント内の複数箇所で使用するような場合に、エンティティを有効に使用することができます。エンティティの定義ダイアログでは、目的のデータの短縮形であるエンティティを定義します。エンティティを定義すると、ドキュメント内の複数箇所でそのエンティティを使用することができるようになります。この機能により、作業の時間短縮とメンテナンスが簡単に行えるようになります。

ドキュメント内で使用することのできるエンティティには、大きく分けて2種類あります。**パースされたエンティティ**は（テキスト文字列または XML ドキュメントの）XML データで、**パースされていないエンティティ**はバイナリファイルの様な、XML ではないデータ（通常、画像、音声、またはマルチメディアオブジェクト）となります。各エンティティは名前と値を持ちます。パースされたエンティティの場合、エンティティは XML データに対するプレースホルダーとなります。エンティティの値は、XML データそのものか、XML データを含む .xml ファイルを指す URI となります。

エンティティを定義するには：

1. **Authentic | XML エンティティを定義 ...** をクリックします。エンティティの定義ダイアログが表示されます（以下のスクリーンショットを参照）。



2. エンティティの名前を Name フィールドに入力します。この名前がエンティティ入力ヘルパーに表示されます。
3. Type フィールドにあるドロップダウンリストから、エンティティの種類を選択します。**Internal** エンティティとは、使用されるテキストが XML ドキュメント内部に保管されるエンティティです。**PUBLIC** または **SYSTEM** を選択することで、リソースが XML ファイルの外部に存在していると認識され、それぞれシステム識別子またはパブリック識別子を使ってリソースの識別が行われます。システム識別子とは URI によりリソースの位置を示すものです。パブリック識別子とは場所に依存しない識別子で、プロセッサーによりリソースの識別を行うことができます。パブリックならびにシステム識別子の両方を使用すると、パブリック識別子がシステム識別子の解決を行い、結果としてシステム識別子が使用されます。
4. Type に PUBLIC を選択した場合、PUBLIC フィールドにパブリック識別子を入力してください。Internal または SYSTEM を Type に選択した場合、PUBLIC フィールドは無効になります。
5. Value/Path フィールドでは、以下のデータを入力することができます：

- エンティティの種類が Internal の場合、エンティティの値となるテキスト文字列を入力します。エンティティを区切る目的で引用符を入力しないでください。入力された引用符はテキスト文字列の一部として扱われます。
- エンティティの種類が SYSTEM の場合、リソースの URI を入力するか、参照ボタンを使ってローカルネットワーク上のリソースを選択します。リソースにパースされたデータが含まれている場合、リソースは XML ファイルである必要があります（つまり .xml 拡張子でなければなりません）。リソースは GIF ファイルといったバイナリファイルでも問題ありません。
- エンティティの種類が PUBLIC の場合、このフィールドにシステム識別子を入力する必要があります。

6. NDATA エントリーでは、エンティティがパースされる代わりに、適切なプロセッサーに送信されるよう指定します。従って、NDATA フィールドには、エンティティがパースされていないものであることを示す値が含まれていなければなりません。

ダイアログの機能
エンティティの定義ダイアログでは以下の操作を行うことができます：

- エンティティの追加。

- エンティティの挿入。
- エンティティの削除。
- カラムのヘッダーをクリックすることで、エンティティをアルファベット順に並び替え。1回クリックすることで昇順、2回クリックで降順の並び替えとなります。
- ダイアログボックスならびにカラムの幅を変更。
- エンティティのロック。エンティティが1度 XML ドキュメント内で使用されると、そのエンティティはロックされ、エンティティの編集ダイアログにて編集することができなくなります。ロックされたエンティティは、最初のカラムにロックシンボルが表示されます。エンティティのロックを行うことで、エンティティにおけるXML ドキュメントの妥当性が保証されます（参照されているエンティティが定義されていない場合、ドキュメントは不正となります）。
- 複製されたエンティティがある場合は、その旨が表示されます。

エンティティの制約

- 他のエンティティ内に含まれているエンティティは、ダイアログ、Authentic View 、XSLT 出力にて解決されず、そのようなエンティティのアンパサンドはエスケープされた形で（つまり&）表示されます。
- イメージ以外のパースされていない外部エンティティは、Authentic View で解決されません。デザイン内から URI により参照されるイメージが、外部エンティティを読むように定義されており、URI セットがエンティティ名（例えば 'logo'）の場合、このエンティティ名を、イメージファイルの URI を解決する値を持った、パースされていない外部エンティティとして、エンティティの定義ダイアログで定義することができます（上のスクリーンショットを参照）。

8.3.6 XML 署名

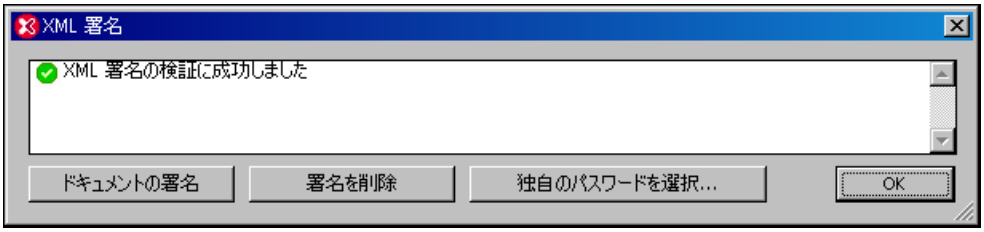
Authentic View に対して XML 署名を埋め込むように SPS ファイルをデザインすることもできます。SPS にて XML 署名が有効になっている場合、Authentic View ユーザーは Authentic XML ファイルにデジタル署名を施すことができます。ドキュメントへの署名が行われると、そのドキュメントに対するいかなる変更を行うことでも、署名の検証が失敗するようになります。署名されたAuthentic XML ドキュメントがAltova 製品のAuthentic View で開かれると、ドキュメントに対する検証処理が開始され、その結果がウィンドウに表示されます。

メモ: 以下にある Altova 製品の Enterprise ならびにProfessional エディションの Authentic View にて XML 署名の使用ならびに検証を行うことができます：Authentic Desktop、Authentic Browser、XMLSpy、StyleVision。

XML 署名アクション

署名に対して以下のAuthentic View ユーザーアクションを行うことができます：

- *証明書 //パスワードの選択*：署名の認証は、パスワードまたは証明書により行われます。署名が作成される際または検証を行う際に、認証オブジェクト（証明書またはパスワード）が必要になります。Authentic XML ドキュメントに関連付けられている SPS ファイルで署名が有効になっている場合、SPS によりデフォルトで使用される証明書やパスワードが指定されている場合があります。デフォルトの証明書またはパスワードが指定されている場合でもそうでない場合でも、Authentic View にて使用する証明書やパスワードの設定を行うことができます。Authentic View ユーザーは XML 署名ダイアログにより、いつでもこの設定を行うことができます（以下のスクリーンショットを参照）。独自の証明書 /パスワードを選択することで、デフォルトの証明書 /パスワードが書き換えられます。独自の証明書 /パスワードはメモリ上に保管され、現在のセッションに対して使用されます。独自の証明書 /パスワードを選択した後に Authentic View ユーザーがファイルやアプリケーションを閉じると、SPS デフォルトの証明書 / パスワードにリセットされます。



- *ドキュメントの署名* : Authentic XML ドキュメントの署名は、自動または手動で行うことができます。自動署名の設定はSPS デザイナーにより署名構成にて行われ、Authentic XML ドキュメントが保存される際に署名が行われます。自動署名オプションが有効になっていない場合、ドキュメントの署名を手動で行うことができます。XML 署名ツールバーアイコン をクリックするか、「Authentic | XML 署名」コマンドを選択して、XML 署名ダイアログにてドキュメントの署名ボタンをクリックします（上のスクリーンショットを参照）。署名を埋め込むかたちでドキュメントの署名を行う場合、ルート要素（ドキュメント要素）最後の子要素としてSignature 要素が許可されてる必要があることに注意してください。そうでない場合、ドキュメントがスキーマに対して不正となります。ドキュメントの署名を行う場合、署名の構成に従い認証オブジェクトの種類や署名の配置位置が決定されます。認証情報へアクセスできることを確認して下さい。この問題に関する詳細については、SPS デザイナーへ相談してください。

- *Authentic XML ドキュメントの検証* : SPS にてXML 署名が有効になっている場合、Authentic View XML ドキュメントがロードされるたびに、署名に対して認証処理が行われます。パスワードまたは証明書の鍵情報がそれぞれSPS や署名に保存されていない場合、パスワードを入力するか証明書を選択するようAuthentic View ユーザーは促されます。埋め込まれた署名が生成される場合、XML ファイルが保存される際に、署名もXML ファイルへ保存されることに注意してください。署名をXML ファイルへ保存したくない場合、（XML 署名ダイアログの署名の削除ボタンにより）明示的に削除する必要があります。同様に、Detached な署名が生成されており、その署名が必要でない場合も、明示的に削除する必要があります。

8.3.7 Authentic View におけるイメージ

Authentic View では最終的な出力ドキュメント（HTML、RTF、PDF、そしてWord 2007）で使用されるイメージを指定することができます。フォーマットやアプリケーションによってはイメージフォーマットがサポートされないということに注意してください。例えば、SVG フォーマットはPDF でサポートされますが、RTF ではサポートされず、HTML にて表示するにはブラウザーアドオンが必要になります。従って、イメージフォーマットを選択する際には、ドキュメントの出力フォーマットでサポートされているフォーマットを選択するようにしてください。殆どのイメージフォーマットは全ての出力ドキュメントでサポートされてます（以下のリストを参照）。

Authentic View はInternet Explorer をベースにしており、お使いのInternet Explorer バージョンにて表示できる殆どのイメージフォーマットを表示することができます。広く使われている以下のイメージフォーマットがサポートされています：

- GIF
- JPG
- PNG
- BMP
- WMF (Microsoft Windows Metafile)
- EMF (Enhanced Metafile)
- SVG（PDF の出力に対してのみ）

相対パス

相対パスはSPS ファイルに対して相対である様に解決されます。

8.3.8 Authentic View におけるキーストローク

Enter キー

Authentic Viewにおいて、**Enter** キーはカーソルが特定の位置にある時に、更なる要素を追加するため使用することができます。例えば、本の中にある章に（スキーマの定義上）複数の段落がある場合、段落のテキスト内部で**Enter** を押下することで、新たな段落が現在カーソルがある段落の直後に追加されます。1つの章の中に1つのタイトルと複数の段落を加えられる場合、章の中（但し段落要素の外）で**Enter** を押下すると、現在位置している章の後に新たな章が作成されます（ここでは複数の章がスキーマにより許されていると仮定しています）。

注意：**Enter** キーにより改行コードは挿入**されません**。このルールはカーソルが段落のようなテキストノード内部にある時でも適用されます。

キーボードの使用

キーボードはタイピングとナビゲーションという一般的な目的に使用されます。

- **Tab** キーによりカーソルを前に移動させ、ノードの直前や直後で停止し、ノードのコンテンツをハイライトすることができます。静的なコンテンツにて停止します。
- ...を追加 や、ノードを追加ハイパーリンクはノードコンテンツとして認識され、Tab キーを使用した際にはハイライトされます。スペースバーや **Enter** キーを押下することでアクティブにすることができます。

8.4 Authentic スクリプト

Authentic スクリプト機能により、SPS デザインに対して更なる柔軟性と双方向性を追加することができます。これらデザインはStyleVision Enterprise ならびにProfessional Edition で作成または編集することができ、Altova 製品のEnterprise ならびにProfessional Edition に搭載されたAuthentic ビューにて閲覧することができます。

以下のテーブルではAltova 製品ごとにサポートされる機能のリストを示します。Authentic Browser Plugin Trusted バージョンはセキュリティー上の観点から内部スクリプトが無効になっていることに注意してください。

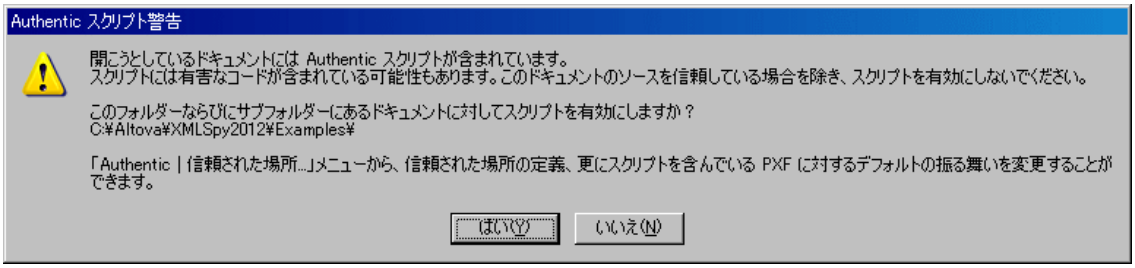
Altova 製品	Authentic スクリプトの作成	Authentic スクリプトの有効化
StyleVision Enterprise	はい	はい
StyleVision Professional	はい	はい
StyleVision Standard *	いいえ	いいえ
XMLSpy Enterprise	いいえ	はい
XMLSpy Professional	いいえ	はい
XMLSpy Standard	いいえ	いいえ
AuthenticDesktop Enterprise	いいえ	はい
AuthenticDesktop Community	いいえ	いいえ
Authentic Browser Plug-in Community **	いいえ	いいえ
Authentic Browser Plug-in Enterprise Trusted ***	いいえ	はい
Authentic Browser Plug-in Enterprise Untrusted	いいえ	はい

* *Authentic ビュー無し*
** *Trusted ならびにUntrusted バージョン*
*** *スクリプトありのデザインは表示可能。内部マクロの実行やイベントハンドリングは無し。外部イベントの実行。*

Authentic スクリプトの動作は全 Altova 製品において同一のものとなり、ある製品に特化したコードや設定は必要ありません。

Authentic スクリプト警告ダイアログ

PXF ファイルまたはSPS へリンクされたXML ファイルにスクリプトが含まれており、そのファイルが開かれた状態でAuthentic ビューへの切り替えが行われると、警告ダイアログが表示されます（以下のスクリーンショットを参照）。



以下のオプションから選択を行うことができます：

- **はい**をクリックすると、ファイルを含んでいるフォルダーがAuthentic スクリプトの信頼された場所リストへ追加されます。今後、信頼されたフォルダー内にふくまれるファイルをAuthentic ビューで開いても、警告ダイアログは表示され

なくなります。信頼された場所のリストへは「**Authentic | 信頼された場所**」メニューコマンドからアクセスならびに編集することができます。

- いいえをクリックすることで、ファイルが含まれているフォルダーを信頼された場所のリストへ追加することなく処理が継続されます。ファイルはスクリプトが無効になった状態で開かれます。ファイルをAuthentic ビューで開くたびにAuthentic スクリプト警告ウィンドウが表示されます。ファイルが含まれているフォルダーを信頼された場所のリストへ追加するには、「**Authentic | 信頼された場所**」メニューコマンドから信頼された場所のダイアログを開き、フォルダーを追加ならびに必要に応じて変更することができます。

信頼された場所ダイアログに関する詳細については、ユーザーリファレンスの「**Authentic | 信頼された場所**」メニューコマンドのセクションを参照ください。

メモ COM インターフェースを介したXMLSpy へのアクセスを行う場合、**セキュリティーのチェックは行われず、「Authentic スクリプト警告」ダイアログは表示されません**(COM インターフェースに関する情報は プログラマーのレファレンス を参照ください)。

Authentic スクリプトの使用方法

SPS デザインを作成したデザイナーは以下にある2つの方法により、Authentic ドキュメントをインタラクティブなものにすることができます:

- スクリプトをユーザー定義のアクション (マクロ) に割り当てることで、要素、ツールバーボタン、そしてコンテキストメニューのアイテムをデザイン
- Authentic ビューイベントに反応するデザインイベントハンドラーを追加

Authentic ドキュメントをインタラクティブなものにするために必要な全てのスクリプトは、StyleVision GUI (Enterprise ならびにProfessional Edition) 内で作成することができます。フォーム、マクロ、イベントハンドラーはStyleVision に搭載されたスクリプトエディター内で作成され、これらスクリプトはSPS 内に保存されます。その後、StyleVision のデザインビューにて、保存されたスクリプトは要素、ツールバーボタン、コンテキストメニューに割り当てられることになります。SPS をベースにしたXML ドキュメントがAuthentic スクリプトをサポートするAltova 製品 (上のテーブルを参照) により開かれると、ドキュメントには新たな柔軟性や双方向性が追加されます。

Authentic スクリプトのドキュメンテーション

Authentic スクリプトに関する情報はStyleVision のドキュメンテーションでご利用になれます。Altova ウェブサイトの製品ドキュメンテーションページから参照することができます。

9 HTML とCSS

XMLSpy では、HTML、とCSS ドキュメントに対するインテリジェントな編集機能が提供されます。全てのドキュメントはテキストビューにて編集することができ、アクティブなHTML はブラウザービューにてプレビューを行うことができます。

各ドキュメントの種類に関するインテリジェント編集機能についてはこのセクション以下のサブセクションに記されます：HTML、とCSS。

9.1 HTML

テキストビューではHTML ドキュメントを編集することができ、編集された内容を、すぐにブラウザービューで確認することができます。テキストビューでは以下にあるような数々のHTML 編集機能が提供されています。詳細はテキストビューのセクションを参照ください。

サポートレベル

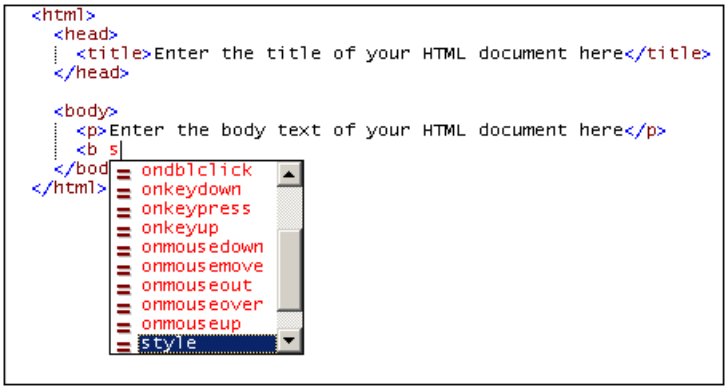
XMLSpy ではHTML 4.0 ならびにHTML 5.0 がサポートされます。入力ヘルパーならびにインテリジェントな編集機能が対応したHTML のバージョンに対して利用可能です。これらの機能を以下に記します。

入力ヘルパー

HTML ドキュメントがアクティブになっていれば、要素、属性、エンティティ入力ヘルパを利用することができます。入力ヘルパーの内容は文脈により変化し、現在カーソルのある位置に入力することができるアイテムが入力ヘルパーには表示されます。HTML 入力ヘルパーはテキストビューにあるその他の入力ヘルパーと同じように使用することができます。

自動補完

HTML ドキュメントにマークアップを入力する際、XMLSpy は自動補完ヘルプを表示します。ポップアップにはカーソルのある位置に挿入することができるノード全てが表示され、入力がなされるごとに、リスト内にある最も近いマッチの先頭に選択が移動します（次のスクリーンショットを参照）。選択されたアイテムをクリックすることで、そのアイテムがカーソル位置に挿入されます。



要素に対する自動補完は、ノード開始タグが入力されたときに表示されます。ノードの開始タグがドキュメントに入力されると、その終了タグも自動的に挿入されます。この機能により整形式が保証されます。

ノードの開始タグにて要素名のあとにスペースが入力されると、属性に対する自動補完が表示されます。自動補完ポップアップにある属性名をクリックすると、属性名と引用句が入力され、カーソルが引用句の中間に移動します。

エンティティ入力ヘルパーにはHTML 4.0 、とHTML 5.0エンティティセットLatin-1、特殊な文字、そしてシンボルが含まれています。

HTML 情報ウインドウ

（以下のスクリーンショットにある）HTML 情報ウインドウを使えば、アクティブなHTML ファイルに素早くアクセスすることができます。例えば、HTML ファイルがXMLSpy でアクティブであれば、HTML 情報ウインドウ内にあるMozilla Firefox アイテムをダブルクリックすることで、Mozilla Firefox が起動され、そのHTML ドキュメントがロードされます。

以下の点に注意してください：

- *プログラムからHTML を開く*アイテムの右にあるアイコンを使うことで、*プログラムからHTML を開く*リストにアプリケーションを追加することができます。システムにインストールされているブラウザーまたはその他の (テキストエディターのような) アプリケーションを追加することができます。通常追加されるのはブラウザーまたはエディターアプリケーションです。
- *アプリケーション*が*プログラムからHTML を開く*リストに追加されたら、(**「インストールされたブラウザーを追加」**コマンドを使用した場合を除いて) アイテムを選択して、**[F2]** を押すことで、リスト内に表示されるアプリケーションの名前を変更することができます。
- リストの各アイテムの右側にあるアイコンを使えば、(i) アプリケーションを開始したり、(ii) アプリケーションを開始した後でリンクされたHTML ファイルを開くか、(iii) リストからアプリケーションを削除することができます。アプリケーション名をダブルクリックすることで、リンクされたそのアプリケーションでHTML ファイルが開かれます。
- プログラムからHTML を開くリストに表示されるアプリケーションは、CSS 情報ウィンドウのアプリケーションアイテムと連動しています。

DTD の割り当て

[DTD/スキーマ] メニューを使えば、XHTML ドキュメントに対してDTD またはXML スキーマを参照、割り当てることができます。XHTML ドキュメントの編集はXML ドキュメントと同じように行うことができます。

ブラウザービューコマンド

ブラウザービューコマンドは **[ブラウザー]** メニューから使用することができます。

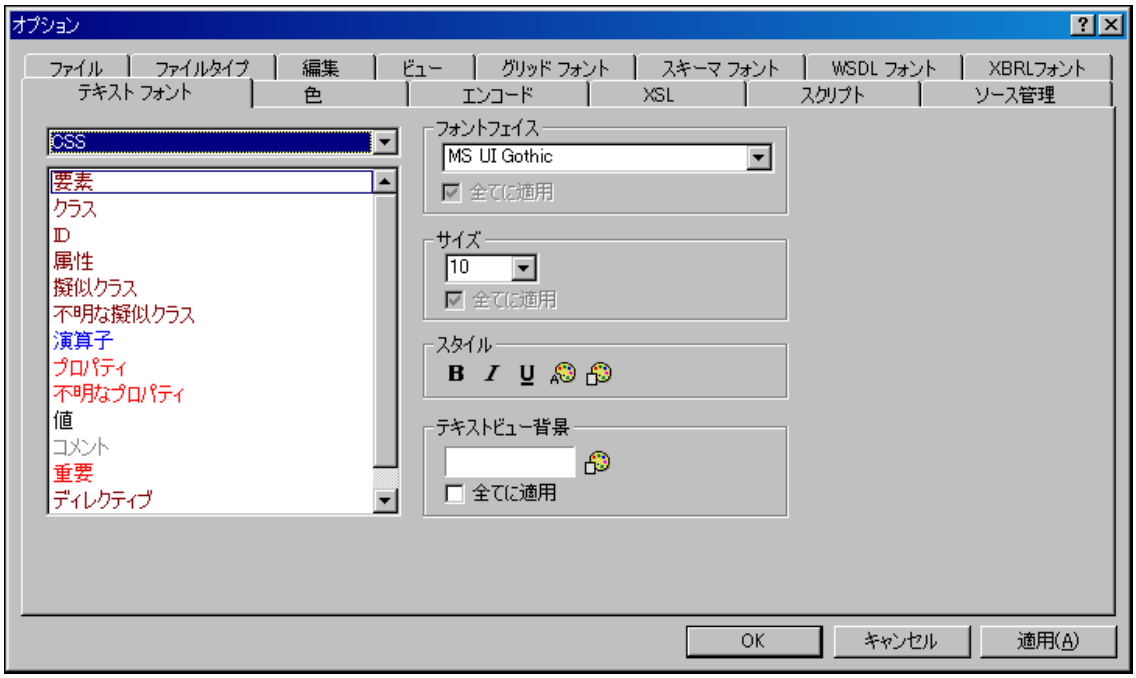
9.2 CSS

テキストビューのインテリジェント編集機能を使用して CSS ドキュメントの編集を行うことができます。CSS の編集に利用することができる編集機能は以下のとおりです。

構文による色分け：CSS はセレクター、(1つ以上の) プロパティ、そしてそれらプロパティの値により構成されます。これら3つのコンポーネントは、細かな指定を行うため、更に細かなカテゴリに分割されることもあります。例えば、セレクターにより指定されるものにクラス、擬似クラス、ID、要素、または属性があります。更に、CSS ドキュメントは (例えばコメントのような) ルール以外のアイテムも含むことができます。テキストビューにおいて、そのようなアイテムのカテゴリは、オプションダイアログ (以下を参照) により指定されるように、異なる色で表示されます。

```
.header
{
    font-family: "Arial", sans-serif;
    font-weight: bold;
    color: red;
}
```

(以下のスクリーンショットにあるように) オプションダイアログのテキストフォントタブから、様々な CSS コンポーネントの色を指定することができます。左上にあるコンボボックスから CSS を選択し、各 CSS アイテムに対して必要な色をスタイルペインから選択してください。



ソースの折りたたみ

ソースの折りたたみは、ソースの折りたたみ余白内で +/- サインにより表示される、各 CSS ルールを展開または折りたたむ機能を指します。余白はテキストビュー設定ダイアログにより切り替えのオンとオフを行うことができます。ノードが折りたたまれると、視覚的に省略記号により表示されます。ルールが折りたたまれると、視覚的に省略記号により表示されます。マウスのカーソルが省略記号にポイントされると、折りたたまれたルールの内容がポップアップ内に表示されます。内容がポップアップに大きすぎる場合、これは、ポップアップの下に省略記号で表示されます。

テキストの折りたたみ (またはソースのおりたたみ) は XQuery 中かつ、XQuery コメント、XML 要素、XML コメントにより有効化され、これらのノードを展開または折りたたむ機能を意味します。このようなノードは、+/- サインによりソース折りたたみ余白内で表示されます (下のスクリーンショット参照)。余白はテキストビュー設定ダイアログにより切り替えのオンとオフ

を行うことができます。ノードが折りたたまれると、視覚的に省略記号により表示されます（*下のスクリーンショット参照*）。マウスのカーソルが省略にポイントされると、ポップアップで折りたたまれたノードの内容が表示されます（*スクリーンショット参照*）。内容がポップアップでの表示に大きすぎる場合、ポップアップの下の部分に省略記号で表示されます。

テキストツールバー内の**[全ての折りたたみを切り替える]** アイコン は、**全ての**ノードを展開された書式でまたはトップレベルのドキュメント要素まで折りたたみ表示します。

メモ 中かっこのペアは、ルールを区切り（*上のスクリーンショット*）中かっこの前または後ろにカーソルがポイントされると太字で表示されます。これにより特定のルールがどの位置で開始し終了するかの定義が表示されます。

折り返し余白 ：上のスクリーンショットにて示されるように、各ルールの左側にあるマイナスまたはプラスアイコンをクリックすることで、ルールを縮退、展開することができます。ルールの区切りを表す波かっこにカーソルを移動させると、かっこが太字で表示されることにも注目してください。この機能により、特定のルール定義がどこで始まりどこで終わるのか素早く確認することができます。

CSS アウトライン ：以下のスクリーンショットにある CSS アウトライン入力ヘルパーでは、ドキュメント内にあるセレクターが表示されます。CSS アウトラインにあるセレクターをクリックすると、ドキュメント内の対応する箇所がハイライトされます。左下にあるスクリーンショットでは、セレクターの並べ替えは行われず、ドキュメント内で表示される順番通りに表示されています。右下のスクリーンショットでは、（ツールバーアイコンの）アルファベット順の並び替え機能が有効になっており、セレクターがアルファベット順に並べ替えられています。

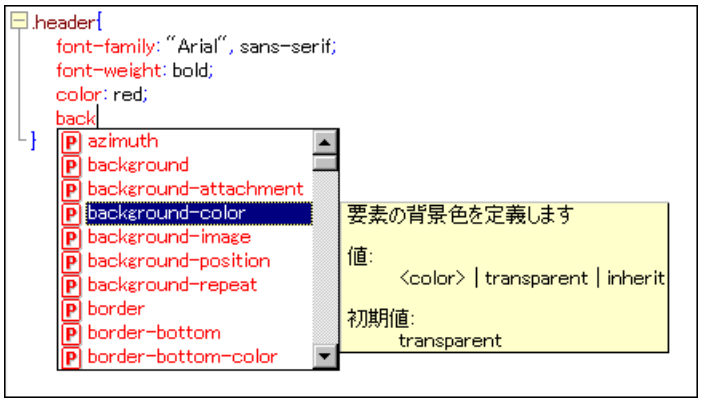
以下の点に注意してください：(i) セレクターの並べ替えを行う際には、ピリオド、ハッシュ、コロンといった セレクター全ての箇所が並べ替え評価の対象になり、(ii) 複数のセレクター（例：`h4, h5, h6 {...}`）が1つのルールとしてグループ化されている場合でも、CSS アウトラインではグループ内の各セレクターは別々に表示されます。

CSS アウトラインイン入力ヘルパーにあるツールバーアイコンを左から順に説明します：

	ドキュメントとの自動同期の有効/無効を切り替えます。自動同期が有効になっている場合、ドキュメント内で入力されたセレクターが（入力中でも）入力ヘルパーに表示されるようになります。
	入力ヘルパーの内容を現在のドキュメントの状態から更新します
	アルファベット順による並び替えの有効/無効を切り替えます。オフになっている場合、セレクターはドキュメント内に記述されている順序で表示されます。アルファベット順で並び替えられた場合、ハッシュ記号により記述されるID が最初のセレクターとして表示されます。

プロパティ入力ヘルパー：以下のスクリーンショットにある) プロパティ入力ヘルパーには、全てのCSS プロパティがアルファベット順に表示されます。プロパティのアイテムをダブルクリックすることで、カーソルのある位置にそのプロパティが挿入されます。

プロパティの自動補完とツールチップ：プロパティの名前を入力しようとすると、XMLSpy により、入力された文字から始まるプロパティのリストが表示されます。その他にも、カーソルをプロパティ名の中まで移動させ、**「Ctrl」 + 「Space」** を押下することでCSS プロパティのリストが表示されます。

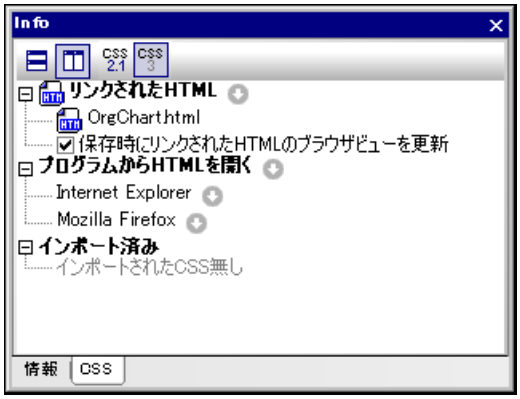


リストをスクロール、または矢印キーによりリストのナビゲーションを行うことで、プロパティの定義と使用可能な値が記されたツールチップが表示されます。表示されるツールチップはハイライトされたプロパティのものです。選択されたプロパティを挿入するには、**「Enter」** を押下するか、プロパティをクリックしてください。

CSS 情報ウィンドウ

CSS ファイルがアクティブになっていれば、CSS 情報ウィンドウが有効になります。CSS 情報ウィンドウでは以下の機能が利用可能です：

- CSS 2.1 とCSS 3.0 との切り替えを行うことができます。GUI における入力ヘルパーとインテリジェントな編集機能が、情報ウィンドウで選択されているCSS のバージョンにより切り替わります。
- CSS ファイルをHTML ファイルにリンクすることができます。この機能によりCSS ドキュメントの修正により反映される結果をただちに確認することができます。他にも、リンクされたHTML を複数のブラウザーで表示することができ、CSS ドキュメントの修正を異なるブラウザーにて確認することができます。
- CSS 情報ウィンドウにはインポートされたCSS スタイルシートが表示され、アクティブなCSS ドキュメントにインポートされたファイル構造の概要を知ることができます。



以下の点に注意してください：

- 情報ウィンドウのツールバーにはCSS 2.1 とCSS 3.0 のアイコンが含まれます。目的のバージョンを選択することで、選択したバージョンのCSS に対応した入力ヘルパーならびに編集機能を使用することができます。
- アクティブなCSS ドキュメントからリンクすることのできるHTML ファイルは1つだけです。*リンクされたHTML* の左にあるアイコンをクリック **「HTML リンクを設定」**コマンドを選択し、目的のHTML ファイルを選択します。選択されたファイルが*リンクされたHTML* アイテムの下に表示されます。このリンクを作成してもCSS やHTML ドキュメントへの修正が施されるわけではありません。このリンク機能により、アクティブなCSS ファイルを適用して、ファイルを試験することができます。
- リンクされたHTML ファイルをダブルクリックすることで、そのHTML ファイルがXMLSpy により開かれます。
- ツールバーアイコンにより、CSS ファイルとHTML ファイルを水平、または垂直に配置することができます。
- 変更されたCSS ドキュメントを保存する際、XMLSpy で開かれているHTML ファイルを自動的に更新することができます。この機能を有効にするには、*保存時にリンクされたHTML のブラウザービューを更新* ボックスにチェックを入れてください。この機能による自動更新は、編集されたCSS ドキュメントへの参照を行っているHTML ファイルに対してしか行われないことに注意してください。
- リンクされたHTML ファイルを変更するには、**「HTML リンクを設定」**コマンドから、別のHTML ファイルを選択してください。
- HTML ファイルへのリンクを削除するには、*リンクされたHTML ファイル* の右にあるアイコンをクリックし、**「リンクを削除」**コマンドを選択してください。
- *プログラムからHTML を開く* の右にあるアイコンを使うことで、リストへアプリケーションを追加することができます。コンピューターにインストールされたブラウザーまたは（テキストエディターの様な）任意のアプリケーションをメニューコマンドに追加することができます。割り当てを行うアプリケーションは、通常ブラウザーかエディターです。
- アプリケーションが*プログラムからHTML を開く* リストに追加された後は、（**「インストールされたブラウザーを追加」**コマンドにより追加されたものを除いて）アイテムを選択して、**「F2」**キーを押下することにより名前の変更を行うことができます。
- *プログラムからHTML を開く* に表示されているアプリケーションの右にあるアイコンからは、(i) アプリケーションを開く (i) アプリケーションを開き、リンクされているHTML ファイルを開く (ii) リストからアプリケーションを削除するコマンドを選択することができます。アプリケーション名をダブルクリックすることで、リンクされているHTML ファイルをそのアプリケーションで開くことができます。
- *プログラムからHTML を開く* リストに（から）追加（削除）されたアプリケーションは、HTML 情報ウィンドウのアプリケーションとも連動してしています。
- *インポート済み* アイテムには、アクティブなCSS ファイルにインポートされたCSS ファイルが表示されます。

10 JSON と JSON スキーマ

JSON（ジェイソン、JavaScript Object Notation）は、軽量のデータストレージとJavaScript 構文を使用する交換形式です。また、XML のように人間が判読できる、テキストのみの書式です。JSON テキストは、すべてプログラミング言語で読み込まれ、使用されることができるので、広範囲におよび、特にウェブ上で、データ交換書式として使用されるようになりました。

IDE 機能の一部として、XMLSpy は、JSON *データ(インスタンス)ドキュメント*の編集および検証をサポートし、構文的にも、文法的にも正確な JSON *スキーマ*ドキュメントの作成をサポートします。

XMLSpy は Avro と Avro *スキーマ*へのサポートを提供します。

JSON5

ECMAScript 5 拡張を追加する JSON の拡張が JSON5 です（詳細に関しては json5.org を参照してください）。JSON5 は、JavaScript の厳密なサブセットあり、新規の JSON データ型を追加せず、全ての既存の JSON コンテンツと作業することができます。

JSON *インスタンス*ドキュメントのために使用することのできる、すべての XMLSpy 機能は、JSON5 *インスタンス*ドキュメントに対して使用することができます。しかしながら、JSON5 と JSON の間には以下の主要な違い、および、これら2種類のフォーマットを XMLSpy が扱う方法の違いが存在します：

- JSON5 は JSON の公式な継承ではありません。このため、自身のファイル拡張子を使用しています：`.json5`。
- デフォルトでは、XMLSpy は、`.json` ファイル拡張子を JSON *インスタンス*ドキュメントとして認識します。`.json5` ファイル拡張子を持つ*インスタンス*ドキュメントを JSON5 *インスタンス*ドキュメントとして認識します。
- JSON5 *スキーマ*は JSON *スキーマ*に対して検証することができます。Avro *インスタンス*を表す、JSON *インスタンス*ドキュメントは、JSON *スキーマ*（JSON *インスタンス*）と Avro *スキーマ* Avro *インスタンス*の両方に対して検証することができます。詳細に関しては、*JSON ドキュメントの検証* を参照してください。

このドキュメント内では、*JSON インスタンス*という*用語*は、指定されない限り、JSON と JSON5 *インスタンス*ドキュメントを指します。次を参照してください：JSON5 と JSON 間の違い。

XMLSpy 内の JSON、JSON スキーマ

JSON *インスタンス*と JSON *スキーマ*? の両方のドキュメントの型は、JSON フォーマットで書かれており、有効性のために整形式の JSON ルールに準拠しなければなりません。(*インスタンス*と*スキーマ*）の両方のドキュメントは、通常、`.json` ファイル拡張子を持っています。JSON *インスタンス*は、テキストビューとグリッドビューで編集することができ、JSON *スキーマ*ドキュメントをこれら2つのビュー、および、視覚的な*スキーマ*ビューである JSON *スキーマ*ビューで、編集することができます。

XMLSpy は、JSON *インスタンス*と JSON *スキーマ*ドキュメントとの作業のために以下をサポートしています：

- テキストビューには、構文の色分け、構文のチェック、JSON *スキーマ*の関連性がある場合の*インスタンス*ドキュメント内の自動補完、余白の折り返し、構造マーキングなどの機能を使用して、有効な JSON *インスタンス*と JSON *スキーマ*ドキュメントの編集を素早く行うことができます。テキストビューには、*インスタンス*および*スキーマ*ドキュメントの検証が搭載されています。
- グリッドビューには、ドキュメント構造を更にビジュアル化するテーブルグリッド構造が搭載されています。グリッドビューで直接編集することができます。切り替え可能なテキストビューおよびグリッドビューは、編集に関するニーズに応えます。グリッドビューには、*インスタンス*および*スキーマ*ドキュメントの検証が搭載されています。
- テキストビューおよびグリッドビュー内の JSON *インスタンス*ドキュメント検証。情報ウィンドウ内で割り当てられ

たJSON スキーマに対して検証は実行されます。
- JSON スキーマビュー は、JSON スキーマをグラフィカルなレイアウトで表示します。これにより、JSON スキーマを素早く作成するためのドラッグアンドドロップ機能を (テキスト入力に加え) 使用することができます。ビュー内の入力ヘルパーは、編集入力を与えます。更に、スキーマの有効性が継続的にチェックされ、エラーが発生するとフラグされます。

JSON インスタンス: 既存のインスタンスドキュメントを開き、新しいインスタンスドキュメントを作成する

- **[オプション] | [ファイルの型]** タブ内で、JSON/JSON5 インスタンスドキュメントを開くためのデフォルトのビュー (テキストビュー または グリッドビュー) を設定することができます。既存のJSON/JSON5 ドキュメントは、選択されたデフォルトのビューで開かれます。 テキストビュー と グリッドビュー をいつでも切り替えることができます。
- 新しいJSON または JSON5 インスタンスドキュメントを作成するためには、**「ファイル | 新規作成」** をクリックして、`json: JavaScript Object Notation` または `json5: JSON with ECMAScript 5 extensions.` をそれぞれ選択します。任意で、新しいインスタンスファイルのためのJSON 、または、(JSON5 ではなくJSON のための) Avro スキーマファイルを選択するように問われます。スキーマを割り当てるには、割り当ては情報ウィンドウ内で入力されます。**[オプション] | [ファイルの型]** タブ内の設定に従い、新規 インスタンスドキュメントは、テキストビュー または グリッドビューで開かれます。

JSON スキーマ: 既存の スキーマを開く または、新しいスキーマを作成する

- 既存のJSON スキーマ ドキュメントを JSON スキーマビューで開きます。テキストビュー または グリッドビュー にいつでも切り替えることができます。
- 新規 JSON スキーマ ドキュメントを作成するには、**「ファイル | 新規作成」** をクリックして、`json: JSON スキーマ` を選択します。ドキュメントの最初に `$schema` キーワードを持つ、新規 JSON スキーマ ドキュメントが JSON スキーマビューで開かれます。テキストビュー または グリッドビュー にいつでも切り替えることができます。

これら全てのビュー (テキストビュー グリッドビュー および JSON スキーマ) に関しては、このセクションのサブセクションで説明されています。

このセクション
このセクションは以下のトピックに整理されています:

- JSON データについて は、JSON ドキュメントの基本的情報について説明しています。
- JSON スキーマについて は、JSON スキーマについてまたその作動について説明しています。
- テキストビュー内の JSON ドキュメント は、テキストビューの JSON に関連した機能と作業する方法について説明しています。
- グリッドビュー内の JSON ドキュメント は、グリッドビュー内の JSON ドキュメントの編集方法について説明しています。
- JSON データドキュメントの検証 は、JSON ドキュメントへの JSON スキーマの割り当てと JSON ドキュメントの検証について説明しています。
- JSON インスタンスからの JSON スキーマの生成 は、インスタンスからのスキーマの生成機能について説明しています。
- JSON スキーマビュー は、ビューの JSON スキーマ編集機能および JSON プロジェクトを作成する際での使用方法

10.1 JSON データ

このセクションは、どのように JSON データが構成されているかを簡単に説明します。JSON データは通常ドキュメントに保管されていますが、JSON (インスタンス) ドキュメントまたは他の型内でデータフラグメントとして保管されることもできます。JSON データフラグメントまたはドキュメントは、下で定義されているように JSON データ構造です。

XMLSpy は、ECMScript 5 拡張を追加する JSON の拡張である**JSON5** もサポートします。詳細に関しては、json5.org を参照してください。

JSON オブジェクトと配列

(通常ファイル拡張子 `.json` と共に保存される) JSON ドキュメントは、次のコアデータ構造により構築されます :

オブジェクト

オブジェクト は、中かっこで区切られており、順序のないゼロまたは `key:value` ペアのコレクションです。これらの `key:value` ペアは、**オブジェクトのプロパティ**です。キーは常に文字列である必要があり、引用符で囲まれている必要があります。(プロパティ名と呼ばれる) キーは、コロンにより値から区切られています。プロパティ値は、JSON データ型 (*下のリスト参照*) のすべての値を使用することができます。プロパティはカンマで区切られています。下のリストは、(アトミック型の値を持つ) 3 つのプロパティを持つオブジェクトの例です :

```
{
  "emailtype":"home",
  "emailaddress":"contact01.home@altova.com",
  "citycode":22
}
```

配列

配列 は、角かっこで区切られており、カンマで区切られたゼロまたは複数の**アイテム**のリストです。これらのアイテムは、JSON データ型のすべてを使用することができます (*下のリスト参照*)。

- *2 つのオブジェクトを含む配列*

  この配列は、(それぞれ中かっこにより囲まれた) 2 つのオブジェクトから構成されています。配列自身は、角かっこにより表示されています。

```
[
    {
    "emailtype":"home",
    "emailaddress":"contact01.office@altova.com",
    "citycode":22
    },
    {
    "emailtype":"office",
    "emailaddress":"contact01.office@altova.com",
    "citycode":22
    }
]
```

- *オブジェクトのプロパティの値である配列*

  この リストは、3 つのキーを持つオブジェクトです。各値は、タプル (シーケンス) を含む配列です。(タプルは、ひとつの次元からなる配列と考えられることができます)。各タプル内の 3 つのアイテムは単純型です。

```
{
    "x":[ 1,2,"abc" ],
    "y":[ 3,4,"def" ],
    "z":[ 5,6,"ghi" ]
}
```

JSON データ型

オブジェクトプロパティ値と配列アイテムは次の型であることができます：

- `string` (引用符で囲まれている必要があります)。文字列は `date-time` または `email` フォーマットなどの追加的にフォーマットを持つように指定されることができます。
- `number`: 小数点の部分を持つ数値。整数を含みます。
- `integer`: 小数点の部分を持たない数値。`number` 型のサブセット
- `boolean` (引用符で囲まれていない `true/false`)
- `object`: 他のオブジェクト内で使用されると、データをネストすることができます。
- `array`: オブジェクトにより許可されているより複雑な構造を作成する機能を与えます。
- `null` (引用符で囲まれていない `null`)

JSON データのサンプル

下は、JSON データフラグメントのサンプルです。ドキュメントがどのようにオブジェクトと配列に構造化されているかに注意してください。また、キーの値のデータ型、引用内の文字列の値、他の型は緑に色付けされている点などに注意してください。

```
{
  "first":"Jason",
  "last":"Jones"
  "isManager":true,
  "age":35,
  "address":{
    "street":"Jason Avenue",
    "city":"Jasonville",
    "state":"JS",
    "postcode":"JS12 ON34"
  },
  "phone":[
    {
      "type":"home",
      "number":"12 3456-7890"
    },
    {
      "type":"office",
      "number":"789 012-34567"
    }
  ],
  "children":[],
  "partner":null
}
```

JSON5 と JSON 間の違いについて

JSON5 は、JavaScript の厳密なサブセットあり、新規の JSON データ型を追加せず、全ての既存の JSON コンテンツと作業することができます。主要な違いは以下にリストされるとおりです：

- JSON5 はコメントをサポートします。コメントは以下のように区切ることができます: `// comment //` または `/*`

`comment */`.

- JSON5 内では、`key:value` ペアのキーは、引用符で囲まれる必要はありません。
- JSON5 内では、文字列を複数のラインにわたり作成することができます。
- JSON5 ドキュメントを Avro に対してではなく JSON スキーマに対して検証することができます。

10.2 JSON スキーマ

XML スキーマが XML ドキュメントの構造とコンテンツを指定するように JSON スキーマは JSON ドキュメント内の JSON データが整理されるかを指定します。どのデータフィールドが期待されるか、また、値がどのように表示されるかを指定します。JSON スキーマ仕様と JSON スキーマについての情報はここで確認することができます。

JSON スキーマ自身は、JSON オブジェクトです。構文的には、スキーマ全体が JSON オブジェクトの区切りである中かっこで囲まれています (*下のリスト参照*)。スキーマは JSON 構文で書かれており、通常 `.json` の拡張子を持ちます。トップレベルオブジェクトの最初のキーワードである **`$schema`** キーワードを使用して、JSON スキーマとして表示されています。また以下の値を持つべきです : **`"http://json-schema.org/draft-04/schema#"`** 。(`$schema` キーワードは任意ですが奨励されます、またトップレベルオブジェクト内のすべての箇所では発生することができます。)

```
{
  "$schema":"http://json-schema.org/draft-04/schema#",
  …
}
```

メモ: `$schema` キーワードは、スキーマの最新のバージョンを指定する以下の値を持つことができますが **`"http://json-schema.org/schema#"`**、特定のバージョンを指定するを使用することが奨励されます。現在は、以下のとおりです : **`"http://json-schema.org/draft-04/schema#"`**.

XMLSpy の JSON スキーマビュー内で、グラフィック性に富んだ JSON スキーマを作成することができます。作成方法は、JSON スキーマビュー に説明されています。スキーマ編集機能に加え、JSON スキーマビューで以下のスキーマに関連した機能を使用することができます :

- XMLSpy の JSON バリデータを使用して検証 : JSON インスタントドキュメントに JSON スキーマを割り当て、XMLSpy 内から、インスタントドキュメントを検証します。 詳細に関しては、JSON ドキュメントの検証 を参照してください。
- JSON インスタンスからの JSON スキーマの生成: JSON インスタントドキュメントが既存の場合、JSON スキーマを生成することができます。その後、必要に応じて、スキーマを編集することができます。
- JSON と XML の間の変換: 2 つのフォーマットのドキュメント間での変換が行えます。

用語

GUI とこのドキュメント一般的に使用される JSON スキーマ用語が以下にリストされています。

用語	定義
スキーマ	JSON スキーマドキュメント内のトップレベルのスキーマオブジェクトであるスキーマファイル
オブジェクト	ゼロまたは複数のプロパティを含む JSON 型
プロパティ	オブジェクトの `key:value` ペア。値は任意の JSON データ型であることができます。
キーワード	オブジェクトの `key:value` ペアの `key` 部分。常に文字列です。
サブスキーマ	演算子または依存性の子であるオブジェクト
定義	任意の JSON 型の完全な詳細。定義は、グローバルまたはローカルであることができます。
配列	任意の JSON データ型のゼロまたは複数のアイテムのコンマにより区切られた順序リスト。
アトミック型	`string`, `number`, `integer`, `boolean`, および `null` JSON データ型
型セレクター	すべておよび複数型の選択する `any` および `multiple` 型
演算子	子として定義に追加することのできる発生セレクター。*詳細に関しては、演算子を参照してく*

	だされ。

JSON データ型

オブジェクトプロパティ値と配列アイテムは次の型であることができます：

- `string` (引用符で囲まれている必要があります)。文字列は `date-time` または `email` フォーマットなどの追加的にフォーマットを持つように指定されることができます。
- `number`: 小数点の部分を持つ数値。整数を含みます。
- `integer`: 小数点の部分を持たない数値。`number` 型のサブセット
- `boolean` (引用符で囲まれていない `true/false`)
- `object`: 他のオブジェクト内で使用されると、データをネストすることができます。
- `array`: オブジェクトにより許可されているよりも複雑な構造を作成する機能を与えます。
- `null` (引用符で囲まれていない `null`)

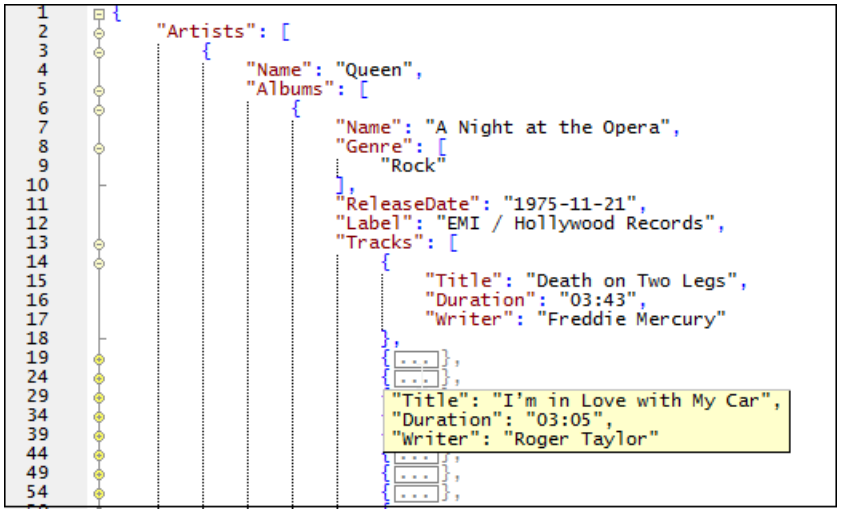
10.3 テキストビュー内の JSON ドキュメント

Altova ウェブサイト JSON エディター

JSON スキーマ、Avro スキーマ、および JSON/JSON5 インスタンスドキュメント (JSON フォーマットの Avro データインスタンスを含む) 場合、テキストビューの機能は以下を含みます：折り返し余白、構造マーキング、構文色分け、構文チェックおよび自動補完。XMLSpy はまた両方の方向の JSON と XML 間の変換 を行うことができ、JSON インスタンスからの JSON スキーマの生成を可能にします。

折りたたみ余白

ソースの折りたたみは、JSON キーワードと定義に対して有効化され、ノードの展開および折りたたみを行う機能を意味します。このようなノードは +/- サインによりソースの折りたたみ余白内で表示されます (*下のスクリーンショット参照*)。余白は、テキストビュー設定ダイアログ内で切り替えオンとオフすることができます。ノードが折りたたまれると、視覚的に省略記号で表示されます (*下のスクリーンショット参照*)。マウスのカーソルが省略記号にポイントされると、折りたたまれたノードの内容がポップアップで表示されます (*スクリーンショット参照*)。内容がポップアップに大きすぎる場合、これは、ポップアップの下に省略記号で表示されます。



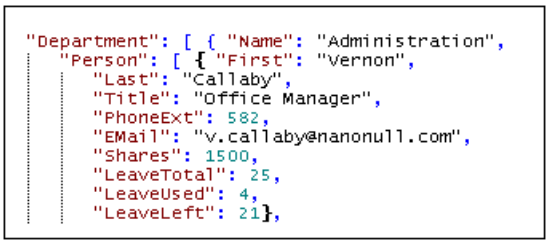
テキストツールバー内の **[全ての折りたたみを切り替える]** アイコン は、**全ての**ノードを展開された書式でまたはトップレベルのドキュメント要素まで折りたたみ表示します。

ノードの +/- アイコンをクリックすると以下のオプションを使用することができます：

Click [-]	ノードを折りたたみます。
Click [+]	子孫ノードが展開されて表示されるか、ノードが折りたたまれた前の状態に従い折りたたまれ、ノードが展開されます。
Shift+ [-]クリック	すべての子孫ノードを折りたたみ、展開フォーム内でクリックされたノードを残します。
Ctrl+ [+]クリック	クリックされたノードとそのすべての子孫ノードを展開します。

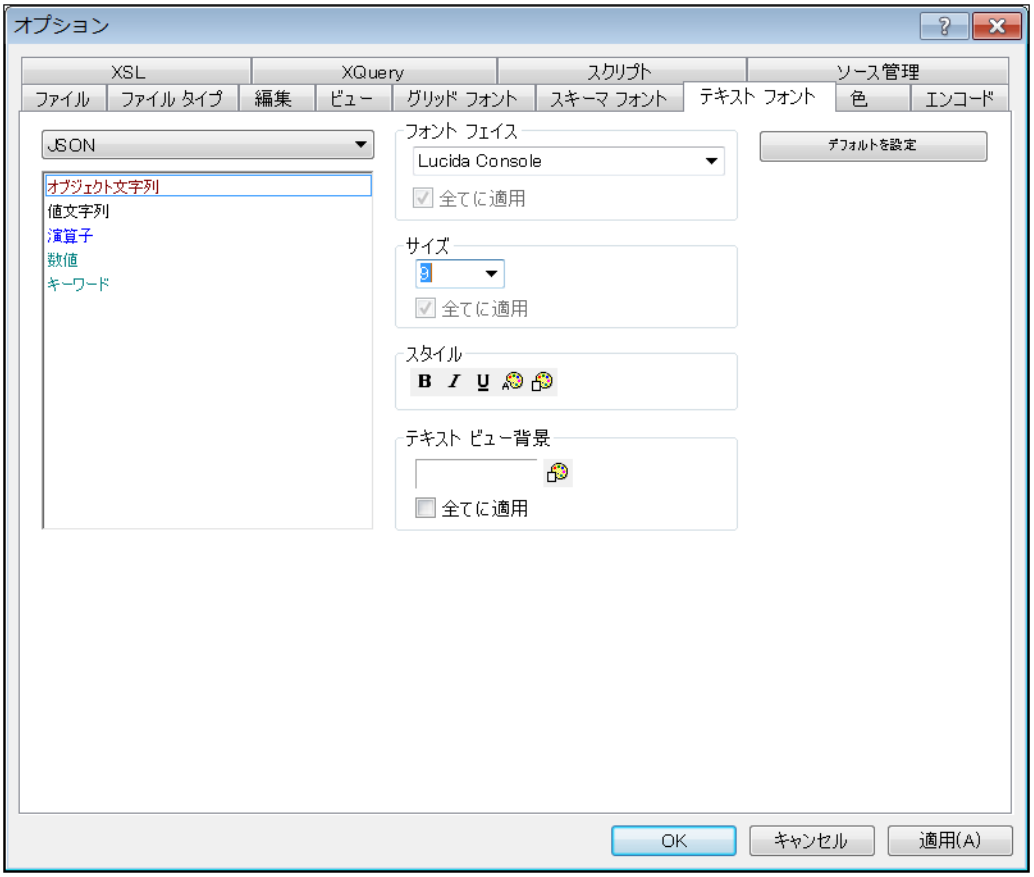
構造マーキング

JSON オブジェクトまたは配列の区切りを行う波かっこや角かっこ上にカーソルが位置している場合、対応するペアと共にかっこが太字で表示されます (*下のスクリーンショット*)。この機能により、特定の要素の開始点と終了点を簡単に確認することができます。



構文による色分け

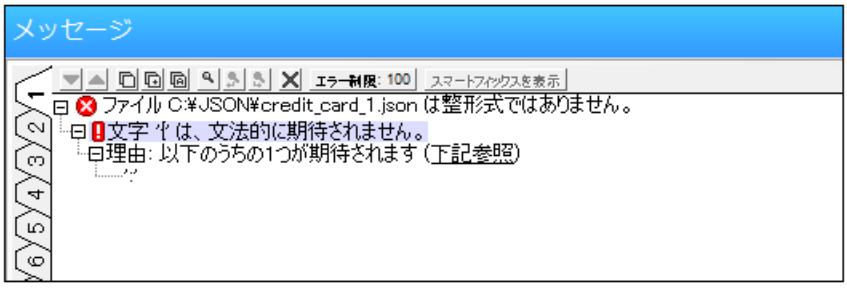
JSON ドキュメント (JSON または Avro *インスタンス*/*スキーマ*)、および JSON5 ドキュメントはオブジェクト文字列、値文字列、演算子、数値、そしてキーワードから構成されます。テキストビューでは、*下のスクリーンショット*にあるように、[オプションダイアログ]の設定に従う形で *上のスクリーンショット*で示されるように各カテゴリーが異なる色で表示されます。オプションダイアログのテキストフォントタブでは、異なる JSON コンポーネントに対する色を指定することができます。左上のコンボボックスから JSON を選択し、各 JSON アイテムに対する目的の色を (*スタイルペイン*) から選択してください。



メモ: JSON ではなく JSON5 構文はコメントの使用を許可しています。JSON5 内のコメントは次のように区切ることができます: `// comment //` or `/* comment */`。

構文のチェック

[XML | 整形式のチェック(F7)] コマンドを選択することで JSON ドキュメント (JSON または Avro インスタンス/スキーマ) の構文チェックを行うことができます。整形式チェックの結果は、以下のスクリーンショットにあるような メッセージウィンドウに表示されます。



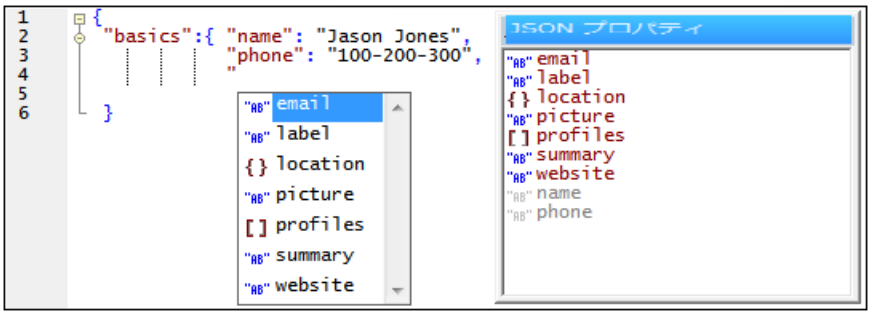
上のスクリーンショットで示されるエラーメッセージでは以下のことが指摘されています：コロンが必要な箇所で中かっこが存在します。

自動補完

自動補完は、編集されるJSON ドキュメント (JSON または Avro インスタンス) がスキーマに関連する場合に有効化されます。

- ドキュメントがJSON の場合、JSON スキーマのためのスキーマに自動的に関連付けられます (JSON のコア/メタスキーマの検証)。そして自動補完はこのスキーマをベースにします。
- ドキュメントがJSON/JSON5 インスタンスの場合、自動補完が有効化されるためには、JSON スキーマがインスタンスに割り当てられていなければなりません。
- ドキュメントがJSON フォーマットのAvro データドキュメントの場合、自動補完が作動するためには、Avro スキーマがインスタンスに割り当てられていなければなりません。
- ドキュメントがAvro スキーマの場合、自動的にAvro スキーマのためのスキーマに関連付けられ、自動補完はこのスキーマをベースにします。

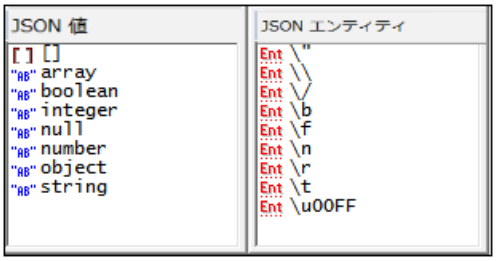
自動補完は、カーソルのロケーションで使用可能な入力オプションを提供します。これは以下により行うことができます (i) メインウィンドウ内のポップアップ (ii) 入力ヘルパーの使用 (下のスクリーンショット参照)。ポップアップと入力ヘルパーはそれぞれ、カーソルのロケーションで有効な入力のリストを表示します。ポップアップリストのエントリを閲覧するには、矢印キーを使用して移動します。スキーマがエントリの詳細を含む場合は、詳細は (スキーマ内のエントリの説明 キーワード) ハイライトされたポップアップのエントリの横に表示されます。ポップアップウィンドウからエントリを選択または入力ヘルパー内のエントリをダブルクリックして挿入します。



上のスクリーンショットに表示されるインスタンスドキュメント内で、ポップアップとJSON プロパティ入力ヘルパーが、プロパティ名を示す引用符の後にカーソルがポイントされると表示されます。入力ヘルパーはこの時点で許可されているすべてのプロパティを表示し、既に入力されているプロパティは灰色で表示され無効化されています。ポップアップはその時点で使用することのできるプロパティを表示します。

他にも2つの入力ヘルパーがあります：JSON 値とJSON エンティティ (下のスクリーンショット)。これらはそれぞれ、文字列

内のエスケープ文字のための許可されている `key:value` ペアとエンティティを表示します。JSON 値入力ヘルパーは、JSON スキーマを編集中に許可されている型キーワードを表示しています。JSON エンティティ入力ヘルパー `\u00FF` 内の最後の入力は、ユニコード文字をあらわすプレースホルダです。青でハイライトされた部分を使用するユニコード文字で置き換えることができます。



他のコンテキストを考慮した自動補完のエントリまたはヒントは、スキーマ内で指定されている場合、以下を含みます: 列挙、詳細、必要とされる発生とデフォルトの値。

10.4 グリッドビュー内の JSON ドキュメント

グリッドビューにより、JSON ドキュメント (JSON インスタンスや JSON スキーマまたは、Avro スキーマ) の構造をグリッドで確認することができ、構成されたテキストのブロックを再構成することができます。これにより、特に長く複雑に構成されたドキュメントの場合、テキストビューで必ず使用できるとは限らない、概要と編集機能が与えられます。JSON ドキュメントの場合、そのような複雑性は、他の配列とオブジェクトで複数のレベルにおよびネストされている配列とオブジェクトのフォームで出現する可能性があります。例えば、(テキストビューに表示される同様) 下にリストされる JSON テキストおよびグリッドビューでの表示を比較する場合 (更に下のスクリーンショットで表示されています)。

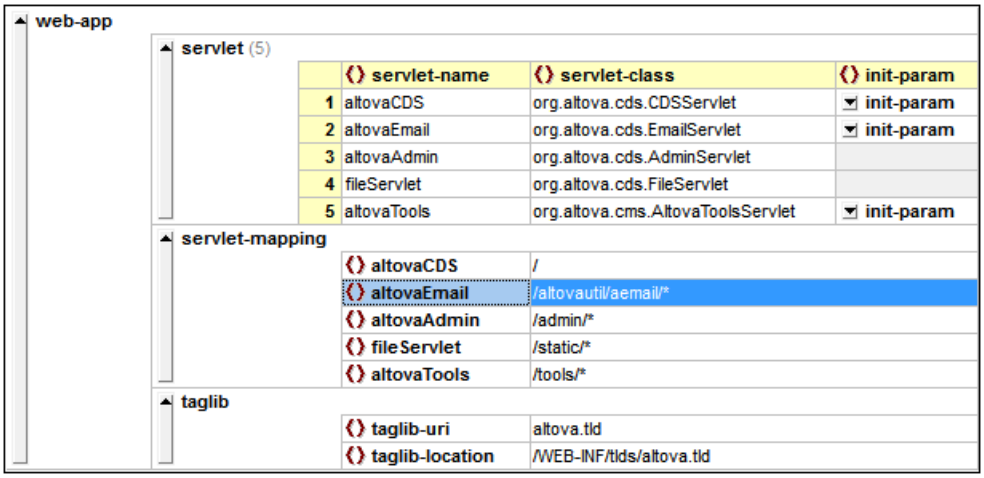
メモ: Avro スキーマは、Enterprise Edition のみでサポートされています。

テキストビュー内にリストされる JSON コード


```
{
  "web-app":{
  "servlet":[
    {
      "servlet-name":"altovaCDS",
      "servlet-class":"org.altova.cds.CDSServlet",
      "init-param":{
      "configGlossary:installationAt":"Philadelphia, PA",
      "configGlossary:adminEmail":"ksm@pobox.com",
      "configGlossary:poweredBy":"Altova",
      "configGlossary:poweredByIcon":"/images/altova.gif",
      "configGlossary:staticPath":"/content/static",
      "templateProcessorClass":"org.altova.WysiwygTemplate",
      "templateLoaderClass":"org.altova.FilesTemplateLoader",
      "templatePath":"templates",
      "templateOverridePath":"",
      "defaultListTemplate":"listTemplate.htm",
      "defaultFileTemplate":"articleTemplate.htm",
      "useJSP":false,
      "jspListTemplate":"listTemplate.jsp",
      "jspFileTemplate":"articleTemplate.jsp",
      "cachePackageTagsTrack":200,
      "cachePackageTagsStore":200,
      "cachePackageTagsRefresh":60,
      "cacheTemplatesTrack":100,
      "cacheTemplatesStore":50,
      "cacheTemplatesRefresh":15,
      "cachePagesTrack":200,
      "cachePagesStore":100,
      "cachePagesRefresh":10,
      "cachePagesDirtyRead":10,
      "searchEngineListTemplate":"forSearchEnginesList.htm",
      "searchEngineFileTemplate":"forSearchEngines.htm",
      "searchEngineRobotsDb":"WEB-INF/robots.db",
      "useDataStore":true,
      "dataStoreClass":"org.altova.SqlDataStore",
      "redirectionClass":"org.altova.SqlRedirection",
      "dataStoreName":"altova",
      "dataStoreDriver":"com.microsoft.jdbc.sqlserver.SQLServerDriver",
      "dataStoreUrl":"jdbc:microsoft:sqlserver://LOCALHOST:1433;DatabaseName=goon",
      "dataStoreUser":"sa",
      "dataStorePassword":"dataStoreTestQuery",
      "dataStoreTestQuery":"SET NOCOUNT ON;select test='test';",
      "dataStoreLogFile":"/usr/local/tomcat/logs/datastore.log",
      "dataStoreInitConns":10,
      "dataStoreMaxConns":100,
      "dataStoreConnUsageLimit":100,
      "dataStoreLogLevel":"debug",
      "maxUrlLength":500
    }
    },{
      "servlet-name":"altovaEmail",
      "servlet-class":"org.altova.cds.EmailServlet",
      "init-param":{
```

```
        "mailHost":"mail1",
        "mailHostOverride":"mail2"
      }
    },{
      "servlet-name":"altovaAdmin",
      "servlet-class":"org.altova.cds.AdminServlet"
    },{
      "servlet-name":"fileServlet",
      "servlet-class":"org.altova.cds.FileServlet"
    },{
      "servlet-name":"altovaTools",
      "servlet-class":"org.altova.cms.AltovaToolsServlet",
      "init-param":{
        "templatePath":"toolstemplates/",
        "log":1,
        "logLocation":"/usr/local/tomcat/logs/AltovaTools.log",
        "logMaxSize":"",
        "dataLog":1,
        "dataLogLocation":"/usr/local/tomcat/logs/dataLog.log",
        "dataLogMaxSize":"",
        "removePageCache":"/content/admin/remove?cache=pages&id=",
        "removeTemplateCache":"/content/admin/remove?cache=templates&id=",
        "fileTransferFolder":"/usr/local/tomcat/webapps/content/fileTransferFolder",
        "lookInContext":1,
        "adminGroupID":4,
        "betaServer":true
      }
    }
  ],
  "servlet-mapping":{
    "altovaCDS":"/",
    "altovaEmail":"/altovautil/aemail/*",
    "altovaAdmin":"/admin/*",
    "fileServlet":"/static/*",
    "altovaTools":"/tools/*"
  },
  "taglib":{
    "taglib-uri":"altova.tld",
    "taglib-location":"/WEB-INF/tlds/altova.tld"
  }
 }
}
```


(*上にリストされる*) テキストビュー内のドキュメント構造を完成することは、時間をかけ慎重に確認する必要があります。(*下のスクリーンショットの*) グリッドビューでの構造は一目で確認することができます。



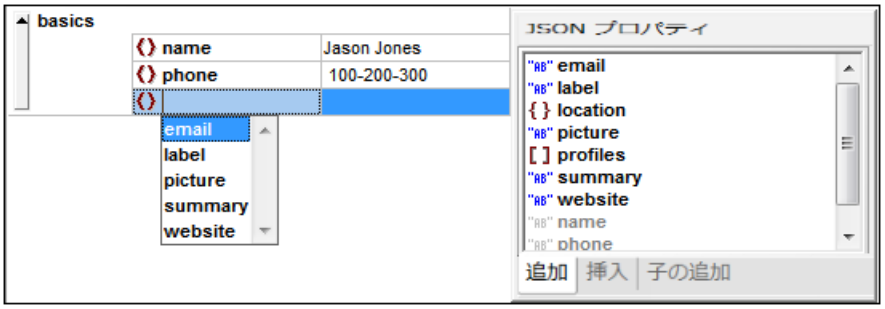
更に、構造は、簡単にオブジェクトを追加、削除、またはグリッド内でオブジェクトを移動させることで簡単に変更することができます。テキストの全体のブロックを(例えば、並べ替えまたは移動して)再編成することができます。コンテンツも同様にグリッドビューで編集することができます。グリッドビュー内で構成されたテキストと作業する方法に関しては、グリッドビューのセクションを参照してください。

自動補完

自動補完は、編集されるJSON ドキュメント (JSON *インスタンス*)スキーマまたはAvro スキーマ)がスキーマに関連する場合に有効化されます。

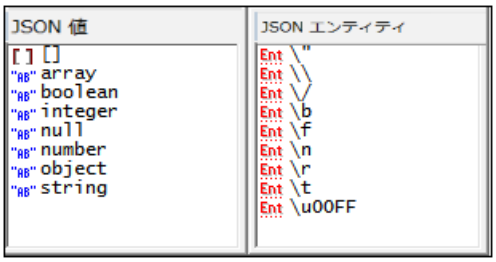
- ドキュメントがJSON の場合、JSON/JSON5 *スキーマのためのスキーマ*に自動的に関連付けられます (JSON の コア*メタスキーマの検証*)。そして自動補完はこの*スキーマ*をベースにします。
- ドキュメントがJSON *インスタンス*の場合、自動補完が有効化されるためには、JSON *スキーマ*が*インスタンス*に割り当てられていなければなりません。
- ドキュメントがJSON フォーマットのAvro データドキュメントの場合、自動補完が作動するためには、Avro *スキーマがインスタンスに割り当てられていなければなりません。*
- ドキュメントが Avro *スキーマ*の場合、自動的にAvro *スキーマのためのスキーマ*に関連付けられ、自動補完はこの*スキーマ*をベースにします。

自動補完は、カーソルのロケーションで使用可能な入力オプションを提供します。これは以下により行うことができます (i) メインウィンドウ内のポップアップ (ii) 入力ヘルパーの使用 (*下のスクリーンショット参照*)。ポップアップと入力ヘルパーはそれぞれ、カーソルのロケーションで有効な入力のリストを表示します。ポップアップリストのエントリを閲覧するには、矢印キーを使用して移動します。ポップアップウィンドウからエントリを選択、または、入力ヘルパー内のエントリをダブルクリックして挿入します。



上の*スクリーンショット*で表示されている*インスタンス*ドキュメントでは、カーソルがプロパティの名前フィールドにポイントされると、ポップアップウィンドウとJSON プロパティ入力ヘルパーが表示されることを示しています。入力ヘルパーは、その時点でのすべてのプロパティを表示しています。既に入力されているプロパティは灰色で表示され無効化されています。ポップアップはその時点で使用することのできるプロパティを表示します。

他にも2つの入力ヘルパーがあります :JSON 値とJSON エンティティ (*下のスクリーンショット*)。これらはそれぞれ、文字列内のエスケープ文字のための許可されている `key:value` ペアとエンティティを表示します。JSON 値入力ヘルパーは、JSON *スキーマ*を編集中に許可されている型キーワードを表示しています。JSON エンティティ入力ヘルパー `\u00FF` 内の最後の入力は、ユニコード文字をあらわすプレースホルダです。青でハイライトされた部分を使用するユニコード文字で置き換えることができます。



他のコンテキストを考慮した自動補完のエントリまたはヒントは、スキーマ内で指定されている場合、以下を含みます: 列挙、詳細、必要とされる発生とデフォルトの値。

10.5 JSON ドキュメントの検証

XMLSpy は、以下を行うことのできるJSON 検証エンジンを搭載しています：

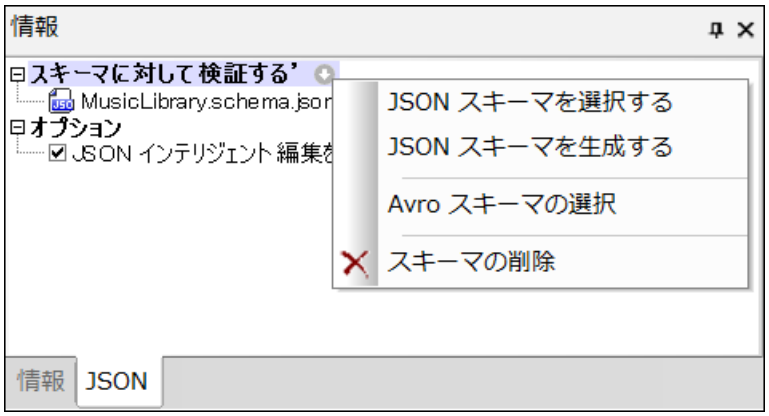
- *JSON スキーマがアクティブなドキュメントの場合*：JSON スキーマをJSON スキーマ仕様に対して検証します。（このため、スキーマの割り当ては必要ありません。）検証は、以下の3種類のビューで行うことができます（テキスト、グリッド、およびJSON スキーマ）。
- *JSON インスタンスがアクティブなドキュメントの場合* JSON インスタンスをJSON スキーマに対して検証します。スキーマは、JSON インスタンスに下で説明されるように割り当てられます。JSON インスタンス検証は、テキストビュー とグリッドビューで実行されます。
- *JSON5 インスタンス がアクティブなドキュメントの場合*：JSON インスタンスをJSON スキーマに対して検証します。JSON5 インスタンスにスキーマが以下に説明されるとおり割り当てられます。JSON5 インスタンス検証はテキストビュー とグリッドビューで実行することができます。

Avro 検証（Enterprise Edition のみ）
Avro データとAvro スキーマ ドキュメントをJSON ドキュメントとして行われる検証は、テキストビュー とグリッドビューで実行されます。

- *JSON フォーマットのAvro データインスタンスが がアクティブなドキュメントの場合*：Avro インスタンスを Avro スキーマに対して検証します。スキーマは下で説明されるようにインスタンスに割り当てられます。
- *Avro スキーマがアクティブなドキュメントの場合*：Avro スキーマをAvro スキーマ仕様 に対して検証します（スキーマの割り当ては必要ありません）；検証は、テキストビュー とグリッドビューで実行されます。

JSON インスタンスにJSON または Avro スキーマを割り当てる

JSON インスタンスをJSON スキーマ、またはAvro スキーマ スキーマに対して検証するには、アクティブなインスタンスドキュメントにスキーマを割り当てる必要があります。アクティブなJSON インスタンスドキュメントの情報ウィンドウ（下のスクリーンショット）。または プロジェクトプロパティダイアログを使用して、割り当てを入力します（オプションつきの検証）。JSON5 インスタンス ドキュメントは、JSON スキーマに対してのみ検証されることに注意してください。



情報ウィンドウのJSON タブで、スキーマに対する*検証*の横のアイコンをクリックします、そして、表示されるメニューから、**「JSON スキーマ、または、Avro スキーマの選択」** をクリックします（*下のスクリーンショット*）。JSON5 インスタンスドキュメントのためには、JSON スキーマオプションのみを有効化することができます。スキーマを参照して、**「OK」**をクリックします。スキーマがアクティブなJSON インスタンスドキュメントに割り当てられ、スキーマのファイル名が情報ウィンドウに入力されます。JSON インスタンスドキュメントが空の場合、JSON またはAvro スキーマのインスタンスへの割り当ては、自動的にJSON インスタンスをスキーマをベースにしたサンプルデータで埋めます。

割り当てを解除するには、同じメニューから **「スキーマの削除」**を選択します（*上のスクリーンショット参照*）。

JSON インスタンスからの JSON スキーマの生成に関する詳細は、JSON インスタンスからの JSON スキーマの生成のセクションを参照してください。

インスタンス および スキーマドキュメントの検証

アクティブな JSON ドキュメント (JSON インスタンス/スキーマ または、Avro スキーマ) を検証するには、コマンド **XML | XML の検証 (F8)** を選択、または、ツールバー内の **検証** (F8) アイコン をクリックしてください。インスタンスドキュメントが検証される場合、スキーマドキュメントがインスタンスに割り当てられる必要があります (上を参照)。検証の結果はメッセージウィンドウに表示されます。

エラーは、番号の付けられている余白内でもフラグが付けられます。エラーのために スマートフィックスを使用することができる場合、電球アイコンがエラーを生成するライン内に表示されます。アイコンにマウスをポイントすると、使用することのできるスマートフィックスのリストのポップアップが表示されます。適用するフィックスを選択します。

メモ: 上記の検証エラーインジケーターとスマートフィックスは **XML | XML の検証 (F8)** コマンドが実行された場合のみ更新されます。これらは背景で更新されます。このため、エラーを修正した後、**検証** (F8) コマンドをもう一度実行し、エラーがフィックスされていることを確認する必要があります。

インスタンスドキュメントからスキーマドキュメントに移動するには、情報ウィンドウ内のスキーマをダブルクリックします (上のスクリーンショットを参照) または、コマンド **DTD/スキーマ | スキーマに移動** を選択します。JSON キーワードのスキーマ定義、または、オブジェクトに直接移動するには、インスタンスドキュメント内のキーワードまたはオブジェクトを選択し、**DTD/スキーマ | 定義に移動** を選択します。

検証 コマンドを使用して、JSON ファイルを含むプロジェクトフォルダーを検証することもできます。

10.6 JSON インスタンスからの JSON スキーマの生成

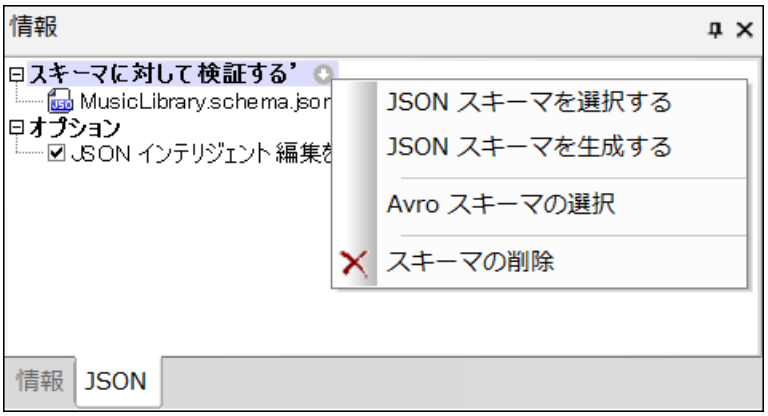
XMLSpy は、(JSON5 インスタンスを含む) JSON インスタンスドキュメントから JSON スキーマを生成することができます。この機能は、既存の JSON インスタンスをベースにしたスキーマを素早く提供し、スキーマをはじめから手動で作成する手間を省くため、とても役に立ちます。必要に応じて作成されたスキーマを修正することもできます。

メモ: Avro スキーマは、Enterprise Edition のみでサポートされています。

スキーマの生成

JSON インスタンスから JSON スキーマを生成するには、以下を行います:

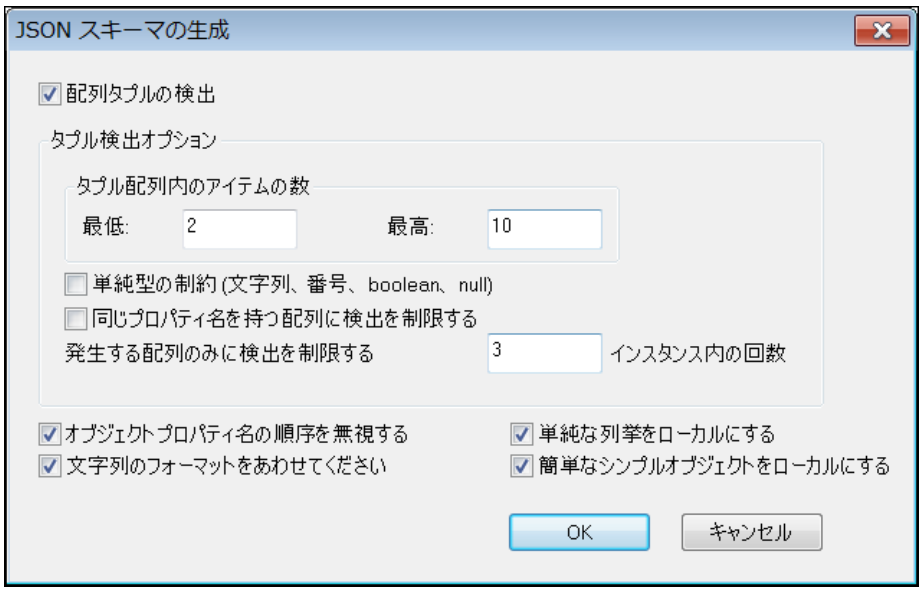
1. JSON インスタンス ドキュメントをアクティブなドキュメントにする。
2. 情報ウィンドウの JSON タブで、スキーマに対して検証するの横の矢印のアイコンをクリックすると、メニューが表示されます。**『JSON スキーマの生成』** をクリックまたは **[DTD/スキーマ| DTD の生成 /スキーマ]** を選択します。



3. 表示された JSON スキーマの生成 ダイアログ (*下の次のセクションを参照*) 内で設定を変更します (*詳細は下を参照*)。編集が完了すると **『OK』** をクリックします。
4. 生成された JSON スキーマのためのパスとファイル名を与えるようにプロンプトされます。**『保存』** をクリックすると、スキーマが生成されアクティブドキュメントになります。インスタンスドキュメント内で、生成されたスキーマファイルは検証で使用されるスキーマとして割り当てられます。以前に割り当てられたスキーマは上書きされます。割り当てを変更するには、コンテキストメニューの **『JSON スキーマの選択』** コマンドを使用します (*上のスクリーンショット参照*)。

スキーマの生成のための設定

JSON スキーマの生成 ダイアログのスキーマの生成を指定することができます (*下のスクリーンショット*)。このダイアログにアクセスする方法の詳細に関しては、前のセクションを参照してください。



配列タプルの検出
配列 *タプル*は*配列内のアイテムのシーケンス*です。例えば、次の配列には3つの*アイテム*を持つ*タプル*があります：`[ 1,2, "abc" ]`。配列の検証のためには、スキーマは*配列 (タプル)*の順序とデータ型を考慮するか指定することができます。*配列タプルの検出* オプションがチェックされていると、*(上のスクリーンショット参照)*、アイテムの順序とデータ型が検出されます。検出された内容に従い、スキーマ内に対応する定義が作成されます。この設定のオプションは以下のとおりです：

- *タプル アイテムの数*：指定することのできる最少および最大の*タプル*アイテムの数。*インスタンス*内の*タプル*がこの範囲内のアイテムの数を持つ場合、配列は検出され定義されます。
- *単純型のみ*：単純型アイテム (アトミック型 `string`、`number`、`integer`、`boolean`、および`null`) を持つ*タプル*のみが検出対象となります。
- *同一の名前を持つ配列*：同じ名前を持つプロパティの値として定義された配列のみが検出対象となります。例えば、下のJSON データフラグメント内では、赤でハイライトされたかっこでマークされた配列は (青でハイライトされた)`a1` という名前のプロパティの値です：`{"object1":[ {"a1":[1,2,"abc" ] }, {"a1":[3,4,"def" ] }, {"a1":[5,6,"ghi" ] } ] }`。
- *配列の最少の数値*：指定することのできる配列の検出の最少の数です。

他の設定

- *オブジェクト プロパティ名前の順序を無視する*：選択されると、プロパティのオブジェクトの順序がチェックされ、元に限りなく近く再作成されます。
- *文字列のフォーマットに一致させる*：文字列データ型を指定するスキーマは特別なフォーマットを持つ必要があります。このオプションが選択されていると、XMLSpy は可能な箇所で文字列フォーマットを検出およびフォーマット定義を追加します。
- *単純配列をローカルに設定する*：単純配列は、同じ単純データ型のすべてのアイテムの1つです。選択されると、すべての単純 配列は、ローカルで参照されているグローバル定義を使用するかわりにスキーマ内でローカルに定義されます。
- *単純オブジェクトをローカルに設定する*：単純オブジェクトは、すべてのプロパティの値が同じ単純データ型をもつものです。選択された場合、ローカルで参照されているグローバル定義を使用するかわりに、すべての単純オブジェクトはスキーマ内で定義されます。

メモ： JSON スキーマが生成されると、個々のオブジェクトと配列の定義のローカル定義をグローバルにまたその反対に設定することができます。詳細に関しては、グローバルとローカル定義のセクションを参照してください。

10.7 JSON インスタンスを JSON スキーマから生成する

テキストビュー、グリッドビューまたはスキーマビューで JSON スキーマがアクティブなファイルの場合、JSON スキーマから JSON インスタンスを生成することができます。 **「DTD/スキーマ」 サンプル XML/JSON ファイルを生成する」** をクリックします。このコマンドは、JSON5 ドキュメントではなく JSON ドキュメントを生成することに注意してください。

10.8 JSON と XML の間の変換

以下の変換を行うことができます：

- XML インスタンスを JSON に変換：XML インスタンスドキュメントがアクティブなドキュメントの場合、JSON または JSON5 インスタンスドキュメントの生成を選択することができます。コマンド **[変換 | XML インスタンスを JSON へから変換]** を使用します。
- JSON インスタンスを XML に変換：JSON/JSON5 インスタンスドキュメントがアクティブなドキュメントの場合、XML インスタンスドキュメントは JSON インスタンスから **[変換 | XML インスタンスを JSON へから変換]** をクリックすることで生成されます。
- XML スキーマを JSON スキーマに変換：XML スキーマドキュメントがアクティブなドキュメントの場合、JSON スキーマドキュメントは XML スキーマから **[変換 | XML スキーマを JSON スキーマへから変換]** をクリックすることで生成されます。
- JSON スキーマを XML スキーマに変換：JSON スキーマドキュメントがアクティブなドキュメントの場合、XML スキーマドキュメントは JSON スキーマから **[変換 | XML スキーマを JSON スキーマへから変換]** をクリックすることで生成されます。

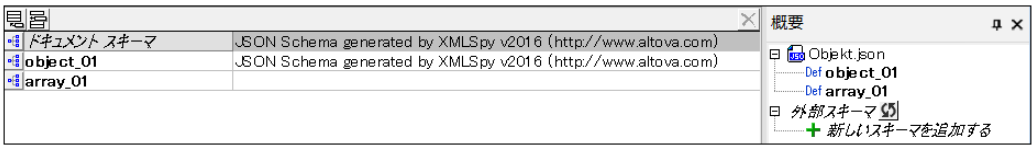
これらの変換はテキストビューとグリッドビューで有効化されます。上のリンクをクリックするとそれぞれの機能を確認することができます。

10.9 JSON スキーマビュー

JSON スキーマビューは、JSON スキーマドキュメントを閲覧し編集するために使用することができます。JSON スキーマビュー ウィンドウの主な部分は以下のとおりです：

- **定義概要グリッド** と **デザインビュー** 間を切り替えるメイン ウィンドウ
- 3 つの入力ヘルパーウィンドウ (メインウィンドウの右側にデフォルトで表示されます):概要、詳細、制約
- メッセージウィンドウ (メインウィンドウの下にデフォルトで表示されます)
- 情報ウィンドウ (アプリケーションウィンドウの左側にデフォルトで表示されます)

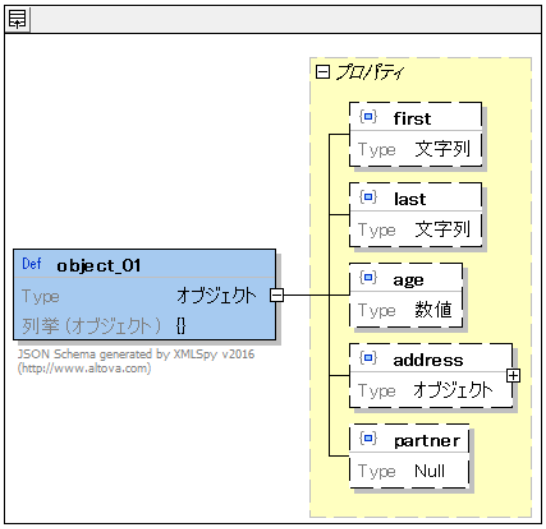
下のスクリーンショットは、メインウィンドウおよび概要入力ヘルパーを表示しています。



メインウィンドウ

メインウィンドウは、定義概要グリッド (*上のスクリーンショットに表示*) とデザインビュー (*下のスクリーンショット*) を切り替えます。定義概要グリッドは、現在のドキュメントの ("*ドキュメントスキーマ*"として)リストされている) メインスキーマを表示し、またスキーマに追加する定義を表示します。 (定義は、JSON データ構造の詳細です。上のスクリーンショットでは、`object_01` および `array_01` がそれぞれオブジェクトと配列の定義です。)定義は、概要入力ヘルパーにもリストされています (*上のスクリーンショット参照*)。

定義概要グリッドは、JSON スキーマのハイレベルな表示を行いますが、概要内にリストされている定義の内容を表示しません。デザインビュー内の定義を確認して編集するには (*下のスクリーンショット*)、定義のアイコン (*上のスクリーンショット参照*) をクリックするか、または概要入力ヘルパー内の定義をダブルクリックします (*上のスクリーンショット参照*)。



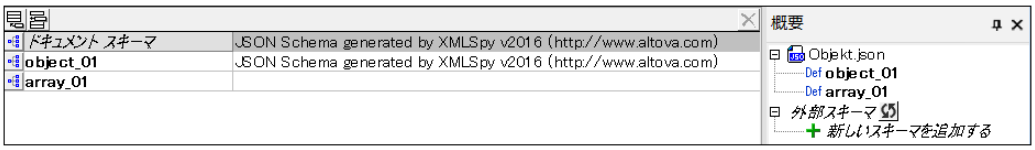
デザインビューから定義概要グリッドに切り替えるには、デザインビューの左上にある**[定義グリッドに切り替え]** アイコンをクリックします (*上のスクリーンショット参照*)。デザインビューを構成するには、メニューコマンド **スキーマデザイン | 構成ビュー** をクリックします。

入力ヘルパー

スキーマビューの2つのモード (定義概要グリッド とデザインビュー) には、3つの入力ヘルパーがあります :概要、詳細、制約。これらの入力ヘルパーは以下のためのメカニズムが搭載されています : (i) スキーマとその定義に関する情報を表示 (ii) 定義に関連する情報と値の入力。詳細に関しては、入力ヘルパー :概要、詳細、制約を参照してください。

10.9.1 グローバル定義の追加

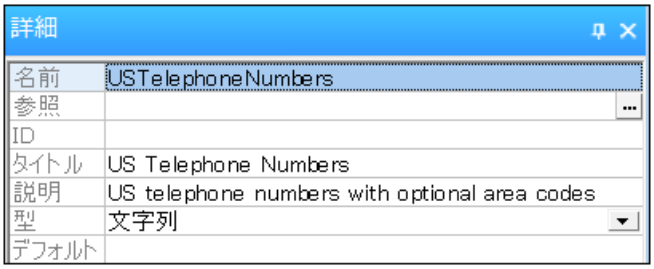
メインウィンドウ内の**定義概要グリッド** (*下のスクリーンショット*) は、スキーマのグローバル定義を表示します。これらのグローバル定義は、以下のとおりです : (i) **メインドキュメントスキーマ** 定義、(ii) オブジェクト、配列、文字列などの、**グローバルな JSON 型** の定義。グローバルに JSON 型を定義することは、型が同じスキーマまたは他のスキーマで再利用する必要がある際に役に立ちます。例えば、米国の電話番号のための JSON 文字列型を1つの JSON スキーマで定義することができます。そして、この定義を同じスキーマ内および他の JSON スキーマから参照します。



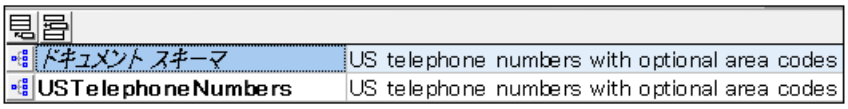
定義と関連したアクションの追加

定義概要グリッド内で定義を追加および編集するために次のアクションを使用することができます。

- *定義を追加する*:定義概要グリッドの左上の**[名前を付けられたスキーマ定義を追加]** または **[名前を付けられたスキーマ定義を挿入]** をクリックします (*上のスクリーンショットを参照*)。新しい空の定義がグリッド内に作成されます。新規の定義は、概要入力ヘルパーにも *Def* としてリストされます (*上のスクリーンショットを参照*)。
- *定義の型の変更*:`Any` の型の新しい定義が作成されます。詳細入力ヘルパーで型を変更することができます (*下のスクリーンショットを参照。型が 'String' である箇所*) または、デザインビュー内の定義を編集。



- *定義の名前の変更*:名前をダブルクリックして編集します。 または、詳細入力ヘルパー内の *名前* フィールドを編集します。
- *定義の説明の入力*:詳細入力ヘルパー内の *詳細* フィールドを編集します。定義の名前の横の定義概要グリッド内に詳細が表示されます (*下のスクリーンショットを参照*)。詳細を編集するために定義概要グリッドの詳細フィールドをダブルクリックすることもできます。



- *定義の参照*:詳細に関しては、概要入力ヘルパーの説明とトピックグローバルとローカル定義を参照してください。
- *定義の編集*:定義概要グリッド内の定義アイコンをクリック または概要入力ヘルパーの定義をダブルクリックします。これにより定義を編集することのできるデザインビュー内の定義が開かれます。

10.9.2 入力ヘルパー :概要、詳細、制約

JSON スキーマビュー入力ヘルパーは、デフォルトで、アプリケーションウィンドウの右側に表示されます。メインウィンドウの両方のモードで使用することができます :(i) 定義概要グリッド、(ii) デザインビュー。入力ヘルパーウィンドウのタイトルバーをドラッグしてスクリーンの他の場所に移動することができます。入力ヘルパーのタイトルバーを代わりにダブルクリックして、入力ヘルパーをドッキングまたはドッキング解除することができます。詳細に関しては、入力ヘルパーのセクションを参照してください。

概要入力ヘルパー

概要入力ヘルパー (*下のスクリーンショット*) は、現在のスキーマの定義および現在のスキーマのすべてのグローバル定義をリストします。定義をダブルクリックすることにより、デザインビュー内でその定義を開き編集することができます。ことができます。外部スキーマからの定義を使用するには、最初に外部スキーマを追加して、定義を再利用します。

外部スキーマの追加
概要入力ヘルパー内の **新しいスキーマの追加** アイコンクリックして、追加するスキーマを参照し、外部スキーマを追加します。スキーマが追加されると、スキーマが追加されると、その定義は概要入力ヘルパーに表示されます。下のスクリーンショットは、例えば、スキーマ `TelNumbers.json` が追加されたこと、また、このスキーマには、`USTelephoneNumbers` という名前の1つの定義があること表示しています。外部スキーマは任意の数追加することができます。

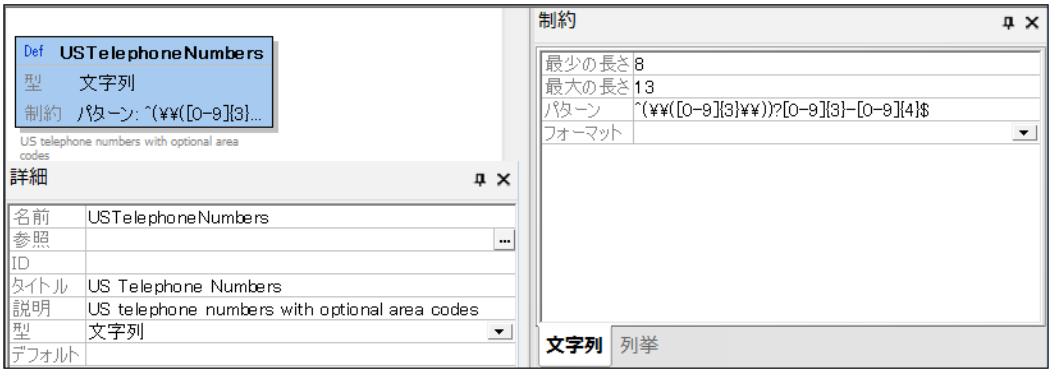
外部定義の再利用
外部スキーマが追加された後、インポートされるスキーマの定義内で再利用することができます。定義の1つが他の定義を参照して使用する場合、その定義のプロパティを取ります。参照は以下の2つの方法で行うことができます：

- *デザインビュー内*：概要入力ヘルパーから定義を定義が使用される箇所にドラッグします。
- *定義概要グリッド または デザインビュー内*：他の定義を再利用する定義の詳細入力ヘルパーの参照フィールドを使用して。この点に関しては、詳細入力ヘルパーで説明されています。

メモ： T概要ウィンドウ内の外部スキーマの横の **更新** アイコンは、追加されたすべての外部スキーマを更新します。追加された外部スキーマから定義が再利用されない場合、そのスキーマはリストが更新されると削除されます。

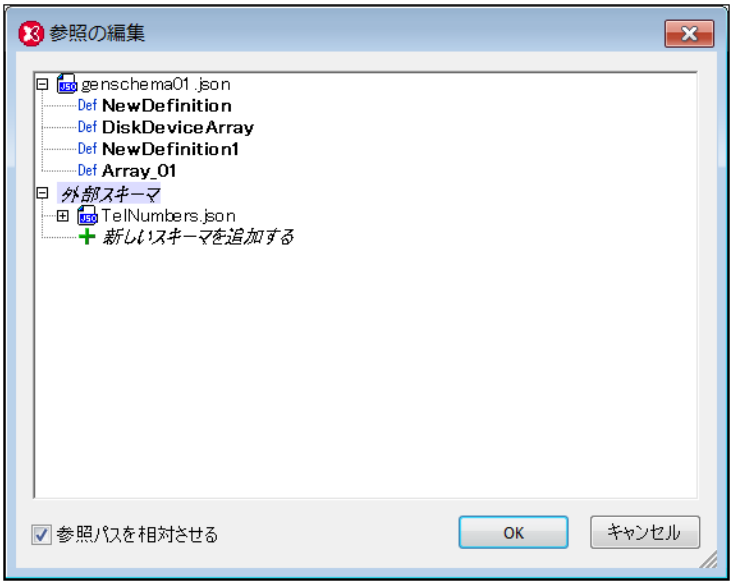
詳細入力ヘルパー

メインウィンドウの2つのモード（定義概要グリッド または デザインビュー）定義が選択されると、定義のプロパティは詳細および制約ウィンドウから入力することができます。下のスクリーンショットは、詳細と制約入力ヘルパーと共にデザインビュー内の `USTelephoneNumbers` の定義を表示しています。2つの入力ヘルパー内の情報はデザインビューの定義（青）ボックスに表示されていることに注意してください。2つの入力ヘルパーに設定することのできるプロパティは下にリストされています。



詳細入力ヘルパーに次の詳細を入力することができます：

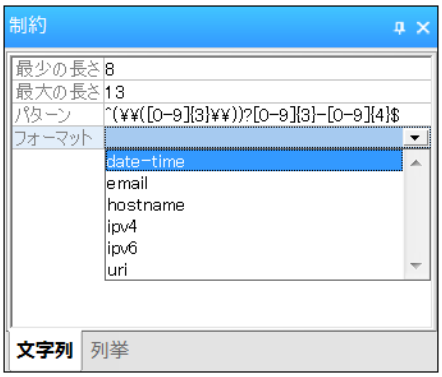
- *名前* 定義の名前．
- *参照* :定義で他の定義を再利用する場合、参照フィールドの **[追加ダイアログ]** ボタンをクリックします。これにより使用可能なすべての現在のスキーマおよび外部スキーマからの定義をリストする参照の編集ダイアログが表示されます（*下のスクリーンショット*）。再利用する定義を選択して、相対パスを使用する場合、相対パスのオプションを選択して、 **「OK」** をクリックします。



- **ID**: これはオプションのキーワードで、親スキーマ内のサブスキーマとして考えられる現在の定義のソリューションスコープを変更するために使用されます。ID 値は、URI である文字列である必要があります。Altova JSON バリデーターは、正規の逆参照のみを使用することに注意してください。詳細に関しては、JSON 仕様を参照してください。
- **タイトル、詳細**: これら2つのキーワードの値は詳細の目的で使用されます。
- **デフォルト**:定義のデフォルトの値。
- **型**: コンボボックスのドロップダウンリストから定義のデータ型を選択してください。型の変更は前の型特有のキーワードを削除することに注意してください。前の定義に戻る場合は、**[元に戻す] (Ctrl-Z)** を押してください。、

制約入力ヘルパー

定義の制約は、型により異なります。各型の制約は、以下に説明されています。



型が下のリストに表示されない場合、制約を定義することはできません。しかしながら、すべての型に対して列挙を定義することはできます：

- *文字列*：文字列の長さと文字列のパターン。パターンは正規表現を使用して指定することができます。フォーマットフィールドから仕様内で定義されているフォーマットの1つを選択することができます (*上のスクリーンショットを参照*)
- *数値* 許可されている数値の範囲
- *配列* 許可されている配列内のアイテムの数
- *オブジェクト* 許可されているプロパティの数

すべての型のための制約入力ヘルパーには列挙タブがあります。その定義の型で許可されているアイテムのリストを指定することができます。

10.9.3 グローバルとローカル定義

JSON スキーマ定義 グローバルまたはローカルで設定することができます。

- **グローバル定義** は、定義を追加し、プロパティを指定することによりメインウィンドウの定義概要グリッド内で作成されます。グローバル定義は、同じスキーマまたは他のスキーマ内の定義により参照されることができます。これによりプロジェクト全体での定義の再利用が有効化されます。現在のスキーマのすべてのグローバル定義は、スキーマの定義概要グリッドに表示されています。他のスキーマからのグローバル定義は、概要入力ヘルパー内の外部スキーマの追加 により再利用のために使用可能になります。
- **ローカル定義** は、グローバル定義に子孫 または 兄弟定義 (プロパティまたはアイテム) を追加することにより、グローバル定義内で作成されます。

グローバル定義の再利用

グローバル定義を再利用するには、以下の1つを行います：

- デザインビューで、グローバル定義を概要入力ヘルパーからドラッグして使用する定義の上にドロップします。
- デザインビュー内の定義概要グリッドから、再利用する定義を選択します。詳細入力ヘルパーの参照フィールド内で、再利用するグローバル定義を選択します。詳細に関しては、詳細入力ヘルパー を参照してください。

メモ: 同じスキーマ内の他の定義で参照された後に、グローバル定義の名前を変更する場合、参照内で名前も変更されます。他のスキーマからの参照は、しかしながら、名前変更を反映するために手動で編集される必要があります。

ローカル定義 を グローバル定義に変換する
ローカル定義を変換するには、デザインビュー内で定義を右クリックし、**[グローバルにする]** を選択します。グローバル定義が作成され、ローカル定義上に参照が作成されます。グローバル定義の名前は、自動的に生成されるため、変更するとその変更はローカル定義の参照にパスされます。、

グローバル定義の参照をローカル定義に変更する
グローバル定義への参照は、ローカルおよびグローバル定義上に存在することができます。参照を削除して、プロパティをローカルにするには、デザインビュー内の (ローカルまたはグローバル) 定義を右クリックして、**[ローカルにする]** を選択します。グローバル定義の プロパティは、定義上にローカルで作成されます。

10.9.4 デザインビュー

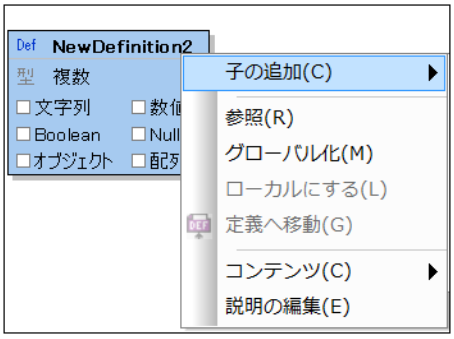
デザインビューでは、構造および許可されているグローバル定義の値を指定することができます。定義は次の GUI コンポーネント またはメカニズムを使用して指定することができます :

- 詳細入力ヘルパー (定義概要グリッド内で使用することもできます)
- 入力ヘルパーの制約 (定義概要グリッド内で使用することもできます)
- 定義のコンテキストメニュー (メインウィンドウの定義ボックスを右クリックしてアクセスすることができます)

詳細と入力ヘルパーの制約により指定することのできる定義は、入力ヘルパー :概要、詳細、制約のセクションで説明されています。これらのプロパティの一部は、メインウィンドウの定義ボックスで指定することもできます。 このセクションでは、次の 3 つのセクションで、メインウィンドウで使用することのできるグラフィカルなメカニズムについて説明を行います。

メモ: 誤った操作および希望しない変更を元に戻すには、**[Ctrl+Z]** を押してください。

コンテキストメニュー
(*下のスクリーンショットの青のボックス*) のコンテキストメニューにより、定義の構造のデザイン、およびプロパティの編集を有効化することができます。



以下のコマンドを使用することができます :

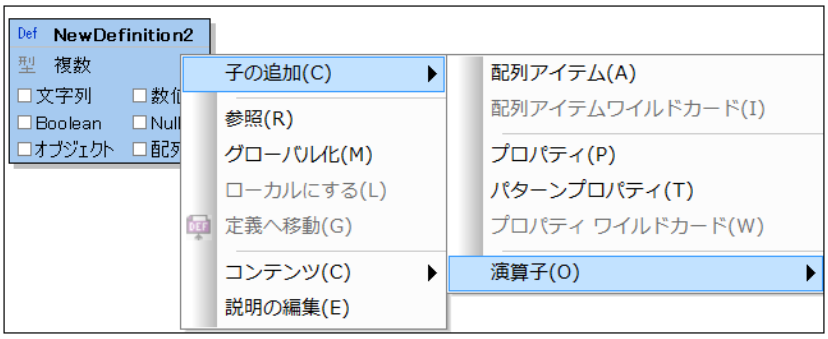
- **子の追加**: 定義の型により追加することのできる子が決定されます (下の子の追加 :構造の作成 を参照してください)。
- **参照**: 定義がグローバル定義を参照することを有効化し、そのグローバル定義のプロパティを取ります。コマンドが

開く参照の編集ダイアログは、詳細入力ヘルパーを使用してアクセスすることのできるダイアログと同じです。詳細は入力ヘルパー:概要、詳細、制約を参照してください。

- **グローバルに設定**: このコマンドは、ローカル定義が定義の場合に有効化されます。現在選択されている定義をグローバル定義とし、現在の選択にそのグローバル定義の参照を追加します。 I
- **ローカルに設定**: このコマンドは、グローバル定義が定義の場合に有効化されます。現在選択されているローカル定義を元のグローバル定義への参照を作成して変換します。
- **定義に移動**: 選択された定義がグローバル定義を参照する定義に含まれる場合、このコマンドは有効化されます。クリックするとグローバル定義に移動します。
- **コンテンツ**: **コンテンツ** コマンドは、選択された定義のコンテンツを切り取り、コピーおよびペーストするを含むサブメニューを表示します。
- **詳細の編集**: 編集する定義の詳細フィールドを有効化します。

子の追加 :構造の作成

定義の構造は、複数のレベルの子孫を追加することにより作成されます。これらのレベルは、コンテキストメニューの**子の追加** コマンドにより作成されます。定義に追加する子はその型により異なります :



- *オブジェクト*: プロパティと演算子を取ります
- *配列*: 配列 アイテムと演算子を取ります
- *アトミック型 (文字列、数値、boolean、null)*: 演算子を取ります
- *Any*: プロパティ、配列 アイテム、 と演算子を取ります
- *Multiple*: 含まれる型によります。選択された方のための許可された子の和集合を取ります。
- *演算子*: 論理演算子を構造を決定するために使用することを有効化します

各型のために作成される構造は、上のリストにリンクされているセクションで説明されています。

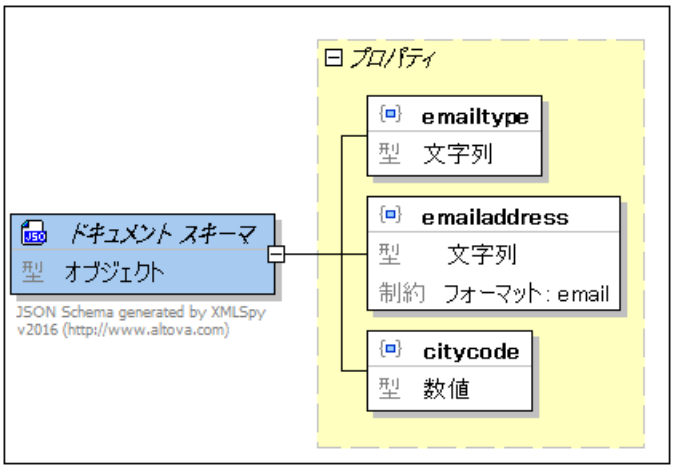
10.9.5 オブジェクトとプロパティ

オブジェクトは、中かっこで区切られており、値にキーを以下のようにマップします :`"MyKey": Value`。キーは常に文字列である必要があり、そのために引用符で囲まれている必要があります。値は、JSON データ型任意のであることができます。各 `key:value` ペアはオブジェクトのプロパティとして知られています (*下のスクリーンショット参照*)。

以下は、 3 つの プロパティを持つインスタンス化されたオブジェクトの例です :

```
{
  "emailtype":"home",
  "emailaddress":"contact01home@altova.com",
  "citycode":22
}
```

オブジェクトのためのスキーマは、デザインビューで以下のように表示されます。



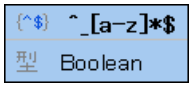
以下の点に注意してください：

- それぞれのプロパティは、インスタンス内に存在する必要があります。これは、プロパティの境界により示されています。プロパティがオプションである場合、境界は点線で表示されています。プロパティが必要であるかまたはオプションであるかをプロパティのコンテキストメニューまたは詳細入力ヘルパーで設定することができます。
- プロパティがインスタンス内で発生する順序は、スキーマ内で定義することはできません。これは、スキーマ内で定義されるプロパティの順序は、無関係であるからです。
- かっこ内の青い角のシンボルは、(他のシンボルにより示されているプロパティワイルドカードまたは プロパティワイルドカードに対して)プロパティを示しています。*下を参照してください。*.
- プロパティの型は、ダイアグラム内の型をダブルクリックして編集することができ、表示されるドロップダウンリストからオプションを選択することができます。または、型は詳細入力ヘルパー内で選択することができます。
- `emailaddress` プロパティの制約の値は、入力ヘルパーの制約で定義することができます。

プロパティ、パターンプロパティ、およびプロパティワイルドカード

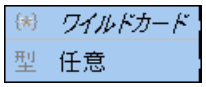
オブジェクトは、プロパティ、パターンプロパティ、および プロパティワイルドカードを持つことができます。これらは、オブジェクトコンテキストメニューを使用して追加することができます：(i) オブジェクト、(ii) 黄色いプロパティボックス（ボックスのプロパティタイトルを右クリック）、(iii) それぞれのプロパティ。プロパティは上で説明されています。ここでは、パターンプロパティとプロパティワイルドカードについて説明をおこないます。

パターンプロパティ(*下のスクリーンショット*) は、プロパティの名前を正規表現として定義します。下のスクリーンショットでは、例えば、正規表現は以下を指定します：(i) プロパティがアンダースコアで始まる名前を有する、そして、 (ii) ブール値を値として有する。ワイルドカードのための制約条件はありません。パターンプロパティは任意の数追加することができます。パターンプロパティのためのアイコンに注意してください。



プロパティワイルドカード(*下のスクリーンショット*) は、オブジェクトのプロパティセットの他のプロパティに加え発生するプロパティを指定します。ワイルドカードはしかしながら、これらの発生のための型を定義します。下のスクリーンショットは、数値以外の値を持つ名前を持つプロパティを定義するプロパティワイルドカードを表示しています。各オブジェクトにプロパティワイルドカードは 1つのみ存在

することができます。ワイルドカードが *Any* 型、に設定されている場合は、入力ヘルパーの制約内のそれぞれの方の制約を設定することができます。プロパティワイルドカードのためのアイコンに注意してください。



メモ: プロパティワイルドカード または プロパティワイルドカードの最小および最多の発生設定はありません。更に深い理解のために[プロパティの検証](#)を参照してください。

プロパティの検証方法

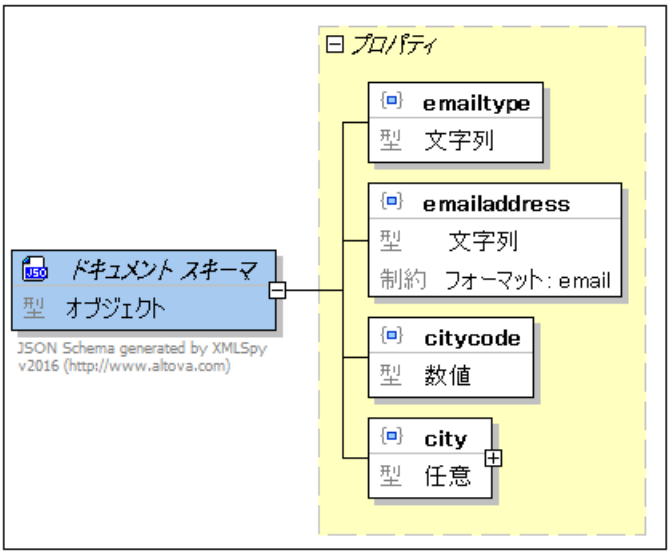
プロパティがインスタンスに遭遇した場合、以下の方法で検証することができます：

1. スキーマ内のプロパティの名前は、オブジェクトのすべての名前付けられたプロパティに対してチェックされます。
2. 一致が検出されない場合、名前はオブジェクトのプロパティセット内のすべてのパターンプロパティに対してチェックされます。
3. まだ一致が検出されない場合、存在する場合はワイルドカードが呼び出されます。
4. まだ名前との一致が検出されない場合、有効性エラーが報告されます。プロパティまたは パターンプロパティに一致する名前、 またはワイルドカードが存在する場合、値がプロパティ定義に対応する値に対してチェックされます。
5. インスタンス値が対応するプロパティ定義の型と制約に一致する場合、プロパティは有効です。それ以外の場合は無効です。

サンプル

下のスクリーンショットは、以下の条件を持つオブジェクトを定義しています：

- `emailtype`、`emailaddress`、および`citycode` という名前を持つ 3 つの プロパティが必要です。
- アンダースコアで始まる名前を持ち、 ブール値を持つ1つ または複数のプロパティを持つことができます (下のスクリーンショットのパターンプロパティを参照)。
- 任意の名前と値を持つ1つ または複数の追加プロパティを持つことができます。

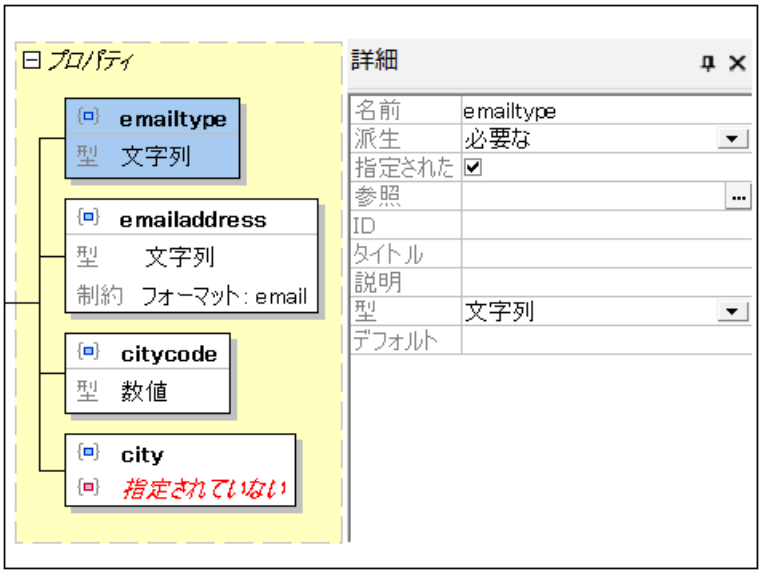


10.9.6 指定されていないプロパティ

下のコードのリストでは、`required` キーワードは、4つのプロパティがこのオブジェクトのために必要なことを指定しています。しかしながら、4つのプロパティの3つは定義されています。4番目のプロパティ`city` は定義されていません。定義されているプロパティは指定されなければならず、定義されていないプロパティは、指定されていません。リストの下のスクリーンショットを参照してください。

コードのリスト:指定されているおよび指定されていないプロパティ

```
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   "description": "JSON Schema generated by XMLSpy v2016 (http://www.altova.com)",
   "type": "object",
   "properties": {
      "emailtype": {
         "type": "string"
      },
      "emailaddress": {
         "type": "string",
         "format": "email"
      },
      "citycode": {
         "type": "number"
      }
   },
   "required": [
      "emailtype",
      "emailaddress",
      "citycode",
      "city"
   ],
   "additionalproperties": false
}
```



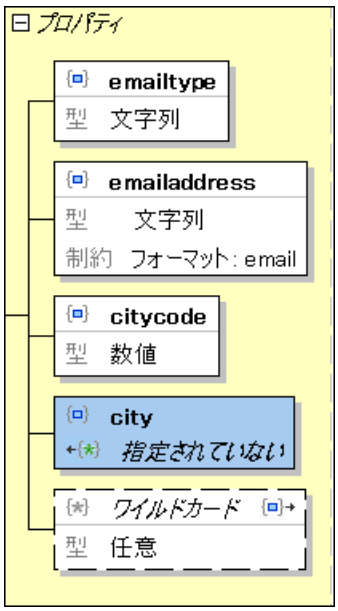
デザインビュー内では、スキーマで必要とされているにもかかわらず定義されていないため、指定されていないプロパティは赤でフラグ付けされています。JSON スキーマ自身は、有効ですが、検証に使用されるインスタントドキュメントが有効ではありません。これには以下の理由があります：() `city` プロパティが存在しない場合、`city` プロパティが必要なため、ドキュメントは無効になります。`city` プロパティが存在する場合、 `city` プロパティが定義されておらず、このプロパティのためのプロパティワイルドカードが許可されていないため、ドキュメントは無効になります（下の*暗黙的にプロパティを指定する*を参照）。

指定されていないプロパティのための定義を作成するには、以下を行います：

1. デザインビュー内の指定されていないプロパティを選択します。
2. 詳細入力ヘルパー内の指定されたチェックボックスをチェックします（*上のスクリーンショット参照*）。または、指定されているフラグをコンテキストメニューを使用して削除します。
3. 必要に応じてプロパティの定義を変更します。

暗黙的にプロパティを指定する

プロパティは、適切なパターンプロパティまたはプロパティワイルドカードを追加することにより暗黙的に指定することができます。下のスクリーンショットは、追加されたプロパティワイルドカードを表示しています。 `city` と名前の付けられたインスタンスプロパティは、このワイルドカードに一致します。スキーマ内では、ですから、 `city` プロパティは、ワイルドカードにより暗黙的に指定されたということができます。`city` プロパティを含むインスタンスファイルはこのスキーマに対して有効になります。



暗黙的に指定されているプロパティ内とプロパティワイルドカード内のそれぞれのアイコンに注意してください。各アイコンは他のプロパティへのリンクです。アイコンをダブルクリックするとその他のプロパティを選択します。

10.9.7 オブジェクトと依存性

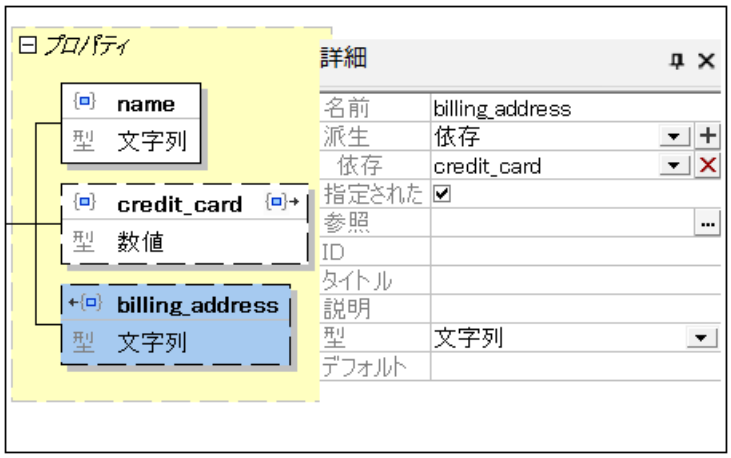
オブジェクトの定義では、特定のプロパティが他のプロパティが存在する場合のみ存在するように指定する必要がある場合があります。最初のプロパティは、2番目のプロパティの依存性となります。このセクションでは依存性を含むシナリオについて説明を行います。オブジェクト (例えば `member`) には `credit_card` という名前のプロパティがあるとします。オブジェクトの `billing_address` プロパティは、`credit_card` プロパティの依存性に設定することができます。これは、`credit_card` プロパティが存在する場合のみ、`billing_address` プロパティが存在することを意味します。

この種類の依存性は2通りの方法で指定することができます：

- プロパティ依存性として (依存性構造はプロパティです)
- スキーマ依存性として (依存性構造はスキーマです)

プロパティ依存性

下のスクリーンショットは、`name` プロパティ (必須)、`credit_card` プロパティ (任意)、および`billing_address` プロパティ (依存性) を持つオブジェクトを表示しています。`billing_address` プロパティは、依存性 `credit_card` プロパティにあります。JSON オブジェクト定義のコードは、下のスクリーンショットにリストされています。また、プロパティ依存性の作成方法は下に説明されています。



プロパティ依存性を持つJSON オブジェクトのコードのリスト

```
{
  "type":"object",
  "properties":{
    "name":{
    "type":"string"
    },
    "credit_card":{
      "type":"number"
    },
    "billing_address":{
      "type":"string"
    }
  },
  "required":["name"],
  "dependencies":{
    "credit_card":["billing_address"]
  },
  "additionalProperties":false
}
```

プロパティ依存性を作成するには、以下を行います：

1. 依存性がベースにされるプロパティを右クリックします。 (サンプルでは、`credit_card` プロパティです。)
2. 表示されるコンテキストメニューから**[依存性の追加 | 依存性プロパティ]** を選択します。新しいプロパティが依存性の値の発生に追加されます。
3. このプロパティの名前と値を定義し、追加の詳細または制約を追加します。

他の プロパティの依存性としてプロパティを指定するには、以下を行います：

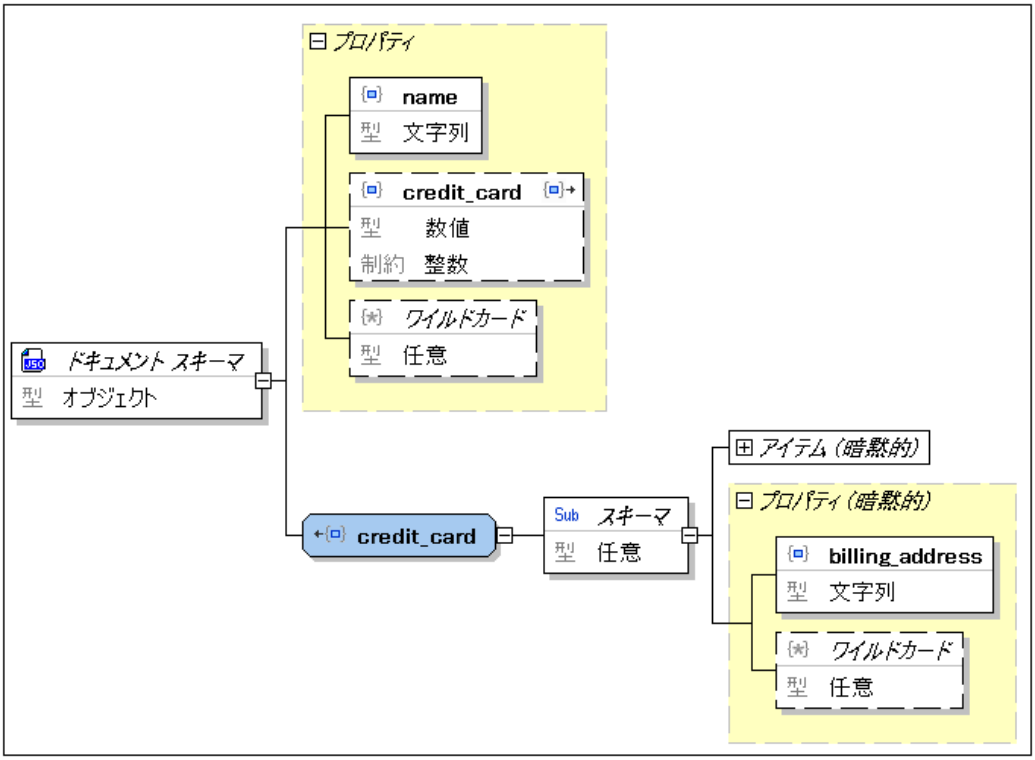
1. 他のプロパティの依存性とするプロパティを右クリックします。(サンプルでは、`billing_address` プロパティ.)
2. 表示されるコンテキストメニューから、**[依存性]** を選択します。または、詳細入力ヘルパーで、出現の入力に移動して、*依存性* を選択します (*上のスクリーンショット参照*)。
3. 詳細入力ヘルパーで、入力の依存性のドロップダウンリストのアイコンをクリックします。ドロップダウンリストはオブジェクトのすべての他のプロパティを表示します。現在のプロパティが依存するプロパティを選択します。

メモ: 依存性に含まれる両方のプロパティのボックスが表示されますアイコン (*上のスクリーンショット参照*)。プロパティのアイコンをダブルクリックすると他のプロパティに移動します。

メモ: プロパティは、複数の依存性 プロパティを持つことができます。

スキーマ依存性

下のスクリーンショットは、前のセクションで同じインスタンスデータ構造をオブジェクトとして説明するオブジェクトについて説明しています。を表示します。2つのオブジェクトの定義は、しかしながら、異なります。前の定義は`billing_address` プロパティを`credit_card` プロパティ上の依存性と定義するために*プロパティ依存性*を使用していますが、現在の定義は、*スキーマ依存性*をこの依存性を定義するために使用しています。後者のJSON オブジェクト定義のコードは下のスクリーンショットにリストされています。どのようにスキーマ依存性を作成するかは下に説明されています。



- *スキーマ依存性を持つJSON オブジェクトのコードリスト*

スキーマ依存性を作成するには、以下を行います:

1. 依存性がベースにするプロパティを右クリックします (サンプル内では`credit_card` プロパティです)。
2. 表示されるコンテキストメニューから**[依存性の追加 | スキーマ依存性]** を選択します。新しいオブジェクト定義が作成されます。オブジェクトは、依存性のプロパティと同じ名前 (サンプル内では`credit_card`) を持ち、子サブスキーマを持ちます。
3. サブスキーマを定義して、追加詳細または制約を追加します。

メモ: 依存性に関連するプロパティおよびオブジェクトのボックス内にアイコンが表示されます (*上のスクリーンショット参照*)。

ボックス内のアイコンをダブルクリックすると他のボックスに移動することができます。

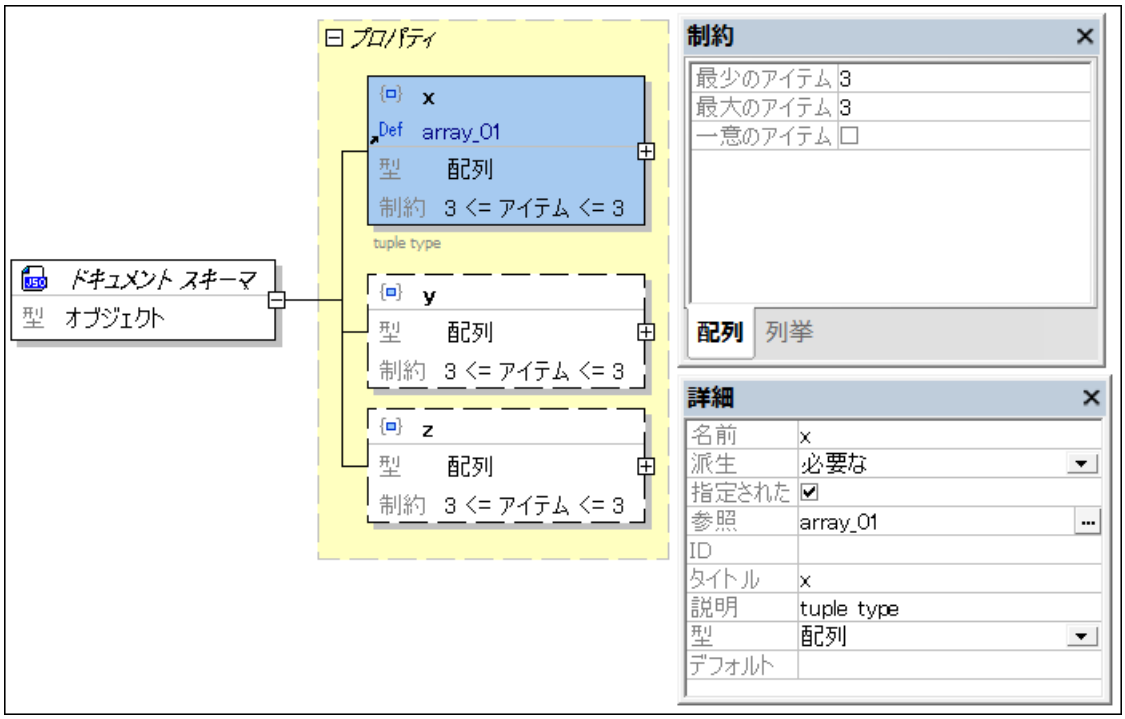
メモ: 複数の依存性を設定するには、依存性 サブスキーマ内で行います。 (*上のスクリーンショット参照*)。

10.9.8 配列

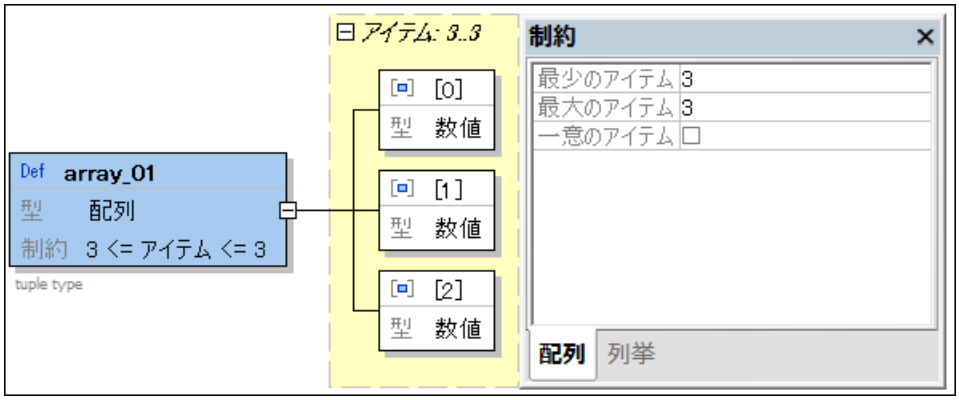
配列は、順序付けされたゼロまたは複数のアイテムで、角かっこにより区切られています。リスト内の各アイテムには、型が割り当てられています。下のインスタンスのリストは、3つのプロパティを持つオブジェクトです。各プロパティの値は黄色のハイライトで区切られている配列です。

```
{
    "x":[ 1,2,"abc" ],
    "y":[ 3,4,"def" ],
    "z":[ 5,6,"ghi" ]
}
```

うえのリスト内の3つすべての配列は、同じ定義を持ちます。それぞれ、3つの順序付けされたアイテムを次の順番で含んでいます：() 数値アイテム、(i) 数値アイテム、(ii) 文字列アイテム。このオブジェクトのスキーマの詳細は、下のスクリーンショットに表示されています。 3つの配列すべての定義は同じなため、定義は、`array_01` という名前のグローバル配列内で作成されています。 3つの配列 (`x`、`y`、および`z`) のそれぞれは 以下のグローバル配列を参照しています：`array_01`。



上のスクリーンショットでは、 青いハイライトで表示されている)配列 `x` が選択されています。詳細および制約は、それぞれ入力ヘルパーに表示されています (*上のスクリーンショット参照*)。許可されているアイテムの制約の数に注意してください。数は、制約入力ヘルパーで編集することができ、ダイアグラム内で表示されています。配列 アイテムは、配列自身の定義内で定義することができます。この場合は、グローバル定義 `array_01` (*下のスクリーンショット*) です。



以下の点に注意してください:

- `unique` 制約は、配列内のすべてのアイテムが一意であることを指定します。
- アイテムの番号付けは `0` から開始されます。
- ダイアグラム内のフレーズ、*3 <= アイテム <= 3* および *アイテム : 3..3* (*上のスクリーンショット参照*)、は、許可される最小および最大のアイテムを表しています。この場合、3つのアイテムが存在する必要があります。

10.9.9 アトミック型

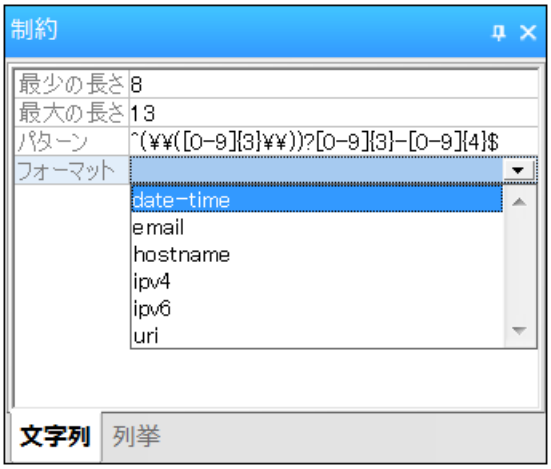
5つのJSON アトミック (単純型 または プリミティブ) として知られる型があります : (i) `string`、(ii) `number`、(iii) `integer`、(iv) `boolean`、および (v) `null`。これらのアトミック型のうちの1つの定義を指定するには、、以下のうち1つを行います :

- 定義のボックス内の *型* 値フィールドをダブルクリックして、型を選択します。
- 詳細入力ヘルパー内で、*型* フィールド内のドロップダウンリストから型を選択します。

各アトミック型の制約は、下に説明されています。

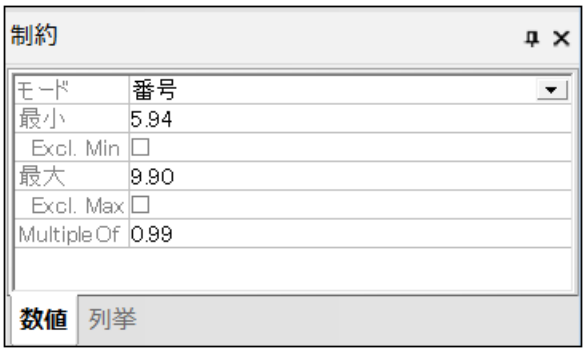
文字列

文字列 型に関しては、以下の制約を指定することができます : (i) 文字列の長さ (i) 文字列のパターンを説明する正規表現、 (ii) 仕様からの定義済みのフォーマット。



数値

数値 型は、2つの型 数値 および整数。 *下のスクリーンショット参照*)のための総合的な名前です。実際の型は、モードフィールド(デフォルトは数値)で設定されます。 2つの型の間の違いは、数値型は小数点を許可しますが、整数型は許可しない点です。*MultipleOf* フィールド内に値が存在すると、インスタンス値は、*MultipleOf* 値の整数である必要があります。



上のスクリーンショットで定義されている 数値 型の有効な値は、以下のとおりです :`5.94`、`6.93`、`7.92`、および `8.91`.

Boolean とNull

`boolean` 型は、値として`true` または`false` を取ります。 `null` 型 は、値として`null` を取ります。Neither 型は任意の制約を取ります。

10.9.10 型セレクター (Any、Multiple)

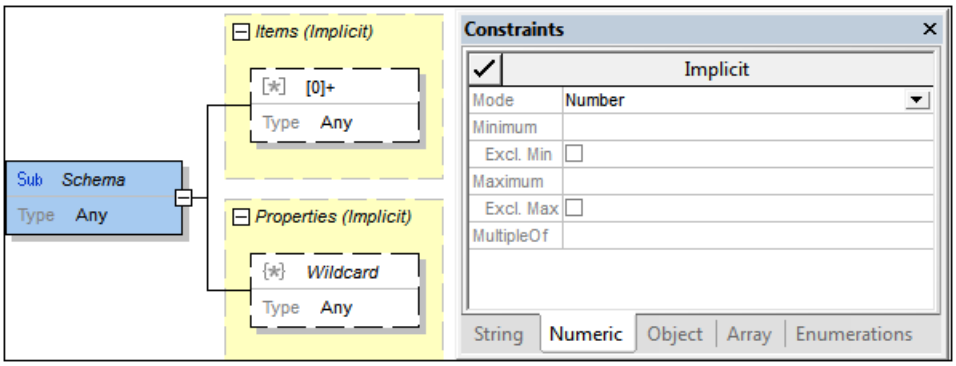
JSON スキーマビューの型コンボボックスのドロップダウンには、JSON 型 : ではない、以下の2つの型 があります : **any** および **multiple**。をリストします。これらは、型セレクターです。

- **any** 型セレクターは、任意のJSON 型を選択します。これは、*インスタンス*内で任意のJSON 型 がその特定の定義に対して有効であることを意味します。
- **multiple** 型セレクターは、1つまたは複数のJSON 型を選択します。これは、*インスタンス*が、スキーマー内で選択されたJSON 型の1つである場合、*インスタンス*はその特定の定義に対して有効であることを意味します。

---

any 型セレクター

any 型セレクターは、型が選択できるすべての箇所で選択されることができます。定義がスキーマに追加されると、 any が デフォルトの多々の選択になります。また、JSON 型 の1つが有効であることを指定します。これは、*インスタンス*型はオブジェクト、配列またはいずれのアトミック型 (string、number、integer、boolean および null) である可能性があります。



上の*スクリーンショット*では、サブスキーマは Any の型を有します。ですから、すべてのJSON 型 は、この定義に対して有効です。以下はユーザーインターフェイスに従い暗黙的に実装されています :

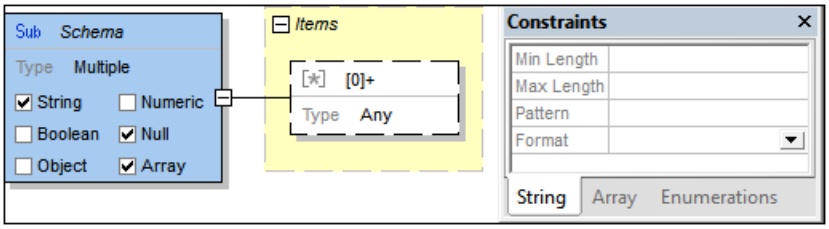
- オブジェクトが許可されているため、*プロパティボックス*は自動的に作成されます (*上のスクリーンショット参照*)。*プロパティボックス*は、デフォルトで、(Any の型のプロパティワイルドカードを使用して)任意の型の任意の数のプロパティを許可するように定義されています。プロパティの定義を自由に変更することができます。
- 配列が許可されているため、*アイテムボックス*は自動的に作成されます (*上のスクリーンショット参照*)。配列ボックスは、デフォルトで、(Any の型のアイテムワイルドカードを使用して)任意の型の任意の数のプロパティを許可するように定義されています。アイテムの定義を自由に変更することができます。
- 文字列と数値 (数字と整数)は、許可されているため、これらのアトミック型のための制約は、制約入力ヘルパー内で定義することができます。

これらの型すべては、ですから、暗黙的に、 Any 型セレクターにより定義されています。特定の型に型を変更する場合はその型を選択してください。オブジェクトと配列を指定するには別の方法があります : オブジェクトまたは配列を右クリックして、**明示的に設定**を選択します。この方法は、選択された型を型とし、他の型を削除、またはオブジェクト/配列型を非アクティブにします。

---

multiple 型セレクター

multiple 型セレクターは、型が選択される箇所すべてで選択されることができます。許可する型をチェックすることにより、1つまたは複数のJSON 型の選択を許可します (*下のスクリーンショット参照*)。選択された制約入力ヘルパー内の型のための制約を指定することができます。



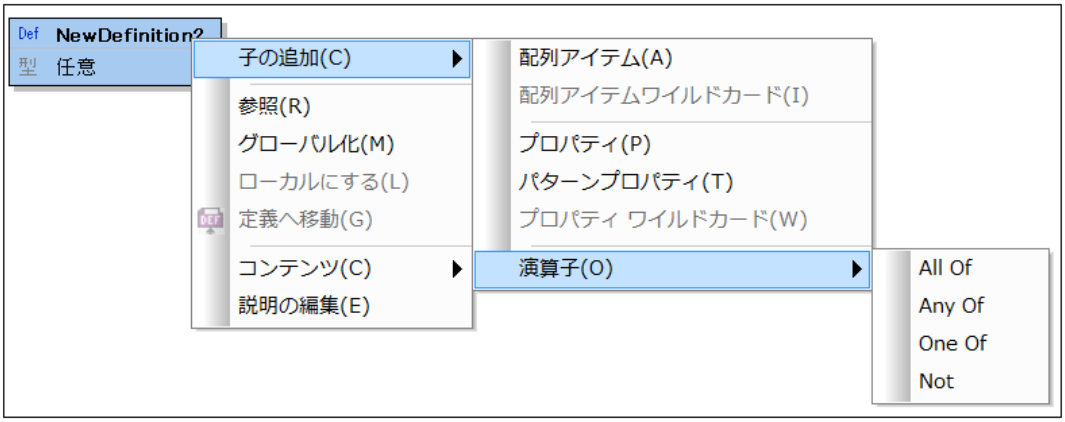
上のスクリーンショットでは、サブスキーマは `string`、`null`、および `array` の型を許可します (*スクリーンショット参照*)。

- 文字列の制約は入力ヘルパーの制約で定義されています。
- null 型は、furr 制約をとりません。
- 配列アイテムボックスが自動的に作成されます。許可された配列アイテムの数値と型を定義することができます。

インスタンスドキュメントでは、選択された型はサブスキーマに対応する箇所に許可されています。

10.9.11 演算子

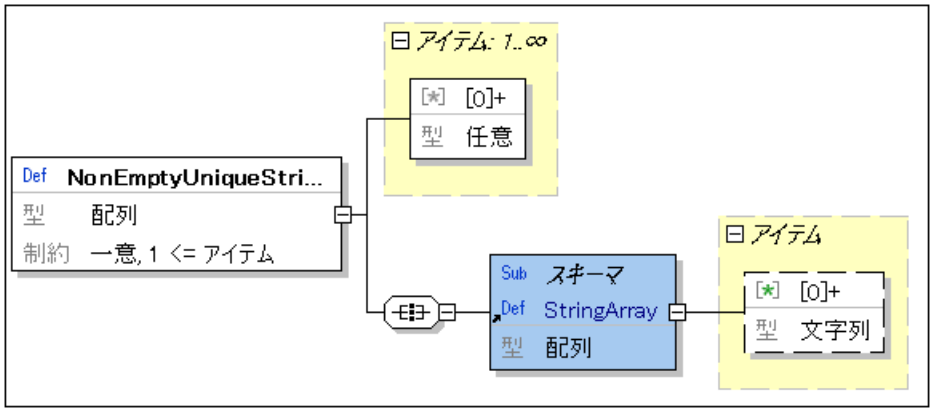
以下の4つの演算子が存在します : (i) `allOf`、(ii) `anyOf`、(iii) `oneOf`、および (v) `not`。演算子は、下に説明されるように有効性の条件を指定します。任意の定義に演算子を追加することができます。演算子 サブメニューにアクセスするには、演算子を追加する定義を右クリックして、**[子の追加 | 演算子]** を選択します (*下のスクリーンショット参照*)。



これらの演算子は、検証を成功するための条件を以下のように定義します :

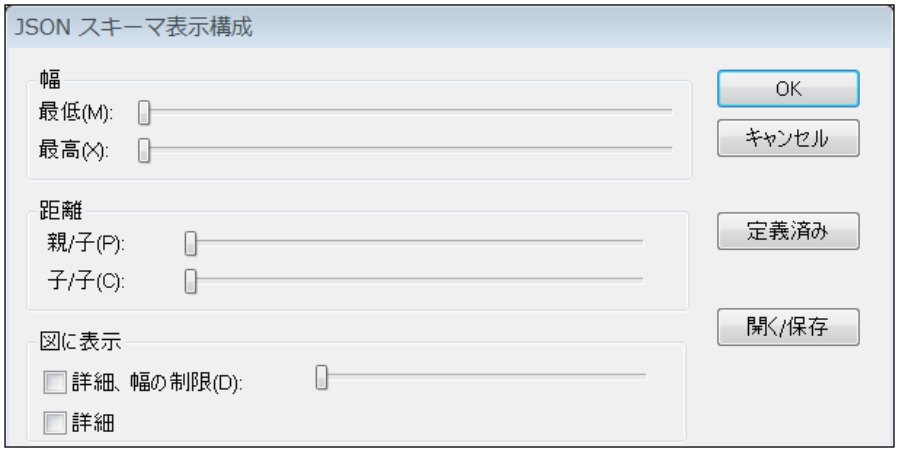
演算子	アイコン	詳細
	All Of	値は、演算子の子として追加された1つまたは複数のサブスキーマ (定義) の配列です。インスタンスは、配列内のすべての定義に対して有効な場合有効です。
	Any Of	値は、演算子の子として追加された1つまたは複数のサブスキーマ (定義) の配列です。インスタンスは、配列内の1つまたは複数の定義に対して有効な場合有効です。
	One Of	値は、演算子の子として追加された1つまたは複数のサブスキーマ (定義) の配列です。インスタンス配列内の1つの定義に対して有効な場合有効です。
	Not Of	値は、演算子の子として追加されサブスキーマ (定義) たです。インスタンスは、与えられた定義に対して無効な場合有効です。

下のスクリーンショットは、3つのサブスキーマ (定義) の配列を子として持つ All Of 演算子を表示しています。インスタンスを有効にするには、(ドキュメント構造のこの箇所で) 3つのサブスキーマ定義の1つに一致する1つのJSON データ構造を持つ必要があります。



10.9.12 デザインビューの構成

メインウィンドウがデザインビューモードに設定されている場合、構成の表示ダイアログ (下のスクリーンショット) にアクセスすることができます。デザインビューの外観をここで構成することができます。



デザインビューの次のアスペクトを構成することができます :

- *幅* .2つのスライダーはそれぞれ、デザインビュー内のボックスの最小および最大の幅を決定します。また、ボックスの許可される幅の範囲を決定します。

- *親 /子 の距離 :*は、階層構造の各レベル間の左右の間隔を設定します。
- *子 /子 の距離 :*は、各ボックス間の上下の間隔を設定します。
- *詳細の幅 :*は、詳細ラインの幅を設定します。テキストが幅をはみ出した場合、次のラインに折り返します。
- *詳細の表示 :*詳細ボックスのオプションをチェックまたはチェックを解除することにより、定義の詳細を表示または非表示に切り替えることができます。

10.9.13 JSON スキーマドキュメンテーションの生成

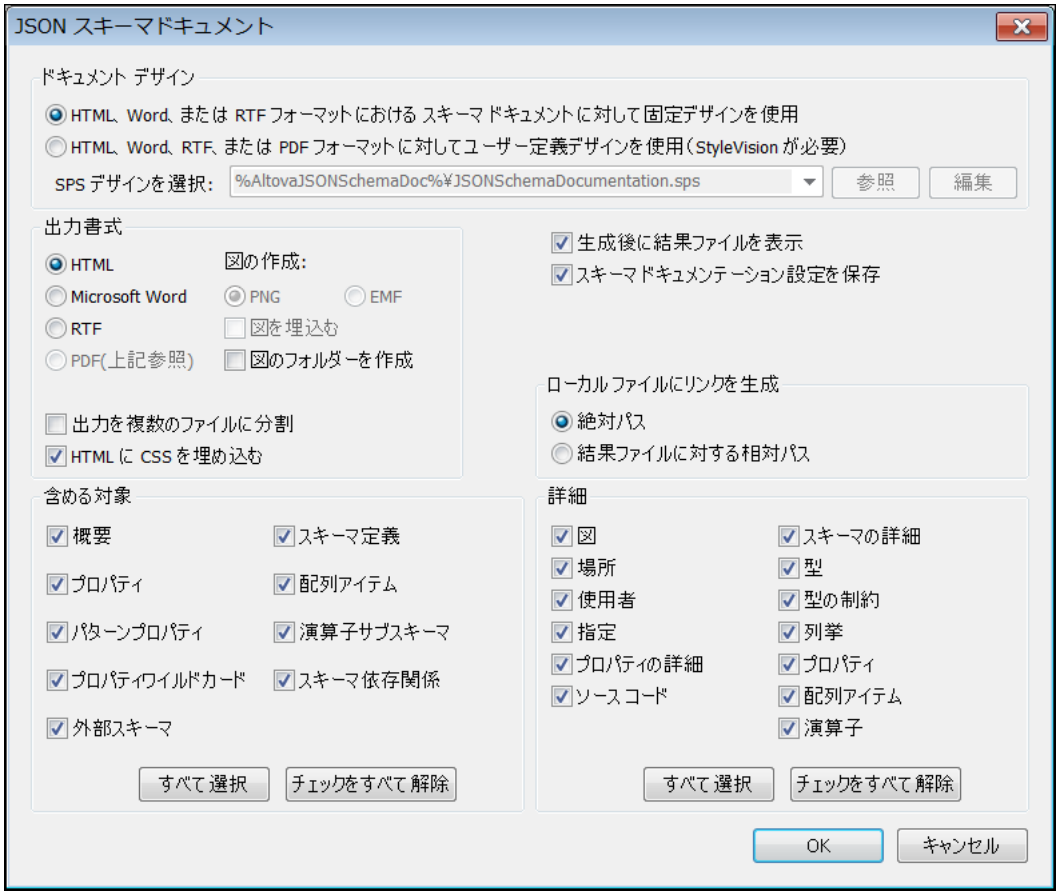
IJSON スキーマアクティブなドキュメントの場合、**[スキーマデザイン| ドキュメントの生成]** コマンドをクリックすることでドキュメントを生成することができます。ドキュメントの出力は、HTML、MS Word、または RTF ファイルで生成することができ、含まれるコンポーネントを指定することができます。関連する JSON コンポーネントは、生成されたドキュメント内でハイパーリンクが付けられ、簡単に移動することができます。

メモ: MS Word フォーマットでドキュメントを生成するためには、MS Word (バージョン 2000 または以降)がインストールされている必要があります。

JSON スキーマドキュメンテーション生成のステップ

JSON スキーマファイルのためにドキュメントを生成するためには、以下を行います :

1. JSON スキーマがアクティブなドキュメントであることを確認します。
2. スキーマビューに切り替えます。
3. メニューコマンド **[スキーマデザイン| ドキュメントの生成]** を選択します。これにより、JSON スキーマドキュメンテーション ダイアログボックスが開かれます (*下のスクリーンショット*)。
4. 生成する出力の型を次から選択します HTML、MS Word、または RTF。
5. ドキュメントに含まれる特別なコンポーネントと詳細を選択し、他のオプションを設定します (*下の JSON スキーマドキュメンテーションオプションを参照*)。



6. 「OK」をクリックして、JSON スキーマドキュメンテーションファイルの名前を、表示される名前をつけて保存ダイアログボックス内に入力します。

JSON スキーマドキュメンテーションオプション

次のドキュメントオプションから選択することができます :

- デザインテンプレート XMLSpy デザイン内でビルドイン(固定された)指定することができ、または SPS ファイル内に保存されるユーザー定義デザインであることもできます。ユーザー定義 デザインの使用方法に関しては、ユーザー定義デザインのセクションを参照してください。
- 出力フォーマットペイン内で必要なフォーマットを指定することができます : HTML、Microsoft Word、または RTF。ドキュメントは単一のファイルまたは複数のファイルに生成されることができます。複数のファイルが生成される場合、それぞれのファイルは、コンポーネントに対応します。どのコンポーネントが出力に含まれるかは、含むペインのチェックボックスを使用して指定します。
- ダイアグラムの埋め込みオプションは、オMS Word とRTF 出力 プションのために有効化することができます。このオプションがチェックされていると、ダイアグラムはPNG またはEMF フォーマットとして結果ファイル内に埋め込まれます。それ以外の場合は、ダイアグラムは、オブジェクトリンクを介した結果ファイル内にPNG またはEMF ファイルとして作成されます。出力がHTML の場合、全てのダイアグラムは、ドキュメント外部 PNG ファイルとして作成されます。
- 含むペイン内で、ドキュメントに含むアイテムを選択することができます。概要オプションは、ファイルの上にコンポーネント型別に整理された全てのコンポーネントをリストします。スキーマ定義が選択されていない場合、(外部スキーマ以外のすべての)子コンポーネントは無効化されます
- 詳細ペインは、各コンポーネントに含まれる可能性のある詳細をリストします。スキーマ定義が選択されていない場

合、全ての詳細は無効化されます。ドキュメントに含む詳細を選択します。

- 結果ファイルの表示オプションは全ての出力オプションのために有効化されています。このオプションがチェックされていると、結果はブラウザービュー (HTML 出力)、MS Word (MS Word 出力)、.rtf ファイルのデフォルトのアプリケーション (RTF 出力) に表示されます。

11 Avro、Avro スキーマ

Apache Avro™ は、データをコンパクトなバイナリフォーマットにシリアル化するためのシステムです。Avro データ構造は、JSON フォーマットで書かれているAvro スキーマで定義されており、実際のデプロイのシナリオでは、Avro ドキュメントは通常、Avro データ構造だけではなく、構造を定義するAvro スキーマを含む、バイナリファイルとしてシリアル化されています。Avro バイナリはですから、データとデータの構造の定義（Avro スキーマ）を含んでいます。Avro データは、しかしながら、JSON としてシリアル化することができ、（JSON ファイル内の）Avro データは、外部のAvro スキーマを参照することができます。

XMLSpy では次のAvro サポートをご利用いただけます：

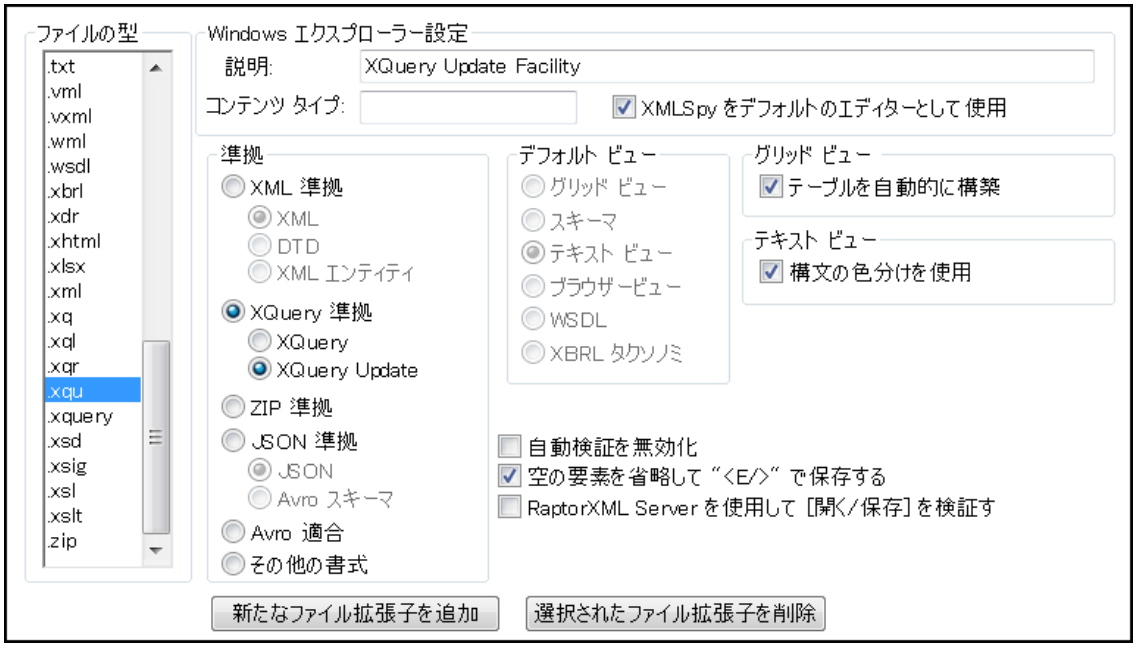
- テキストビューとグリッドビューで、Avro データ（`.json` JSON ドキュメントとして）を編集することができます。両方の種類のビューでインテリジェントな編集機能を使用することができます。データドキュメントにAvro スキーマを割り当て、検証することができます。
- （`.avsc` Avro スキーマドキュメントとして）Avro スキーマをテキストビューとグリッドビュー内で編集することができます。Avro スキーマをAvro スキーマ仕様に対して検証することができ、ビューにはインテリジェントな編集機能が搭載されています。
- テーブルビューでAvro データブロックを表示するAvro ビューでAvro バイナリインスタンス（`.avro` ファイル）を確認することができます。
-

Altova のRaptorXML エディションは、Avro を更にサポートします：

- （Avro スキーマに対しての）Avro データ（JSON-シリアル化された；`.json` ファイル）検証
- Avro データ（バイナリシリアル化された；`.avro` ファイル）検証
- Avro スキーマ（通常 `.avsc` ファイル）の（Avro スキーマ仕様に対して）検証

既存の Avro ドキュメントを開き、新規ファイルを作成する

「オプション」|「ファイルの型」タブから、（下のスクリーンショット）、異なる型のAvro ドキュメント（JSON データフォーマット、Avro スキーマ、Avro バイナリ）が開かれるデフォルトのビューを設定することができます。使用することのできるビューを随時切り替えることができます。



ドキュメントの型	ファイル拡張子	適合性	使用することのできるビュー
JSON フォーマットのAvroデータ	`.json`	JSON 適合 | JSON	テキストビュー、グリッドビュー
Avro スキーマ	`.avsc`	JSON 適合 | Avro スキーマ	テキストビュー、グリッドビュー
バイナリファイル内のAvroデータ	`.avro`	Avro 適合	Avro ビュー

次の点に注意してください:

- 選択された型の既存のドキュメントと新規ドキュメントは、ファイル型タブで選択されているデフォルトのビューで開かれます。
- 読み取り専用のビューであるAvro バイナリは、Avro ビューのみで確認することができます。ファイル型がAvro-適合と定義されている場合、使用することのできるビューはAvro ビューのみです。
- 上にリストされているAvro ドキュメントの1つとして、特定のファイル拡張子のファイルをXMLSpy に読み取らせる場合、子の新規ファイル拡張子を追加し、対応する適合に割り当てます。
- 新規ドキュメントを作成するには、**「ファイル | 新規作成」**クリックして、ドキュメントの型を選択します。Avro バイナリは、バイナリであるため、この方法で作成できず、Avro ビューのみで読み取ることができます。

11.1 Avro スキーマ

Avro スキーマは、Avro データブロックを指定します。どのデータフィールドが期待され、どのように値が示されるかを指定します。Avro スキーマに関する情報、とその仕様はここで確認することができます。

Avro スキーマに関する以下の点に注意してください:

- Avro スキーマをJSON フォーマット内で作成
- Avro スキーマは以下であることができます:JSON 文字列、JSON オブジェクト、または、JSON 配列
- Avro スキーマは、以下の4つの属性を含むことができます:**name**、**namespace**、**type**、と**fields**
- 以下の8つのプリミティブなデータ型が存在します:**null**、**boolean**、**int**、**long**、**float**、**double**、**bytes**、と**string**
- 以下の6つの複合型が存在します:**records**、**enums**、**arrays**、**maps**、**unions**、と**fixed**
- プリミティブな型には属性が存在しません。また、各複合型には、自身の属性のセットが存在します。

Avro スキーマに関する更に詳しい情報は、Avro スキーマ仕様を参照してください。

サンプル

以下に、Avro スキーマの簡単なサンプルが表示されています。それぞれがJSON フォーマットのAvro データスニペットに対応しています。スキーマは特定の構造を定義することに注意してください。一部の場合、定義された構成が複数回インスタンス化される場合があり、結果の出力は、有効ではないJSON になる可能性があります。例えば、スキーマがJSON オブジェクトの構造を定義する場合があります。JSON オブジェクトが複数回インスタンス化されると、各オブジェクトは、個別に、Avro スキーマに対して有効ではあるが、コンテナーオブジェクトが存在しないため、ドキュメント全体は、有効なJSON ではない場合があります。有効なJSON が必要な場合、JSON オブジェクトの配列を検証するためにAvro スキーマを書き直す必要があるかもしれません。サンプル 4 と 5 を比べて、上記の点を確認してください。

- *01: JSON 文字列としてのAvro スキーマ*

  このスキーマは、単一の文字列で、データブロックがAvro (int) プリミティブデータ型 `"int"` の値を持つように指定します。
  有効なAvro: `2016`
  無効なAvro: `"2016"`

- *02: JSON オブジェクトとしてのAvro スキーマ*

  このスキーマは、前のスキーマと同じ内容を指定しますが、JSON オブジェクトです。データブロックはAvro (int) プリミティブデータ型の値であるアイテムを1つ含む必要があります。

```
{
  "type":"int"
}
```

  有効なAvro: 2016
  無効なAvro: "2016"

- *03: JSON オブジェクトとしてのAvro スキーマ:整数の配列*

  このスキーマは、整数の配列を指定するJSON オブジェクトです:

```
{
  "type":"array",
  "items":"int"
}
```

有効な Avro: [2016,2017]
有効な Avro: [2016]
有効な Avro: [2016]
無効な Avro: 2016,2017

- *04: JSON オブジェクトとしての Avro スキーマ :レコード*

  このスキーマは、単一のレコードを指定する JSON オブジェクトです :

```
{
  "type":"record",
  "name":"ages",
  "fields" :[
  {"name":"name","type":"string"},
  {"name":"age","type":"int"}
  ]
}
```

  有効な Avro: {"name":"John","age":35}

- *05:JSON オブジェクトとしての Avro スキーマ複数のレコード*

  このスキーマは、レコードの配列を指定する JSON オブジェクトです。それぞれは JSON オブジェクトである必要があります :

```
{
  "type":"array",
  "items":{
  "type":"record",
  "name":"ages",
  "fields" :[
  {"name":"name","type":"string"},
  {"name":"age","type":"int"}
  ]
   }
}
```

  有効な Avro: [{"name":"Mary","age":34},{"name":"John","age":35}]

Avro スキーマ ファイル型

XMLSpy の機能を Avro に関連した、編集と検証で使用する場合、XMLSpy は、ファイルを Avro スキーマとして認識する必要があります。ファイルは、ファイルの拡張子が XMLSpy のオプションダイアログ (**「ツール」| 「オプション」| 「ファイル型」**) で定義されているとおり Avro スキーマとして識別されます。XMLSpy のデフォルトの設定は、1 つのファイル拡張子 — `.avsc` 拡張子？ を Avro スキーマとして指定しています。Avro スキーマドキュメントを指定する他のファイル拡張子を作成する場合、これらのファイル拡張子を Avro スキーマ拡張子として、オプションダイアログ内のリストに追加します。

Avro スキーマの作成と編集

XMLSpy 内では、Avro スキーマファイル拡張子をファイルの型と指定することにより、Avro スキーマとして、新規のファイルを作成することができます。XMLSpy は、インテリジェントな編集ヘルプを入力時に提供します。この機能にはコンテキストに敏感なキーワードの提案、自動かっこ、引用符の入力、構文の色分け、キーワードの自動補完などが含まれています。

更に、3つの入力ヘルパーが搭載されています：JSON プロパティ、JSON 値、および JSON エンティティ。使用することのできるエントリは、コンテキストに敏感です。挿入するエントリをカーソルが現在ある場所でダブルクリックします。 **検証 | XML の検証 (F8)」**メニューコマンドを利用して、ファイルをAvro スキーマ仕様に対して検証 することができます。

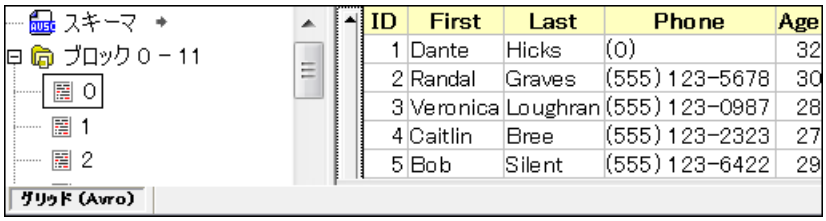
11.2 JSON フォーマットの Avro データ

Avro データは、バイナリフォーマット またはJSON フォーマットでシリアル化されることができます。以下のポイントは、XMLSpy のAvro フォーマットへのサポートについて説明しています。

- JSON フォーマットのAvro データは、通常、`.json` ファイルで保存されます。XMLSpy が追加ファイル拡張子をJSON 適合であると認識することを指定できます。 .
- Avro JSON ファイルは、テキスト ビュー とグリッドビューで開くことができ、これらのビューで編集することができます。
- Avro スキーマ ファイルをAvro JSON ファイル に割り当てることができ、データファイルを Avro スキーマに対して検証することができます。.
- JSON ドキュメントのためのインテリジェントな機能は、テキスト ビュー とグリッドビューの両方で使用することができます。更に、Avro スキーマがJSON フォーマット内のAvro データドキュメントに割り当てられている場合、Avro インスタンス内でスキーマにより定義されたキーワードの自動補完を使用することができます。

11.3 Avro ビュー :Avro バイナリのグリッドビュー

Avro データをバイナリフォーマットまたはJSON フォーマットにシリアル化することができます。バイナリフォーマットは、Avro データ構造とスキーマの両方を含み、自動化されたデータ処理の過程で生成されます。Avro バイナリは、簡単に閲覧できるテーブルフォーマット内のAvro データ構造を表示する、グリッドビューのAvro ビュー内で開くことができます (*下のスクリーンショット参照*)。Avro ビューは、ですから、ユーザーが使いやすいAvro バイナリビューアーとしての役割を果たします。

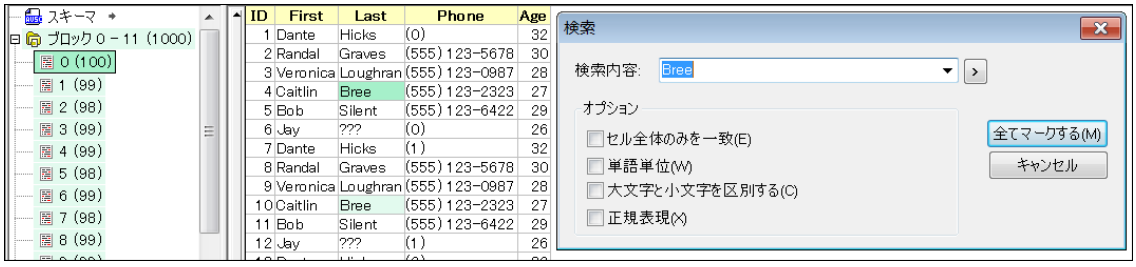


以下の点に注意してください:

- Avro バイナリは、XMLSpy などにより識別される必要があります。これは、**「オプション」| 「ファイル型」** タブで Avro バイナリのファイル拡張子をAvro 適合に設定して行うことができます。デフォルトでは、ファイル拡張子 `.avro` は、Avro に適合するように設定されています。Avro バイナリに適合するファイルの型を追加することもできます。これらのファイルは、Avro ビューで開かれます。
- 2つのペインから構成されAvro ビュー: (i) ナビゲート用の*ブロック*ペイン。および (i) ブロックペイン内で選択するデータ構造を表示する *データ*ペイン。
- *ブロック*ペインは、データブロックを`1000` のグループに整理します。それぞれのグループは、展開および折りたたむことができます。 データブロックは、インデックス番号別に表示されています。
- 特定のデータブロックを確認するには、*ブロック*ペイン内で検索し、ダブルクリックします。
- ブロックペインは、スキーマと呼ばれるエントリを含みます。エントリの右横のボタンをクリックすると、Avro スキーマが Avro バイナリから抽出され、新しいテキストビュータブで開かれます。Avro スキーマを保存することもできます。

テキストの検索

テキスト文字列を検索するには、メニューコマンド **「編集 | 検索」** (**Ctrl+F**) を選択します。表示されるダイアログに、検索用語をテキスト文字列、または、正規表現として入力します (*下のスクリーンショット参照*)。ここで説明されている適用することのできるオプションを選択し、**「全てをマークする」** をクリックします。



- *ブロック*とデータペイン内で一致するエントリはハイライトされます :現在選択されている項目は、深緑色で表示され、その他は薄い緑で表示されています。
- *ブロック*ペイン内では、各ブロック内の数は、エントリの横に表示された数と一致します。
- 一致する項目をブロックごとに確認、ブロック内のフィールドを選択して、または、**F3** (**編集 | 次を検索**) と**Shift +F3** (前を検索) を使用してナビゲートできます。
- Avro ビューは読み取り専用です。Avro バイナリ内のデータを編集することはできません。

12 WSDL と SOAP

Altova ウェブサイト: WSDL エディター

このセクションではXMLSpy のWSDL ならびにSOAP 機能を説明します。

WSDL

WSDL ドキュメントは、ウェブサービスの記述を行うXML ドキュメントです。XMLSpy ではWSDL 1.1 ならびにWSDL 2.0 がサポートされ、WSDL ビューにてWSDL 1.1 ならびに 2.0 ドキュメントの作成や編集を行うことができます。WSDL ビューでは、WSDL のバージョンに関係なく、適切な編集環境が提供されます。

XMLSpy のWSDL ビューでは、グラフィカルな構築ブロックを使用しながらWSDL ドキュメントの構築を行うことができ、作成作業の大幅な簡易化を行うことができます。WSDL ビューに関する記述は編集ビューの対応するセクションを参照ください。WSDL ドキュメントの作成方法に関しては、このドキュメント内のWSDL チュートリアルを参照ください。WSDL ドキュメントの閲覧や編集はテキストビューならびにグリッドビューで行うことができます。両方のビューで、WSDL ドキュメントは通常のXML ドキュメントとして扱われます。

WSDL ビューで表示されるWSDL ファイルに対するXML 署名は独立したファイルとして作成することができ、WSDL ファイル内に含めることができます ("enveloped")。署名の詳しい使用方法については、XML 署名のセクションを参照ください。

SOAP

SOAP はXML のメッセージ交換プロトコルで、アプリケーション間のメッセージ交換に使用されます。XMLSpy では、XML ドキュメントの編集機能を搭載したテキストビューやグリッドビューにてSOAP ドキュメントを作成、編集できるだけでなく、WSDL ファイルからSOAP リクエストファイルを作成することもできます。WSDL ファイルからSOAP リクエストファイルを作成する方法はWSDL チュートリアルに記載されています。 $SOAP メニューのコマンドを使用することでXMLSpy はSOAP リクエストの送受信を行うことができます。更に、このセクション内のサブセクションで解説されるXMLSpy のSOAP デバッガーを使用することでSOAP リクエストのデバッグを行うこともできます。

12.1 WSDL チュートリアル

このチュートリアルは2つのパートから構成されます：

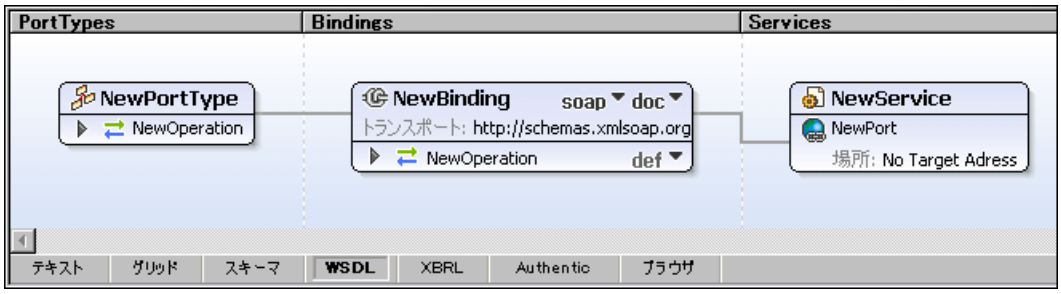
- 最初のパートでは、XMLSpy のグラフィカルな WSDL ビューを使って WSDL 1.1 ドキュメントを作成する方法が記述されます。このパートでは以下のような操作を行ないます。(i) **「ファイル | 新規作成」**メニューオプションを使った基本的な WSDL 1.1 ドキュメントの作成、(ii) ポートタイプの作成、(iii) バインディングの作成、(iv) サービスとポートの作成、(v) ドキュメントの検証と保存。
- 2番目のセクションでは、ウェブサービスへの接続、WSDL ファイルのローカルでの保存、そしてウェブサービスに SOAP リクエストを送信する方法を説明します。

これらの操作は全てグラフィカルな WSDL ビューにて行うことができ、テキストビューを使用する必要はありません。ドラッグアンドドロップにより WSDL コンポーネントの操作を行ったり、WSDL ビューの入力ヘルパーを使用したプロパティ値の変更を行うことができます。

関連項目：　WSDL ドキュメント処理の情報については、WSDL ビューならびに「ユーザーレファレンス | WSDL メニュー」を参照ください。

12.1.1 新規ドキュメントの作成

新しい WSDL ドキュメントを作成するには **「ファイル | 新規作成」**コマンドを選択してください。新規ドキュメントを作成ダイアログにて、WSDL（WSDL Web Service Description v1.1）を作成するドキュメントの種類として選択し、**「OK」**をクリックします。この操作により新たなドキュメントのスケルトンが WSDL ビューにて作成されます（以下のスクリーンショットを参照）。



ターゲット名前空間の割り当て

テキストビューへ画面を切り替えます。ws 要素の開始タグは以下のようになっているはずです：

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
3       xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
4       xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
5       xmlns:xs="http://www.w3.org/2001/XMLSchema"
6       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
7       xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
8       xmlns:tns="http://new.webservice.namespace"
9       targetNamespace="http://new.webservice.namespace">
10
```

ターゲット名前空間属性を`http://mywebservice.namespace` または好みの名前に変更します（このチュートリアルでは WSDL ドキュメントの作成に重点をおいており、実際にサービスを作成するわけではありません）。その他にも、`tns` 属性の値を`http://mywebservice.namespace`（または、ターゲット名前空間に設定した値）に変更します。

メモ　スタート時点のスケルトンドキュメントでは、WSDL 要素がターゲット名前空間内にあり、WSDL 要素への参照は`tns` プレフィックスを使用して行われます（例：`<wsdl:binding name="NewBinding" type="tns:NewPortType">`）。`tns` プレフィックスをターゲット名前空間にマッチさせるには、その名前空間の値もターゲット名前空間と一致させる必要があります。

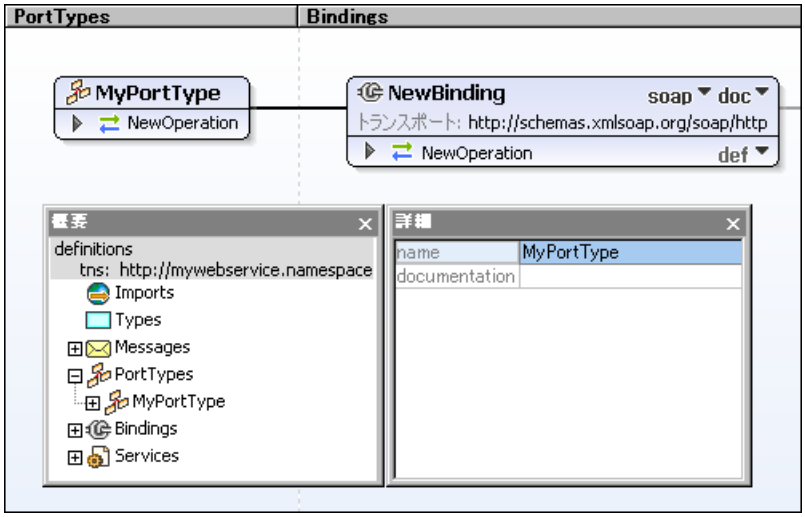
12.1.2 ポートタイプの作成

ポートタイプの作成には以下の操作が含まれます：

- ポートタイプの名前作成
- オペレーションの追加
- 入力ならびに出力メッセージの作成
- メッセージへパラメーターを追加

ポートタイプの名前作成

デザインにある NewPortType ボックスのタイトルバーをダブルクリックして、NewPortType を MyPortType に変更した後、**[Return]** により確定します。概要ならびに詳細入力ヘルパー内に表示されるポートタイプの名前も変更されていることに注目してください（以下のスクリーンショット参照）。



オペレーションの挿入

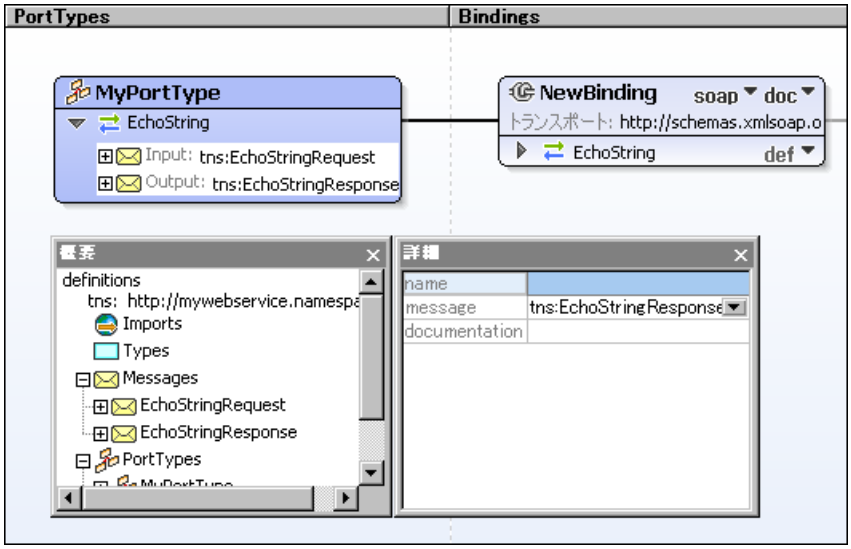
MyPortType には既に NewOperation という名前のオペレーションが存在しているので、このオペレーションを使用することにします。NewOperation という名前を、例えば EchoString に変更します（名前をダブルクリックした後 **[Return]** により確定）。ポートタイプにオペレーションを追加するには、ポートタイプボックスを右クリックして、**[オペレーションを追加]** を選択し、目的の種類に沿ったオペレーションをクリックします。

入力ならびに出力メッセージの追加

オペレーションがポートタイプに追加されていれば、オペレーションの種類を以下の5つの候補から選択することができます：

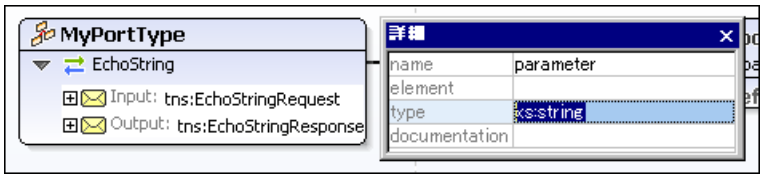
- 応答要求
- 送信請求/応答
- 一方向
- 通知
- 空のオペレーション

各オペレーションの種類に対して入出力メッセージが自動的に追加されます。**[空のオペレーション]** が指定されたときには、右クリックして挿入するメッセージを選択することができます。メッセージを右クリックして、**[入力/output/fault 要素を削除]** を選択することでメッセージを削除することができます。EchoString オペレーションの例では、入力 メッセージを EchoStringRequest へ、Output メッセージを EchoStringResponse へそれぞれ変更します。



メッセージへパラメーターを追加

各 Input または Output メッセージは、デフォルトで追加される単一の `xs:string` メッセージ (またはパラメーター) と共に作成されます。パラメーターの追加を行うには、メッセージまたはパラメーターを右クリックし、**[メッセージパート (パラメーター)]** を追加を選択します。



パラメーターの編集を行うには、(i) テキストをダブルクリックして編集するか、(ii) パラメーターを右クリックして、**[編集]** を選択するか、(iii) 詳細入力ヘルパーを使用 (上のスクリーンショットを参照) します。

12.1.3 バインディングの作成

バインディングは特定のポートタイプに対するデータ形式を指定したプロトコルです。従って、バインディングの作成には、以下のような操作が伴います:

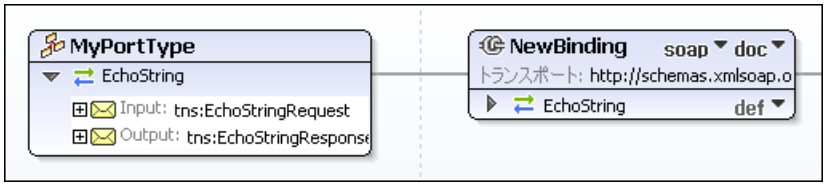
- ポートタイプへバインディングを割り当て
- バインディングのプロトコルとデータ形式の指定
- バインディングをポートへ関連付け

バインディングをポートへ割り当てる

デザイン内の何も無い箇所を右クリックして、**[バインディングを挿入]** を選択するなどして新しいバインディングが作成されたとき、割り当てられているポートはありません (以下のスクリーンショットを参照)。新規 WSDL ドキュメントを作成した場合、デフォルトで作成されたバインディングがデフォルトで作成されたポートタイプに割り当てられ、2つのボックスが線で接続されます。

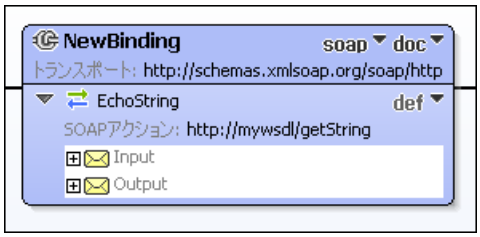
ポートタイプにバインディングを割り当てるには、上のスクリーンショットにあるように新規バインディングボックスにてポートタイ

エントリの下矢印をクリックします。ドキュメント内で定義されているポートタイプのリストが表示されるので、バインディングの割り当てを行いたいポートタイプを選択します。ポートタイプにバインディングが割り当てられた場合、以下にあるようにポートタイプとバインディングのボックス間を結ぶ線として割り当てが表示されます。



プロトコルとデータ形式の選択

バインディングのプロトコルは、バインディングボックスのタイトルバーにある下向き矢印から選択することができ、4つの中からプロトコルを選択します：SOAP、SOAP 1.2、HTTP-GET、HTTP-POST。SOAP 1.1 または SOAP 1.2 プロトコルが選択された場合、データ形式として (プロトコルリストの右側の下向き矢印から表示される) document または rpc を選択することができます。



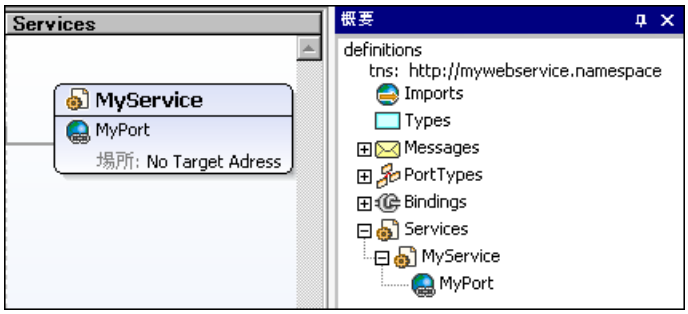
バインディング内部にある各オペレーションは、上のスクリーンショットにあるように、デザインまたはオペレーションを選択時に詳細入力ヘルパーにて定義することができます。

バインディングにポートを割り当てる

バインディングにポートを割り当てるには、まずポートを定義する必要があります。ポートの作成とバインディングの割り当てに関しては[サービスとポートの作成]セクションにて記述されます。

12.1.4 サービスとポートの作成

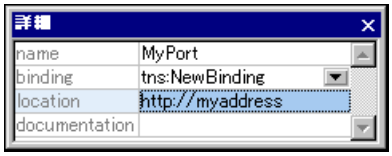
新しいサービスを追加するには、デザインの何も無い箇所を右クリックして、コンテキストメニューから**[サービスを挿入]** を選択します。新規 WSDL ドキュメントを作成した場合、デフォルトでサービスがデザイン内に表示されます。サービス名をダブルクリックし、名前を編集後 **[Return]** を押下することでサービスの名前変更を行うことができます。サービスの名前は概要入力ヘルパーでも変更されることに注目してください (以下のスクリーンショットを参照)。



概要入力ヘルパーにおいて名前の変更を行うには、NewPort エントリをダブルクリックして、MyPort と変更した後 **[Return]** により確定します。デザイン内のポート名も変更されることに注目してください (上のスクリーンショットを参照)。ポートを追加するには、サービスまたはポートを右クリックし、コンテキストメニューから**[ポートを挿入]** を選択します。

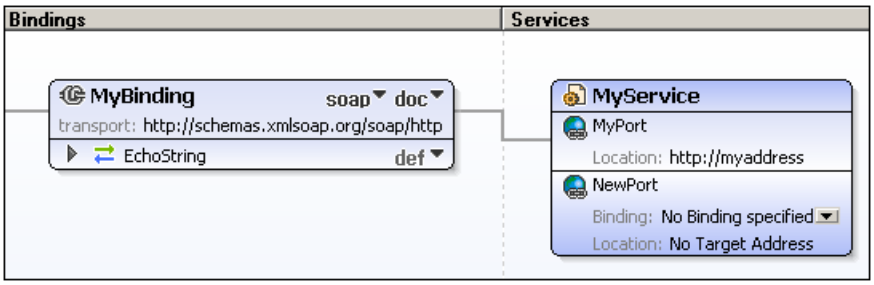
ポートのアドレスを入力

ポートのアドレスは、(i) デザインの場所アイテムから直接、または (ii) 以下のスクリーンショットにあるように詳細入力ヘルパーから (location フィールドをダブルクリックして) 編集することができます。



ポートをバインディングに割り当てる

ポートはバインディングとネットワークアドレスを結びつけるエンドポイントです。ポートのアドレスが一度定義されると、必要な作業はそのポートをバインディングに割り当てるだけとなります。ポートをバインディングに割り当てるには、ポートのバインディングアイテムにある下向き矢印をクリックして、ドキュメント内で定義されているバインディングのリストから選択を行ないます。



注意: ポートが既にバインディングに割り当てられており、他のバインディングへの割り当てを行ないたい場合、まずポートの右クリックメニューから、バインディング参照を削除した後、新規バインディング参照を追加する必要があります。

12.1.5 WSDL ドキュメントの検証

WSDL ドキュメントを完成させると、**[XML | XML 検証 (F8)]** コマンドによりドキュメントの検証を行うことができます。検証の結果はメッセージウィンドウ内に表示されます (以下のスクリーンショットを参照)。

検証により、発見されたエラーに関する詳しい情報が表示され、エラーの場所を素早く見つけることができます。

12.1.6 ウェブサービスへの接続とファイルを開く

このセクションでは以下の操作方法について説明します：

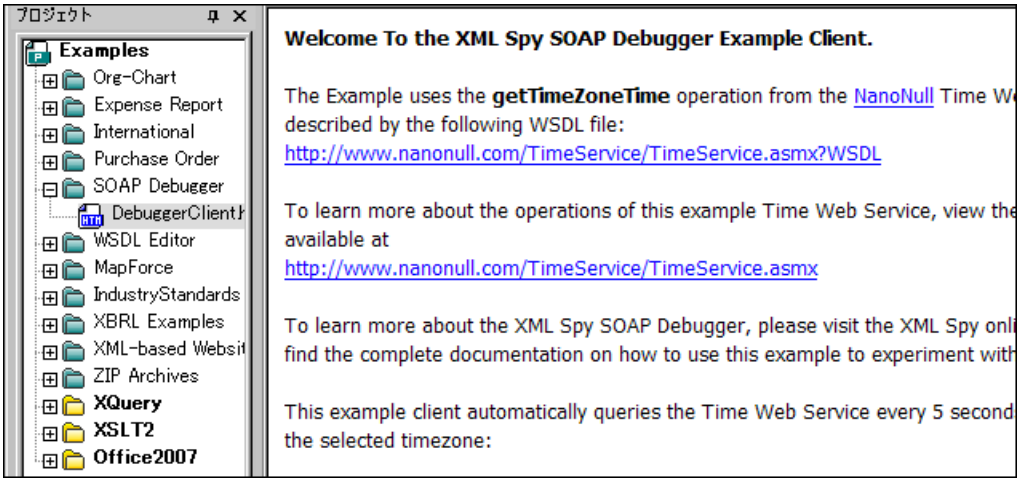
- XMLSpy を使ったウェブサービスへのアクセス
- XMLSpy を使って WSDL ファイルを開く

- XMLSpy を使ってWSDL ドキュメントに割り当てられたXML スキーマを開く

ウェブサービスへのアクセス

通常ウェブサービスへはHTML ページを通じてアクセスが行われます。このようなページの例の1つとして Examples/SOAP Debugger にあるDebuggerClient.htm があります。このページにより表示されるウェブサービスにアクセスするには：

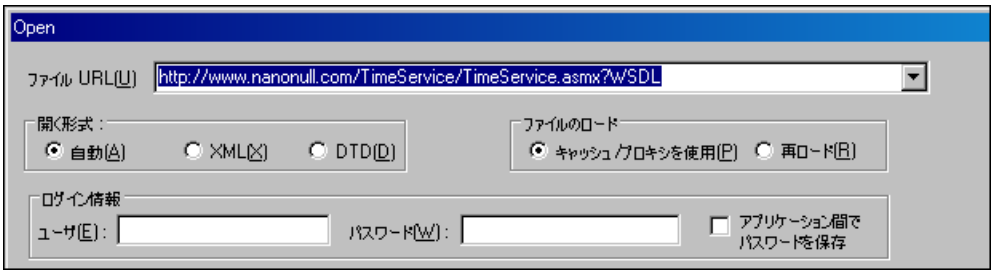
1. プロジェクトウィンドウが表示されていなければ、ウィンドウを有効にします **[ウィンドウ | プロジェクトウィンドウ]**。
2. SOAP Debugger フォルダーにある展開アイコンをクリックし、DebuggerClient.htm をダブルクリックします。この操作により、メインウィンドウにSOAP Debugger Example Client が表示されます。



XMLSpy を使って WSDL ファイルを開く

ウェブベースのWSDL ファイルをXMLSpy で開くには：

1. メニューオプション **[ファイル | 開く]** を選択し、ファイルを開くダイアログにて **「URL にスイッチ」** ボタンをクリックします。アドレスhttp://www.nanonull.com/TimeService/TimeService.asmx?WSDL をダイアログのファイルURL フィールドに入力またはコピーします。



2. **開く** をクリックしてWSDL ファイルをロードします。WSDL ファイルがテキストビューで表示されます。
3. メニューオプションから**[ファイル | 名前を付けて保存]** を選択し、ファイルをローカルに、例えば timeservice.wsdl という名前で保存します。
4. WSDL ビュータブをクリックして、ファイルをグラフィカルなWSDL エディターで表示します。

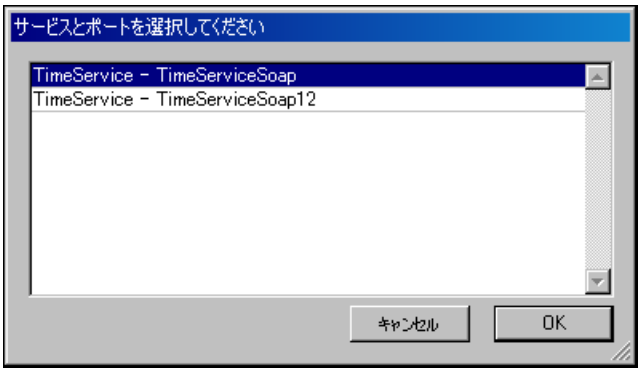
アクティブなWSDL ファイルに割り当てられたスキーマを閲覧する

timeservice.wsdl ファイルをWSDL ビューにてアクティブなドキュメントとして選択している場合、メニューオプションから**[WSDL | 型 | スキーマビューでスキーマを編集]** を選択します。この操作によりtimeservice.wsdl ファイルで使用されている全てのデータ型が定義されたスキーマが開かれます。スキーマの修正や、変更の保存を行うことができます。これらの変更はWSDL ファイルが再度分析される際に反映されます。

12.1.7 WSDL ファイルから SOAP リクエストを送信

timeservice.wsdl ファイルからSOAP リクエストを送信するには：

1. timeservice.wsdl をメインウィンドウでアクティブなファイルとして選択します。
2. メニューオプションから**[SOAP | 新規 SOAP リクエストを作成]** を選択します。
3. timeservice.wsdl を選択して、 **「OK」** で確定します。
4. ドキュメント内で定義されたサービスにて、SOAP 1.1 または 1.2 バインディングを参照するポートが2つ以上ある場合、 以下のスクリーンショットで示されるように、ポップアップが表示され、目的のサービスとポートを選択するよう促されます。選択を行い、 **「OK」** で確定します。



5. その後表示されるダイアログでは、SOAP オペレーション (例えば getServiceTime) を選択して **「OK」** をクリックします。

この操作により getServiceTime オペレーションを含むSOAP リクエストドキュメントが作成されます。必要に応じてドキュメントを保存することもできます。

6. 目的のドキュメントがアクティブな状態でメニューオプションから**[SOAP | サーバーにリクエストを送信]** を選択します。Nanonull.com タイムサービスから得られた現在のサーバーの時間である getServerTimeResult 要素を含むSOAP レスポンスドキュメントがメインウィンドウに表示されます。

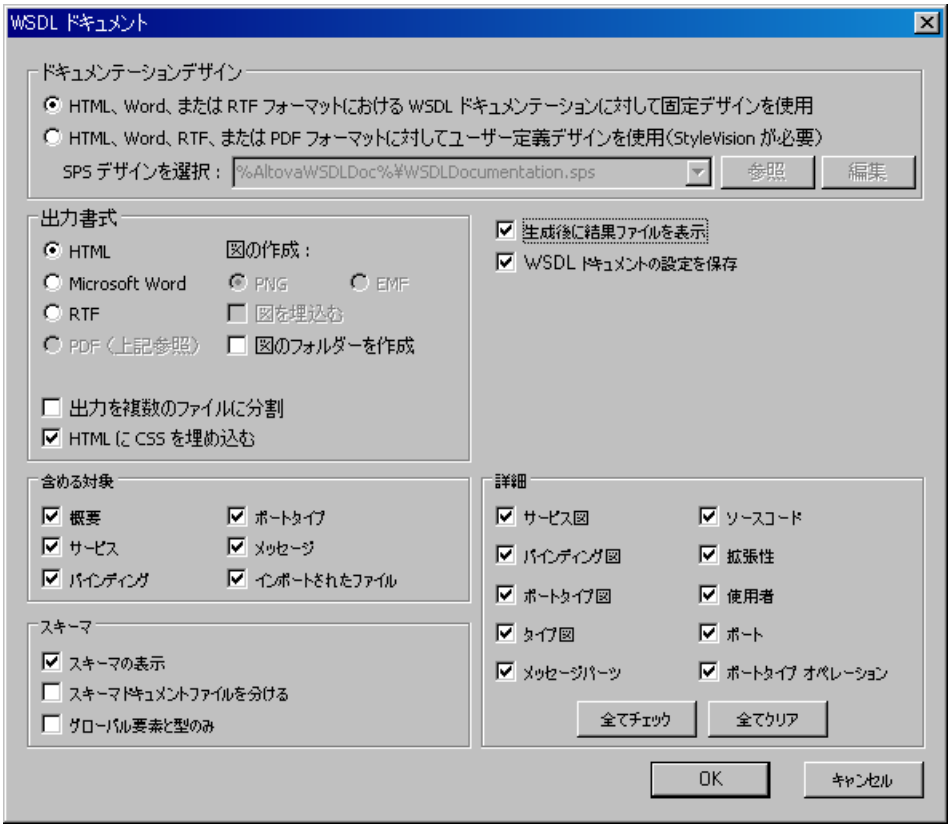
12.1.8 WSDL ドキュメンテーションの作成

[WSDL | ドキュメンテーションの生成] オプションを使えば、現在のWSDL ドキュメントに関する詳細なドキュメンテーションを生成することができます。生成されるドキュメンテーションはHTML、Microsoft Word、またはRTF の形式から選択し、またドキュメンテーションに含まれるコンポーネントも選択することができます。関連するWSDL 要素同士は生成されたドキュメンテーション内でハイパーリンク化され、ナビゲーションを簡単に行うことができます。

注意: Microsoft Word の形式でドキュメンテーションを生成するには、Microsoft Word (バージョン 2000以降) がインストールされている必要があります。

WSDL ファイルのドキュメンテーションを生成するには：

1. timeservice.wsdl をアクティブなドキュメントにします。
2. **WSDL** ビューに切り替えます。
3. メニューオプションから**[WSDL | ドキュメンテーションの生成]** を選択します。
   この操作により以下のスクリーンショットにある WSDL ドキュメントダイアログが起動します。
4. HTML、Microsoft Word、RTF から生成したい出力ファイルの形式を選択します。
5. ドキュメンテーションに含めたい WSDL コンポーネントを選択し、その他のオプションを指定します（以下の WSDL ドキュメンテーションオプションを参照ください）。



6. **「OK」** をクリックして、名前を付けて保存ダイアログボックスにて WSDL ドキュメンテーションファイルの名前を入力します。

WSDL ドキュメンテーションオプション

以下のドキュメンテーションオプションを指定することができます：

- ドキュメンテーションの形式は出力書式ペインにより指定されます。ドキュメンテーションは単一のファイル、または複数のファイルに分割して生成することができます。複数のファイルが生成された場合、各ファイルが各コンポーネントに対応します。ドキュメンテーションに含まれるコンポーネントの種類は、含める対象ペインにより指定されます。
- 図を書き込むオプションは Microsoft Word ならびに RTF 出力オプションに対して有効になります。このオプションにチェックを入れると、ダイアグラムファイルは PNG または EMF ファイルとして出力ファイル内に埋め込まれます。チェックが入れられていない場合、ダイアグラムは PNG または EMF ファイルとして作成され、オブジェクトリンクにより出力ファイルから参照されます。出力形式が HTML の場合、全てのダイアグラムは PNG ファイルとして作成され、ドキュメントから参照されます。
- 含める対象ペインでは、ドキュメンテーションに含めるアイテムを選択します。概要オプションにより、ファイルの最初に全てのコンポーネントがその種類ごとに表示されます。インポートされたファイルオプションにチェックが入っている場合、インポートされたファイル内にあるコンポーネントがスキーマドキュメンテーションに含められます。
- スキーマペインでは、ファイル内にあるスキーマのレポートを生成するか選択することができます。スキーマのレポートを生成するよう選択した場合、更に () スキーマのドキュメンテーションを独立したファイルとして生成するか、またはメイ

ンドキュメンテーションファイル内部に含めるか、そして (i) スキーマ全体のレポートを生成するのか、グローバル要素、単純型、複合型だけを生成するのか選択することができます。

- 詳細ペインでは、各コンポーネントに含める対象が表示されます。ドキュメンテーションに加えたい項目を選択してください。
- **[生成時にファイルを表示]** オプションにより、(HTML 出力の場合は) ブラウザービュー、(Microsoft Word の場合は) Microsoft Word、(RTF ファイルの場合は) `.rtf` ファイルに割り当てられたデフォルトアプリケーションで出力ファイルが開かれます。

12.1.9 WSDL 2.0 への変換

XMLSpy を使えば、既存の WSDL 1.1 ドキュメントを、簡単に WSDL 2.0 フォーマットへ変換することができます。以下のように、`TimeService.wsdl` を例にしてみましょう:

1. `TimeService.wsdl` をアクティブなファイルにします。このファイルは XMLSpy に同封されている `Examples` プロジェクトの `WSDL` エディター フォルダー以下にあります。
2. **[WSDL | WSDL 2.0 に変換]** コマンドを選択します。
3. 名前を付けて保存ダイアログにて、WSDL 2.0 ファイルに使用する名前 (例えば `TimeService20.wsdl`) を入力します。
4. 新たなファイルが作成、自動的に検証され、WSDL ビューにて表示されます。

12.2 SOAP

Altova ウェブサイト: SOAP デバッガー

このセクションでは以下の事柄について説明します：

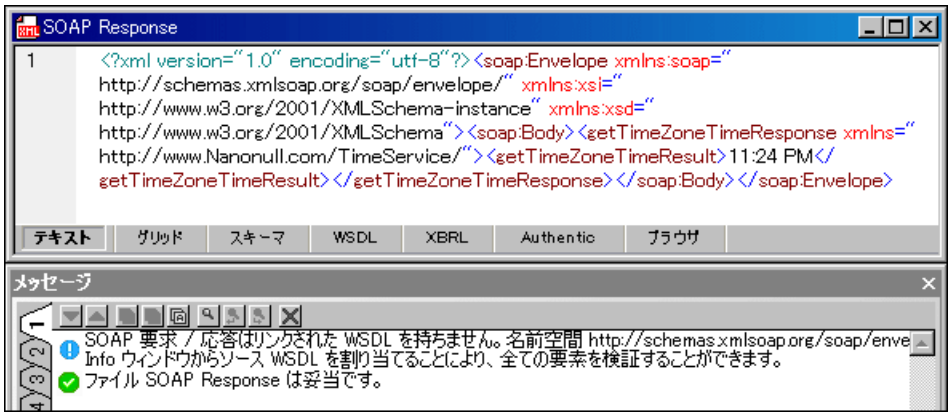
- WSDL に対するSOAP メッセージの検証。SOAP メッセージの検証を、SOAP 仕様だけではなく、WSDL 定義により参照されている任意のXML スキーマに対して行うことができます。
- SOAP デバッガーによるSOAP リクエストの送受信
- SOAP リクエストの送受信にブレークポイントをセットする
- ウェブサービスに送信する前に、不正なSOAP リクエストを編集する

12.2.1 SOAP 検証

SOAP 仕様だけではなく、対応するWSDL 定義により参照されている全てのXML スキーマに対しても、SOAP メッセージの検証を行うことができます。

SOAP ルールに対してのみの検証

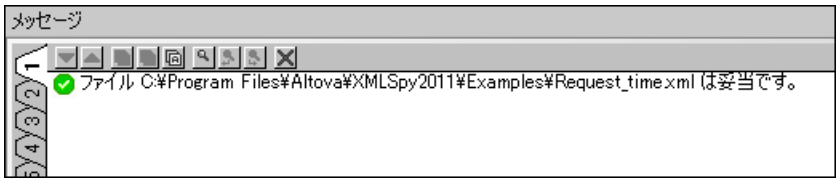
SOAP メッセージの検証を行うには、SOAP メッセージファイルを開き **[F8]** キーを押下するか、メニューコマンドから **[XML | 検証]** を選択します（以下のスクリーンショットを参照）。SOAP メッセージファイルにWSDL ファイルがリンクされていない場合、SOAP メッセージのルールに従い、SOAP メッセージの検証が行われます。これらのルールに準拠していれば、ファイルは妥当ということになります（以下のスクリーンショットのメッセージウィンドウを御覧ください）。



SOAP ルールならびにリンクされている WSDL に対する検証

リンクされたWSDL も含めてSOAP メッセージの検証を行うには、SOAP ファイルにWSDL ファイルをリンクする必要があります。この操作は情報ウィンドウのSOAP タブにて行うことができます（以下のスクリーンショットを参照）。*検証を行うWSDL* アイテムの右にあるボタンをクリックして、**[検証を行うWSDL を選択]** を選択します。表示されるダイアログにて目的のWSDL ファイルを選択し、**「OK」** をクリックします。入力されたWSDL ファイルは情報ウィンドウに表示され、SOAP メッセージファイルがリンクされます。

[F8] を押下（または**メニューコマンド**から**[XML | 検証]** を選択）することで、SOAP メッセージが、SOAP メッセージのルールだけでなく、リンクされたWSDL ファイルのルールに対しても検証されます。



両方のルールセットに従ったものであれば、ファイルは妥当なものとなります（上のスクリーンショットを参照）。

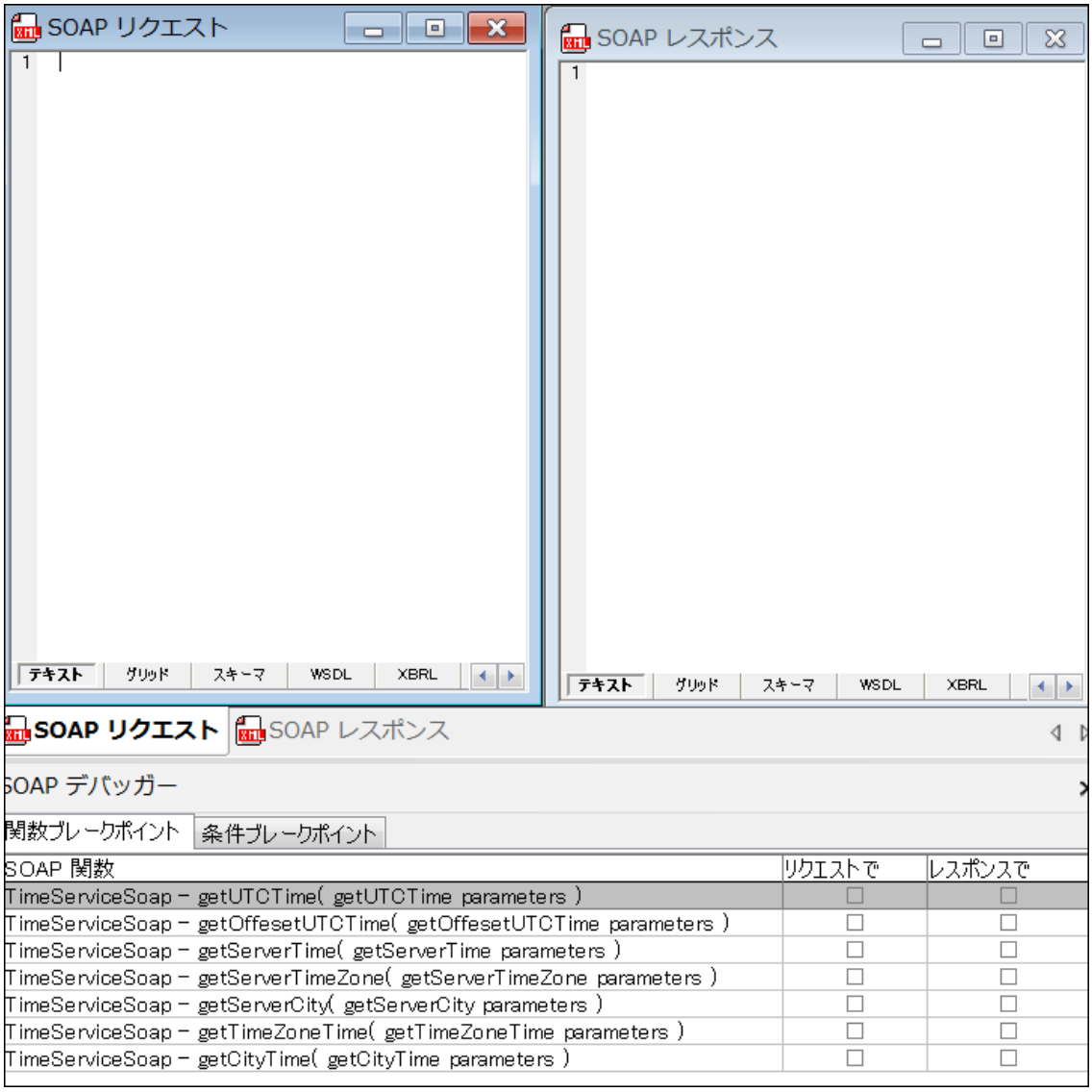
メモ WSDL ファイルからXMLSpy のSOAP リクエスト作成機能（**[SOAP | SOAP リクエストの作成]**）を使用して、SOAP リクエストが作成された場合、SOAP タブが情報ウィンドウに表示されます。（SOAP リクエストがXMLSpy にて作成されなかったため）SOAP タブが情報ウィンドウに表示されていない場合、SOAP リクエストファイルを保存することで、SOAP タブが表示されます。

12.2.2 SOAP デバッガー

SOAP デバッガーにより、SOAP リクエストとSOAP レスポンスを閲覧ならびに分析することができます（以下のスクリーンショットを参照）。デバッガーはクライアントとWeb サービス間の**プロキシ サーバー**として動作します。

SOAP デバッガーでは以下の機能がサポートされます：

- SOAP リクエストとレスポンスのステップ実行
- SOAP リクエストとレスポンスの修正
- クライアントまたはサーバーへのリクエスト転送
- リクエストやレスポンス メッセージにおけるブレークポイントのセットや、XPath 条件式を使った条件付きブレークポイント



SOAP デバッガーは以下のように動作します：

- SOAP デバッガーセッションを開始する前に、SOAP デバッガーオプションをセットします。これらオプションにはコンピューターの IP アドレスや SOAP デバッガーのレイアウト/タイムアウトオプションなどが含まれます。
- SPS デバッガーを開く (セッションを開始する) ために、**[SOAP | SOAP デバッグセッション]** を選択します。この時点で、(i) SOAP 情報が収められている WSDL ファイルの場所と、(ii) ソースならびにターゲットポートの情報を入力する必要があります。
- SOAP デバッガーブレークポイントウィンドウにて、必要なブレークポイントをセットします。
- SOAP リクエストを作成し、SOAP デバッガーを実行するためのファイルを開きます。
- 結果を分析し、エラーがある場合にはそれらを修正します。
- SOAP デバッガーを閉じるには、**[SOAP | SOAP デバッグセッション]** コマンドを選択してください。

`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples` フォルダーに収められているファイル `DebuggerClient.htm` は、サンプルファイルとして使用されています。このサンプルファイルでは、ブラウザーウィンドウは、SOAP メッセージを送信また受信するクライアントアプリとして作動します。`http://www.nanonull.com/TimeService/TimeService.asmx?WSDL` にある、Nanonull Time Service サービスは、サービスサーバーです。

SOAP コミュニケーション処理

プロキシサーバー (SOAP デバッガー) が開始されると、SOAP のやり取りは以下のように行われます：

プロキシサーバーにより外から入ってくるクライアントリクエストのソケットポートを常に監視

- クライアントアプリケーションがプロキシサーバーへリクエストを送信します。
- ブレークポイントがヒットした場合、クライアントリクエストを修正することができます。
- プロキシサーバーリクエストデータがウェブサービスサーバーへ転送されます。

ウェブサービスサーバーはプロキシサーバーに対して応答 (レスポンス) を行ない、レスポンスデータをプロキシサーバーに返信

- ブレークポイントがヒットした場合、サーバレスポンスを修正することができます。
- プロキシサーバーレスポンスデータがクライアントアプリケーションへ転送されます。
- クライアントアプリケーションがプロキシサーバーからレスポンスデータを受信

ポート設定

SOAP デバッガーはクライアントのクエストをモニターするために 8080 ポートを使用します。このポート番号は、新たな SOAP デバッグセッションが開始されたときに変更することができます。このポートは、パーソナルファイアウォールや広告フィルターにより閉じられていることがあります。ポートが閉じられている場合、これらのプログラムを無効にするか、別のポートアドレスを割り振る必要があります。

SOAP デバッガーオプション

SOAP デバッガーオプションダイアログ (スクリーンショットを参照) によりコンピュータのアドレスと下にリストされる他のデバッガーオプションを指定することができます。**[SOAP | SOAP デバッガーオプション]** メニューコマンドを選択し、SPS デバッガーオプションダイアログを開きます

- *コンピューターのアドレス*: デバッガーが実行されるプロキシのサーバーのアドレス。プロキシサーバー上のデバッガーは、ネットワーク上のマシンのリクエストを受け、web サービスに送信します。デバッガーは、XMLSpy 内で作動するため、XMLSpy がインストールされているマシンがプロキシサーバーとして作動します。マシンの IP アドレスは、自動的に検出され、このフィールドに入力されます。IP アドレスが自動的に検出されない場合、このフィールドに IP アドレスを (`http` アドレスとして) 入力します。コンピューターの IP アドレスを検出するには、コマンドプロンプトウィンドウを開き、コマンド `ipconfig /all` を入力して、**「Enter」**を押します。
- *タイムアウト*: タイムアウトの値は、SOAP デバッガーがブレークポイントにて待機する時間として使用されます。デフォルトのタイムアウト値は 5 秒です。
- *入力ヘルパーを閉じる*; *プロジェクト/情報ウィンドウを閉じる*: このオプションによりインターフェイスの SOAP デバッガーウィンドウにより多くの表示領域を割り当てることができます。

デバッグ セッションの開始

SOAP デバッガー セッションは XMLSpy のアクティブなファイルでいつでも開始することができます。セッションは、SOAP リクエスト エントリー ポイント ファイルを開く前、または開いた後に開始することができます。SOAP デバッガー セッションを開始すると以下を入力するように求められます：

1. SOAP 情報を取得するために使用される WSDL ファイルの場所
2. 接続の設定

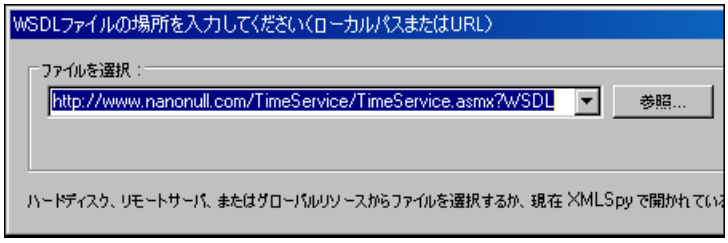
これらステップの詳細については以下を参照ください。

WSDL ファイル ロケーション

デバッガーセッションが開始すると、SOAP情報を取得するための WSDL ファイルの URL を入力するように求められます。 サンプル ファイル `DebuggerClient.html` は、以下の WSDL ファイル url を使用します：

```
http://www.nanonull.com/TimeService/TimeService.asmx?WSDL
```

メニュー コマンドの **[SOAP | SOAP デバッグ セッション]** を選択することで、SOAP デバッガーを開始します。WSDL ファイルの場所を入力するダイアログ ボックスが表示されます（以下のスクリーンショットを参照）。



WSDL ファイルの URL を入力し、**[OK]** をクリックします。ソースおよびターゲットダイアログが表示されます（以下のスクリーンショットを参照）。

接続の設定

ソースとターゲットダイアログの接続の設定ダイアログ（下のスクリーンショット）で以下のリストの設定を行うことができます。

- **ソース ポート**: やり取りに使用されるコンピューター上にあるポート番号を指定します。デフォルトでは 8080 となっています。この設定は SOAP デバッガーが開始されるごとに変更することができます。
- **ターゲット ポートとアドレス** これらの設定は直前のステップにて選択された WSDL ファイルから取得され、ダイアログへ自動的に入力されます。ターゲット ポートが 443 の場合、**[https による安全な接続]** チェック ボックスがデフォルトでチェックされます。80 以外のポート番号が入力された場合、手動でチェック ボックスにチェックを入れる必要があります。**[HTTPS 設定]** ボタンをクリックすることで、セキュアな接続のオプションを選択することができます（オプションの使用方法に関する詳細については、[SOAP リクエスト パラメーターの変更]のセクションを参照ください）。

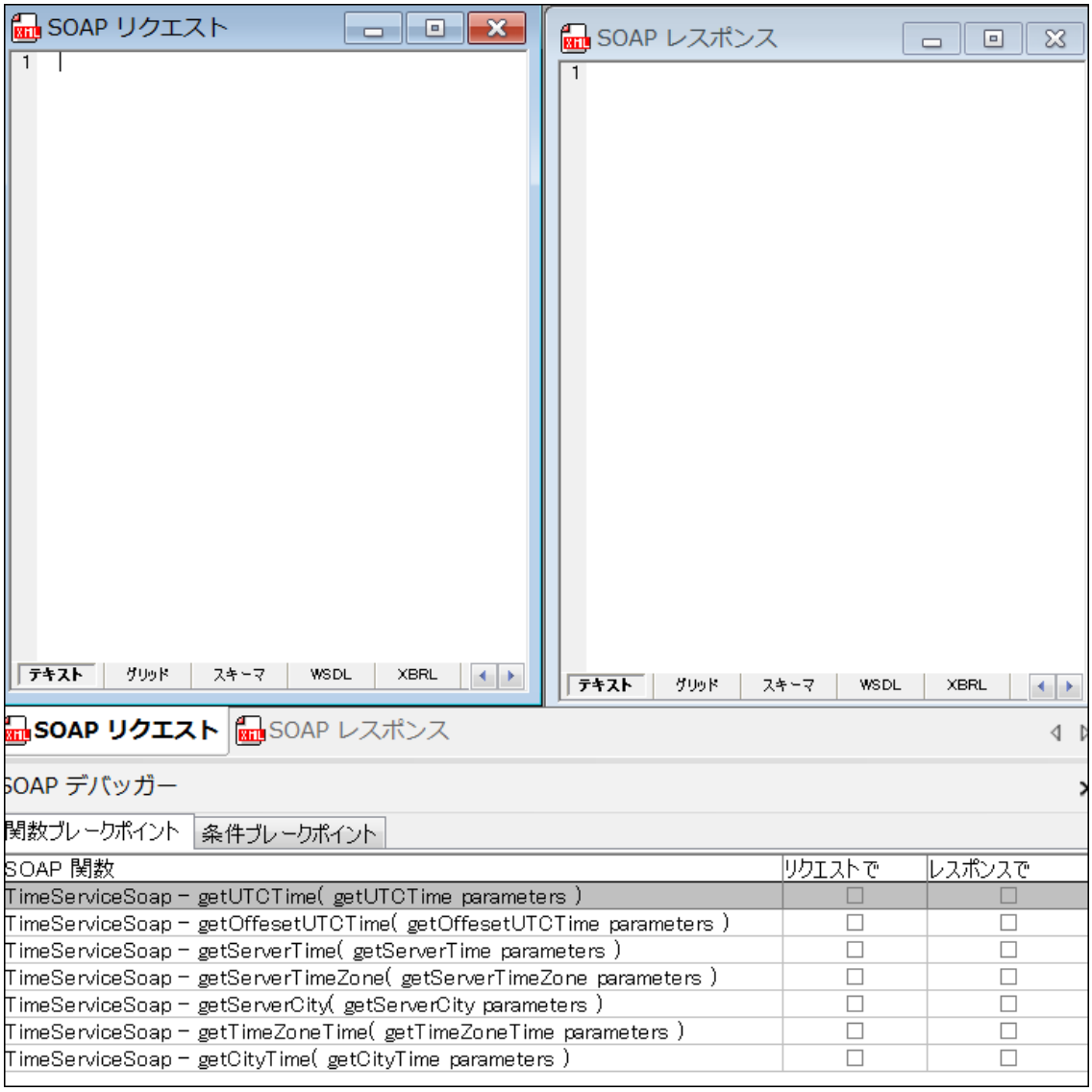
これらの設定を確認し、必要に応じて編集を行った後には、**[OK]** をクリックすることで SOAP デバッガー セッションが開始されます。SOAP デバッガーが開始された段階では、プロキシ サーバーはアクティブとなっていません (SOAP デバッガー ツールバーにあるプロキシ サーバー アイコン がグレー アウトされます)。SOAP ツールバーにある実行アイコンをクリックするか、メニュー コマンドの **[SOAP | 実行]** を選択してください。実際のデバッグについては、[デバッグ] のセクションを参照ください。

SOAP デバッガー レイアウト

SOAP デバッガー ウィンドウには 3 つのウィンドウが含まれます (以下のスクリーンショットを参照):

- SOAP リクエスト ウィンドウ
- SOAP レスポンス ウィンドウ
- SOAP デバッガー、ブレークポイント設定ウィンドウ

デフォルトでは XMLSpy インターフェイスの上部にリクエストとレスポンスのウィンドウが開かれ、ブレークポイント設定ウィンドウが下部に表示されます。デフォルトのレイアウトを以下のスクリーンショットに示します。



XMLSpy サイドバー ウィンドウ (プロジェクト、Info、入力ヘルパー ウィンドウ) を隠すことで、SOAP デバッガー ウィンドウへ更なる表示領域を割り当てることができます。**[SOAP | SOAP デバッガー オプション]** メニュー コマンドからアクセスすることのできるSOAP デバッガー オプション ダイアログでは、これらウィンドウの表示を指定することができます。

信頼された証明書について

Altova 製品はセキュアな Web サーバーの証明書へアクセス、管理するために、Internet Explorer (IE) を使用します。Web サーバーの証明書をIE へインストールすることにより、警告が表示されたり処理を中断することなく、その Web サーバーへアクセスすることができるようになります。以下の手順を踏んでください:

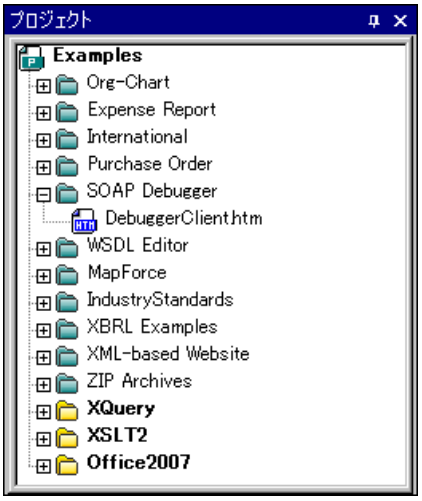
- Internet Explorer 9 以降で、安全な (セキュアな) Web サイトを開きます。
- **[ファイル | プロパティ]** を選択し、[証明書] ボタンをクリックします。
- **[証明書のインストール]** をクリックして証明書インポート ウィザードを表示します (このウィザードへは **[ツール | インターネット オプション | コンテンツ | 証明書 | インポート]** からもアクセスすることができます)。
- 証明書が信頼されたルート証明書ストアに配置され、手動で参照することができるようになります。
- ウィザードを完了し、**[OK]** をクリックして証明書とプロパティダイアログを閉じます。Internet Explorer を再起

動する必要があることもあります。

SOAP リクエスト エントリー ポイント

サンプル プロジェクトにあるHTML ファイル (DebuggerClient.htm) にはSOAP デバッガーのデモを行うためのスクリプトが含まれています。このファイルによりSOAP リクエストをWeb サービスへ送信し、Web サービスからのレスポンスを表示することができます。以下の操作によりXMLSpy でこのファイルを開くことができます :

1. メニュー コマンドから**[プロジェクト | プロジェクトを開く]** を選択します。
2. C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples フォルダーを参照し、Examples.spp ファイルを選択します。プロジェクト ウィンドウにサンプル プロジェクトがロードされます (以下のスクリーンショットを参照)。

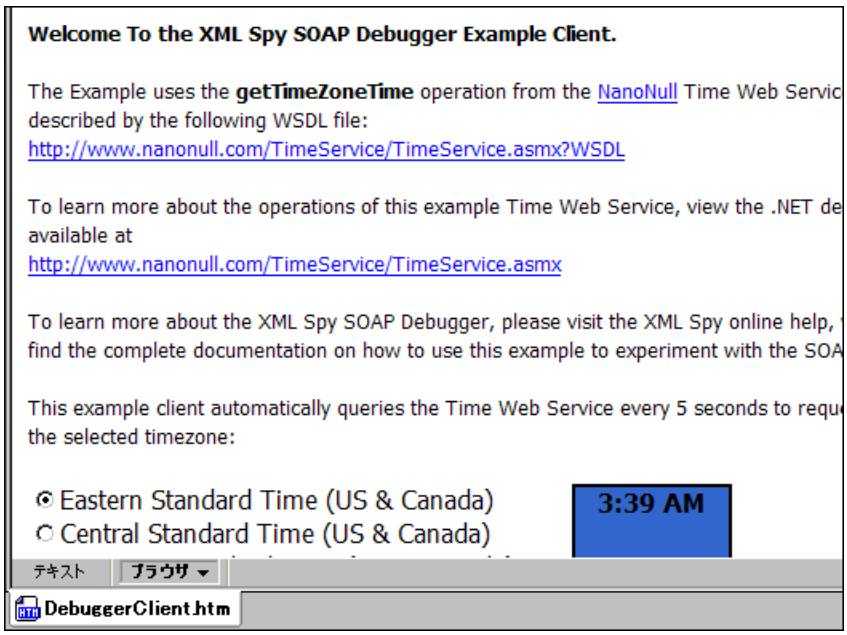


3. SOAP Debugger の+サインをクリックすることで、そのコンテンツを確認することができます。DebuggerClient.htm をダブルクリックすることで、そのファイルがXMLSpy で開かれます。

メモ: メニューの**[ファイル | ファイルを開く]** からもこのファイル (DebuggerClient.htm) を開くことができます。C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples フォルダーに配置されます。

サンプル ファイル

DebuggerClient.htm はXMLSpy のブラウザー ビューにて以下のように表示されます。ラジオ ボタンが選択されるとSOAP リクエストがNanonull Time Web Service に送信されます。Web サービスからのレスポンスがラジオ ボタンの右にある色付きのボックスに表示されます。



Eastern Standard Time (東部標準時) リクエストへのレスポンスが、上のスクリーンショットにある青のボックスに表示されます (6:23)。GMT ラジオボタンをクリックしてください。GMT の値が表示される代わりに、エラーメッセージが表示され、ボックスが赤色で表示されます (以下のスクリーンショットを参照)。

SOAP デバッガーを使用することで、SOAP メッセージを分析し、エラーの識別を行うことができます。以下にある 3 つのセクションでは、(i) ブレークポイントのセット、(ii) `DebuggerClient.htm` とともに SOAP デバッガーの実行、そして (ii) SOAP デバッガー出力の分析とエラーの識別を行う方法について記されます。

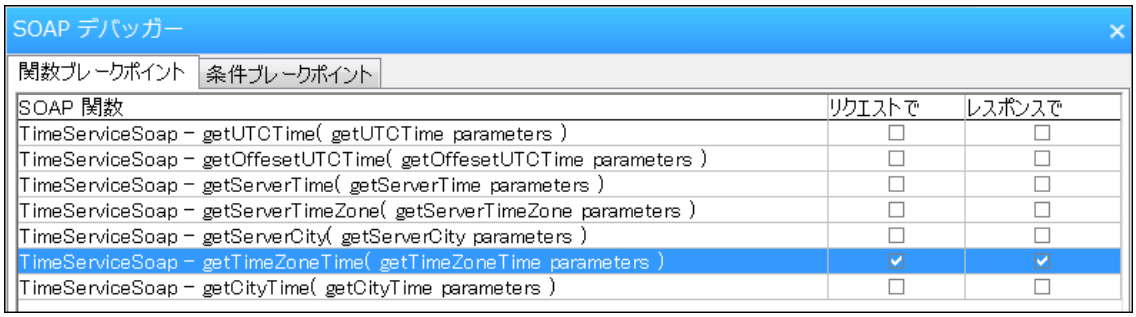
ブレークポイントのセット

デバッグを行う前に、SOAP デバッガーにてブレークポイントをセットする必要があります。デバッグを開始すると、ブレークポイントへ処理が移った際に SOAP デバッガーによりリクエストとレスポンスが表示されます。

SOAP デバッガーでは、SOAP デバッガーが開始された時に選択された WSDL ファイルから取得された情報に従って、(関数ブレークポイントと条件ブレークポイントに) ブレークポイントが表示されます。これらのブレークポイントは、WSDL ファイルから生成することのできる SOAP リクエストに関連付けられます。各 SOAP リクエストに対して対応する列のチェックボックスにチェックを入れることで、リクエストまたはレスポンスにブレークポイントをセットすることができます (以下のスクリーンショットを参照)。

この例では、以下のファイルを使用します :

- SOAP リクエスト エントリー ポイントとしての `DebuggerClient.htm`
- SOAP デバッガーを開始した際に選択された WSDL ファイルの `http://www.nanonull.com/TimeService/TimeService.asmx?WSDL`



`DebuggerClient.htm` からリクエストされたWeb サービスでは、選択されたタイムゾーンの時間を取得するために `getTimeZoneTime` というメソッドが使用されます。SOAP デバッガーでは、選択されたWSDL ファイルから生成することのできるSOAP リクエストがブレークポイントとして表示されます。ここでは `getTimeZoneTime` メソッドの[リクエストで]と[レスポンスで]の両方にブレークポイントをセットします（上のスクリーンショットを参照）。これにより、エラーを含むWeb サービスへのリクエストやWeb サービスからのレスポンスを分析することができます。

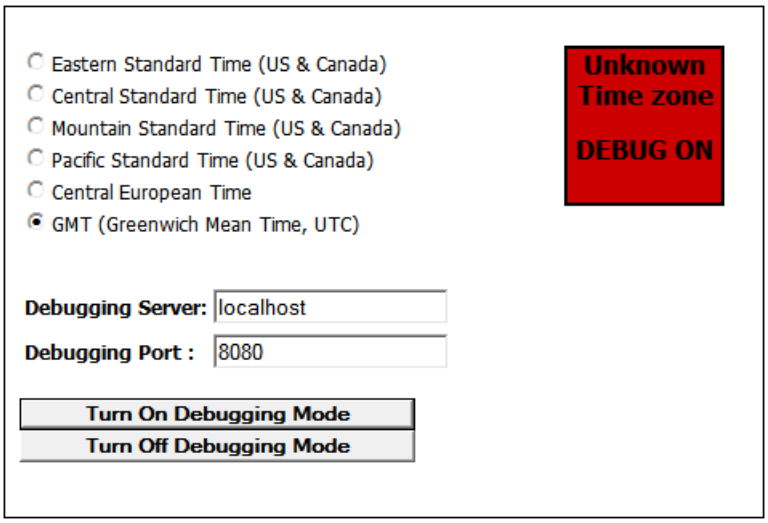
ブレークポイントに関する詳細な情報については、[ブレークポイントに関する更なる情報]のセクションを参照ください。

デバッグ

この例では、以下のファイルを使用します：

- SOAP リクエスト エントリー ポイントとしての `DebuggerClient.htm`
- SOAP デバッガーを開始した際に選択されたWSDL ファイルの `http://www.nanonull.com/TimeService/TimeService.asmx?WSDL`

ブレークポイントをセットした後に**[実行]** アイコン をクリック（またはメニューの**[SOAP | 実行]** コマンドを選択）します。`DebuggerClient.htm` のタブをクリックし、SOAP エントリー ポイント ファイルを表示してください。GMT オプションが選択されていることを確認し、**[Turn on Debugging Mode]** をクリックしてください（以下のスクリーンショットを参照）。



これにより**[Debug On]** メッセージが表示され、SOAP リクエストがSOAP デバッガーへ送信されます。デバッガーの結果がSOAP リクエストならびにSOAP レスポンス ウィンドウに表示されます。詳細については次のセクションの[結果とエラーの分析]を参照してください。

SOAP デバッガー コントロール

SOAP デバッガー ツールバーにはSOAP デバッガーを操作するためのアイコンが含まれています (以下のスクリーンショットを参照)。

これらのアイコンは、左から以下の様な意味を持ちます:

- **実行**: デバッグを開始します。
- **ステップ実行**: リクエスト-レスポンスの処理をステップ実行し、ブレークポイントで停止します。
- **次のリクエストでブレーク**: 次のSOAP リクエストで停止します。
- **次のレスポンスでブレーク**: Web サービスから得られた次のレスポンスで停止します。
- **プロキシ サーバーを停止**: デバッグを停止します。この操作はSOAP デバッグ セッションを終了するのとは異なるということにご注意してください。SOAP デバッグ セッションを終了/開始するには、メニューから**[SOAP | SOAP デバッグ セッション]** コマンドを選択してください。

結果とエラーの分析

SOAP デバッガーの結果はSOAP リクエストとSOAP レスポンスのウィンドウに表示されます。ブレークポイントは、デフォルトでSOAP デバッガー ウィンドウの下部に配置されているSOAP デバッガー ブレークポイント ペインにてセットされます。

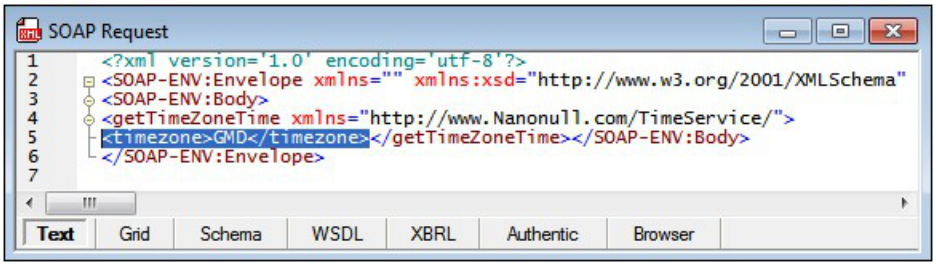
セットされたブレークポイントに従い、対応するウィンドウ (SOAP リクエストまたはSOAP レスポンス) に結果が表示されます。

この例では、以下のファイルを使用します:

- SOAP リクエスト エントリー ポイントとしての `DebuggerClient.htm`
- SOAP デバッガーを開始した際に選択されたWSDL ファイルの `http://www.nanonull.com/TimeService/TimeService.asmx?WSDL`

エラーの検知とフィックスのテスト

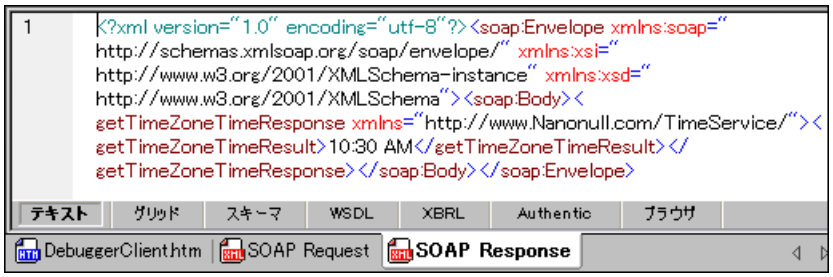
直前のセクション[デバッグ]に記述された方法によりデバッグは開始されます。GMT を選択した際のSOAP リクエストがデバッガーのSOAP リクエスト ウィンドウのテキスト ビューに表示され、編集を行うことができます。



`timezone` 要素に注目すると、値が`GMD` となっています。これは間違った値であるため、`GMT` に変更します。`timezone` 要素をダブルクリックして、要素のコンテンツを`GMT` に変更してください。

問題が解決されたか確認するため、SOAP デバッガー ツールバーにある**実行**アイコンをクリック (またはメニューの**[SOAP | 実行]** コマンドを選択) して、正しいリクエストをWeb サービスへ送信します。数秒後、SOAP リクエストに対するWeb サービスからのレスポンスがSOAP レスポンス ウィンドウに表示されます。**[ビュー | ワード ラップ]** を選択することで、

SOAP レスポンス全体がウィンドウに表示されます (以下のスクリーンショットを参照)。

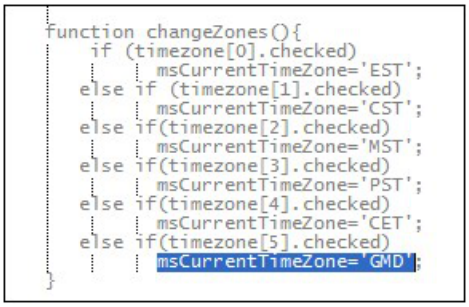


DebuggerClient.htm タブをクリックして、SOAP デバッガー ツールバーにある**[実行]** アイコンをクリックしてください。エラー メッセージが消え、正しい GMT の時間が表示されます (以下のスクリーンショットを参照)。

メニューの**[SOAP | SOAP デバッガー セッション]** コマンドを選択することで、SOAP デバッガー セッションが終了します。

エラーの修正

これで、(GMT ではなく) GMD という無効な値が SOAP リクエストにて生成されているということが分かりました。SOAP リクエスト エントリー ポイント ファイルを開き、(**Ctrl**」 **+F** または**[編集 | 検索]** により) GMD という単語を検索すると、以下のスクリーンショットに示されているように、該当するコードを確認することができます。

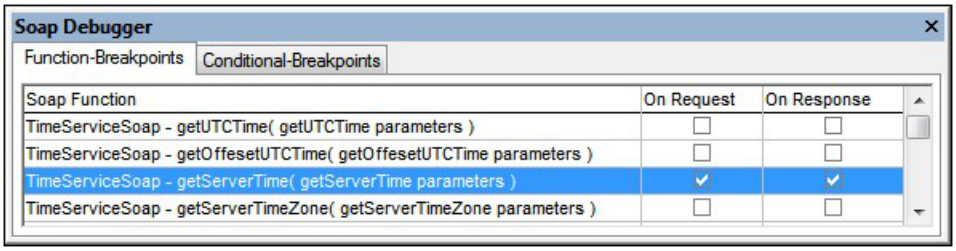


```
function changeZones(){
    if (timezone[0].checked)
        msCurrentTimeZone='EST';
    else if (timezone[1].checked)
        msCurrentTimeZone='CST';
    else if(timezone[2].checked)
        msCurrentTimeZone='MST';
    else if(timezone[3].checked)
        msCurrentTimeZone='PST';
    else if(timezone[4].checked)
        msCurrentTimeZone='CET';
    else if(timezone[5].checked)
        msCurrentTimeZone='GMD';
}
```

このエラーが修正された状態で GMT ラジオ ボタンが選択されると、未知の Timezone エラーの代わりに正しい GMT の時間が表示されます。

ブレークポイントに関する更なる情報

SOAP デバッガー ウィンドウではブレークポイントをセットならびに削除することができます。ウィンドウは 2 つのタブから構成されます (以下のスクリーンショットを参照)。



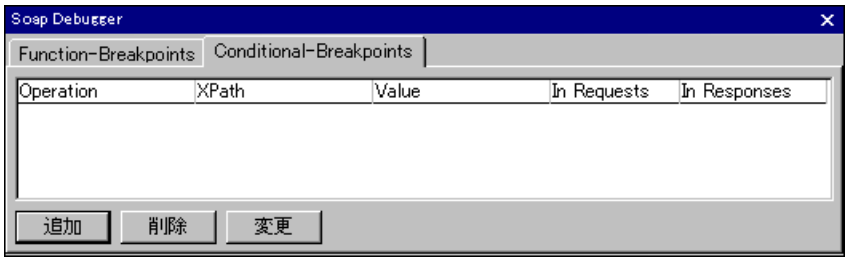
関数ブレークポイント タブ

関数ブレークポイント タブではSOAP リクエストやレスポンスのメソッドにブレークポイントをセットすることができます。デバッガーによりブレークポイントとしてセットされたメソッドの呼び出し行がハイライトされます。クライアントが通信を行うデータ パケットが分析され、WSDL から得られた関数にマッチされます。あるメソッドへブレークポイントがセットされている場合、そこで SOAP デバッガーが停止します。ツールバーのボタンはこの時点で選択可能な状態になっています。

データがSOAP リクエストまたはSOAP レスポンス ドキュメント ウィンドウに表示されます。この時点でSOAP ウィンドウに表示されたSOAP ドキュメントを修正することができます。ツールバーにある**[サーバーを停止]** 以外のアイコンをクリックすることで、データが送信されます。

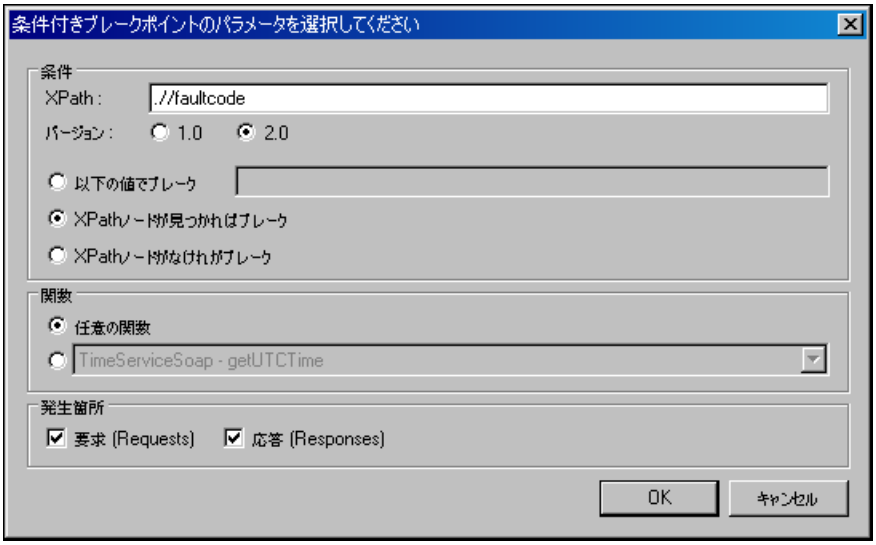
条件ブレークポイント タブ

条件ブレークポイント タブではXPath 条件式によりブレークポイントを定義することができます（以下のスクリーンショットを参照）。SOAP リクエストによりエラーが生じる場合、SOAP レスポンスには `faultocde` 要素が含まれます。そこで、`faultcode` 要素が現れるとデバッガーが停止するようなブレークポイントを作成することにします。



以下の操作により条件ブレークポイントをセットすることができます：

1. 条件ブレークポイント タブをクリックして、**[追加]** ボタンをクリックします。以下に示されるダイアログが表示されます。

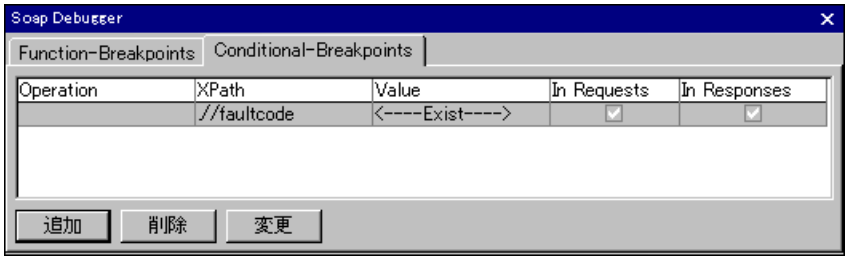


2. XPath フィールドにXPath 条件式 (例えば .//faultcode) を入力します。
3. XPath のバージョンを (1.0 または 2.0)選択し、**[XPath ノードが見つかればブレーク]** ラジオボタンを選択します。
4. **「OK」**をクリックして設定を完了します。SOAP リクエストまたはレスポンスに .//faultcode 要素が出現したところでSOAP デバッガーが停止されるようになります。

このダイアログで使用可能なオプションについて以下に記します :

- **XPath 条件式フィールド**: XPath 条件式を入力してください。XPath を入力することで、ラジオボタンのオプションが利用可能になります。
- **バージョン**: XPath 条件式で使用されるXPath のバージョンを選択します。
- **ブレークラジオボタン**: 選択されたオプションが発生した時にデバッガーが停止します。利用できるオプションには、(i) ターゲットのXPath ノードがフィールドで入力された値にマッチした時にブレーク (i) 指定されたXPath ノードがSOAP リクエストまたはレスポンス内に存在する時にブレーク または (ii) 指定されたXPath ノードがSOAP リクエストまたはレスポンスに存在しない場合ブレークがあります。
- **リクエストとレスポンス**: ダイアログのオプションがSOAP リクエストまたはレスポンス (またはその両方) に対して適用されるか指定します。
- **関数**: 定義した条件ルールがすべてのメソッド関数に対して適用されるのか [任意の関数] ラジオボタン)、または入力したメソッド関数に対してだけ適用するのかを選択できます。

上に示されるダイアログにて条件が定義されると、その条件ブレークポイントが条件ブレークポイント タブに表示されます。



タブに表示される列について以下で説明します :

- **オペレーション**: 検索されるメソッド関数が含まれます。**[任意の関数]** ラジオボタンが選択された場合、空のフィールドが表示されます。特定のメソッド関数が選択された場合、そのメソッド関数が表示されます。
- **XPath**: 定義したXPath 条件式が含まれます。
- **値**: ノードをチェックするためのXPath 値が含まれます。[**以下の値でブレーク**] が選択された場合、指定された文

字列が表示されます。**[XPath ノードが見つかればブレーク]** の場合は `<!--Exist-->` が、**[XPath ノードがなければブレーク]** が選択されている場合には `<!--Missing-->` が表示されます。

- **リクエスト内**と**レスポンス内**: 条件がチェックされる場所がチェックボックスにより示されます。列にあるチェックボックスを直接クリックすることでも、設定を変更することができます。

条件ブレークポイントを編集するには、タブに表示された行をダブルクリックするか、**[変更]** ボタンをクリックします。条件ブレークポイントを削除するには、その行が選択されている状態で、**[削除]** ボタンをクリックしてください。

13 XBRL

Altova Web サイト: XBRL タクソノミエディター、XBRL バリデーター

XMLSpy の XBRL ビューは XBRL タクソノミのグラフィカルな概要を表示し、タクソノミのインテリジェントな編集機能を搭載した XBRL タクソノミエディターです。このセクションでは、XBRL ビューでタクソノミの作成や編集を行う方法を説明します。

このセクションは以下のように構成されます:

- 基本のプロシージャは、最も必要なコンポーネントを含むタクソノミの作成方法を説明します。
- XBRL フォーミュラエディター は、XBRL ビューを使用して XBRL フォーミュラと作業する方法を説明します。
- XBRL テーブル定義 エディターは、テーブル構造、XBRL テーブルの定義のためのエディターの使用方法、およびテーブル レイアウトプレビューの使用方法を説明します。また、テーブルセットと共に使用されるテーブルパラメーターを含む、テーブルパラメーターの使用方法についても説明しています。
- XBRL 内の検索は、XMLSpy の XBRL-固有のパラフルな検索機能について説明しています。

より詳細な情報は、編集ビュー | XBRL ビューと XBRL メニューのコマンドに関する記述を参照ください。例えば、XBRL ビューにあるような) タクソノミのドキュメンテーションの生成に関しては、ユーザーリファレンス | XBRL メニュー | ドキュメンテーションの生成を参照ください。

XBRL ビューで表示される XBRL ファイルに対する XML 署名は独立したファイルとして作成することができます。署名の詳しい使用方法については、XML 署名のセクションを参照ください。

XMLSpy 内の US-GAAP と他のタクソノミへのサポート
XMLSpy は、以下のタクソノミをサポートします:

- US-GAAP 1.0/2009/2011/2012/2013/2014/2015/2016
- IFRS

US-GAAP (2013 と 2014) の最新バージョンは、XMLSpy とともにインストールされています。古い US-GAAP タクソノミを含む、追加タクソノミを、Altova Web サイトから無料でダウンロードすることのできる、タクソノミインストーラーを介して、使用することができます。

13.1 基本のプロシージャ

基本のプロシージャのセクションでは、最も需要なコンポーネントを含むタクソノミの作成方法を説明します。このセクションは以下のように構成されます：

- 最初に新たに作成されたタクソノミと既存のタクソノミの違いと、その重要性について簡単に説明します。その後、XBRL タクソノミを構成するファイルと、それらがどのようにXBRL ビューに表示されるかの説明を行ないます。
- .タクソノミの新規作成セクションからは、XBRL ビューでタクソノミを構築する方法を順を追って解説します。この中で新規タクソノミの作成ウィザードの説明も行ないます。各セクションの終わりには、XBRL ビューを使って、オリジナルのタクソノミを作成するための例題が与えられます。例題は、その前のセクションの例題により作成されたタクソノミを元に構築され、そのセクションで与えられた情報を使った演習やテストとして使用することができます。

13.1.1 タクソノミ:新規と既存

XMLSpy のXBRL ビューでは既存のタクソノミの編集や新規タクソノミの作成を行うことができます。

- *既存のタクソノミ*:既存のタクソノミには2種類あります：(i) 編集することのできない標準タクソノミと、(ii) あなた自身またはサードパーティにより作成された標準ではないタクソノミで、編集可能。
- *新規タクソノミ*:XMLSpy では新規タクソノミの作成を行うことができます。作成方法には2種類あります：(i) 何も無い状態からのタクソノミの作成と、(ii) 標準タクソノミを拡張したタクソノミ。

XBRL ビューでは上に挙げられた両方の種類のタクソノミを閲覧、編集することができます。(タクソノミの拡張のために) 標準タクソノミを作成中のタクソノミにインポートするような場合、インポートされたタクソノミを編集することはできません。インポートされたタクソノミで、編集することが許されていないものはグレーで表示されます。

タクソノミパッケージ

XBRL タクソノミパッケージはタクソノミのオフラインのコピーを含む圧縮されたアーカイブです。タクソノミパッケージは、オフラインのタクソノミのファイルの場所へのURI を再マップするカタログXML ファイルを含んでいます。このため、オフラインのタクソノミがアプリケーションで使用できるようになります。タクソノミパッケージがどのように構成されビルドされるかに関するルールは、XBRL.org のタクソノミパッケージ勧告 で説明されています。

タクソノミパッケージをダウンロードし、XMLSpy に登録すると、XMLSpy は、(スキーマなど) パッケージのオフラインのリソースを検証の際に使用することができます。パッケージの登録は、**[ツール] | [オプション] | [タクソノミパッケージ]** ペインで行うことができ、手順は、このペインの説明で説明されています。

新規タクソノミの作成方法

新規タクソノミは通常既存のタクソノミの上に構築されます。新規タクソノミでは新しい要素が追加され、これら新しい要素とインポートされた要素間のリレーションシップが作成されます。以下では新規タクソノミに求められる一般的な要件と、その作成方法の概要を説明します：

1. 新規タクソノミは、他のタクソノミと区別するために、自身の名前空間内で作成されなければなりません。新規タクソノミが既存のタクソノミを拡張したものであれば、既存のタクソノミを新規タクソノミにインポートする必要があります。
2. 新規タクソノミでは、新しいコンセプト (要素) が定義されます。
3. リレーションシップファイル (またはリンクベース) には定義、Presentation、Calculation、ラベル、そして新規タクソノミへの参照リレーションシップが含まれます。
4. 新規タクソノミへのリレーションシップは、テンプレートなと無く、最初から作成する必要があります。

上の記述にあるように、ここで使用されているタクソノミという用語は、コンセプト定義ファイルやリレーションシップファイルといった複数のファイルからなるタクソノミ全体のことを指しています。タクソノミを構成する様々なファイルの詳細については タクソノミドキュメントファイル を参照ください。

XBRL ビューの使用

以下に続くセクションでは、XBRL ビューの機能を使ったタクソノミの作成や編集方法について説明します。新規タクソノミの作成以降のセクションでは、各セクションの最後に独自のタクソノミを作成するための例題が与えられています。各セクションにある例題は、それ以前のセクションにて構築されたものをベースにしています。リレーションシップの作成：パート1に辿りつく頃には、XBRL ビューの操作方法にも慣れ親しみ、自信を持って使用することができるようになっているでしょう。

ここで作成するタクソノミは、最終的に`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017Examples\XBRLExamples\Nanonull` フォルダー以下に格納されている XMLSpy に付属のファイル `Nanonull.xsd` のようになります。メインタクソノミファイルは常に `.xsd` ファイル拡張子を持ち、`.xbrl` ファイル拡張子を持つのは XBRL インスタンスファイルで、タクソノミファイルではないことに注意してください。

13.1.2 タクソノミファイルの概要

注意深くデザインされた XBRL タクソノミは、タクソノミコンセプトをタクソノミリレーションシップとは別のファイルに保管されます。このファイルのことをメインタクソノミ またはコンセプト定義ファイルと呼びます。更に、リレーションシップにも複数の種類があるため、これらリレーションシップもその種類ごとに個別のファイルで管理されます。以下のテーブルでは、タクソノミドキュメントを一般的に構成するファイルの種類を示します。

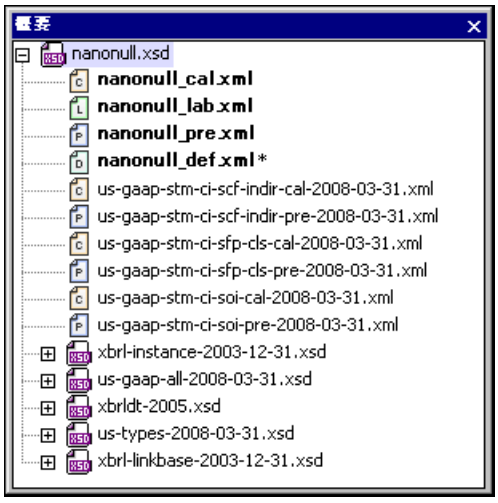
XBRL ファイル	説明	ファイルの種類
コンセプト	各コンセプトは XML スキーマ `element` 要素内で定義されます。	XML スキーマファイル (`.xsd`) コンセプト定義ファイル
定義 リレーションシップ	`definitionLink` 要素にはコンセプトリレーションシップの全てのロケータと定義 アークが含まれます。	XML ファイル (`.xml`)
計算 リレーションシップ	`calculationLink` 要素には全てのロケータと計算アークが含まれます。	XML ファイル (`.xml`)
プレゼンテーション リレーションシップ	`presentationLink` 要素には全てのロケータとプレゼンテーション アークが含まれます。	XML ファイル (`.xml`)
ラベル	`labelLink` 要素には全てのロケータとラベル アーク そしてラベルが含まれます。	XML ファイル (`.xml`)
参照	`referenceLink` 要素には全てのロケータと 参照アーク そして参照リソースが含まれます。	XML ファイル (`.xml`)

リレーションシップファイルの場所はコンセプト定義ファイル (`.xsd` ファイル) 内の `schema/annotation/appinfo` 要素により定義されます：

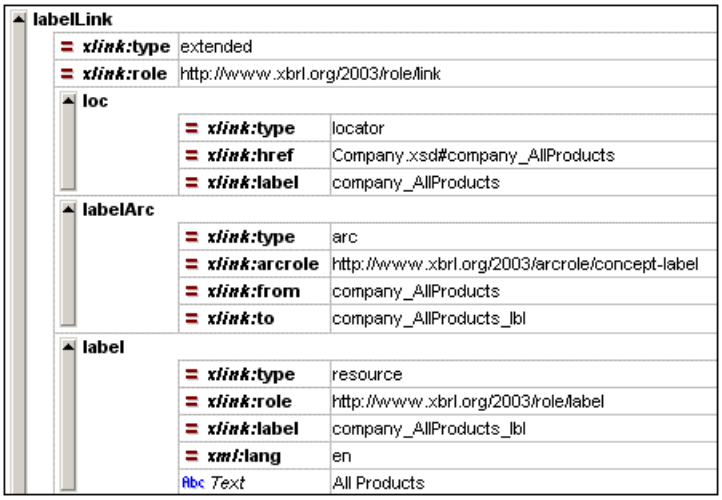
```
<xsd:annotation>
  <xsd:appinfo>
    <link:linkbaseRef xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
      xlink:href="NanonullLabels.xml" xlink:type="simple"
      xlink:role="http://www.xbrl.org/2003/role/labelLinkbaseRef" />
    <link:linkbaseRef xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
      xlink:href="NanonullDefinitions.xml" xlink:type="simple"
      xlink:role="http://www.xbrl.org/2003/role/definitionLinkbaseRef" />
    <link:linkbaseRef xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
```

```
      xlink:href="NanonullPresentations.xml" xlink:type="simple"
      xlink:role="http://www.xbrl.org/2003/role/presentationLinkbaseRef" />
    <link:linkbaseRef xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
      xlink:href="NanonullCalculations.xml" xlink:type="simple"
      xlink:role="http://www.xbrl.org/2003/role/calculationLinkbaseRef" />
    <link:linkbaseRef xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
      xlink:href="NanonullReferences.xml" xlink:type="simple"
      xlink:role="http://www.xbrl.org/2003/role/referenceLinkbaseRef" />
  </xsd:appinfo>
</xsd:annotation>
```

コンセプト定義ファイル（xsd ファイル）がXBRL ビューにて開かれると、概要入力ヘルパーにて様々なタクソノミファイルがツリー構造で表示されます（以下のスクリーンショットを参照）。



上のスクリーンショットにおいて、ファイル名の左にあるアイコンがスキーマを表していることにご注目してください。XML スキーマ（.xsd）ファイルはXSD アイコンにより示されます。リレーションシップファイルは、リレーションシップの種類を表す文字が記された色付のアイコンで表示されます。例えば、 アイコンは 定義 リレーションシップファイルであることを表し、 アイコンは プレゼンテーション リレーションシップファイルであることを表しています。これらファイルをダブルクリックすると、ファイルがXMLSpy のグリッドビューで開かれます（以下のスクリーンショットを参照してください）。



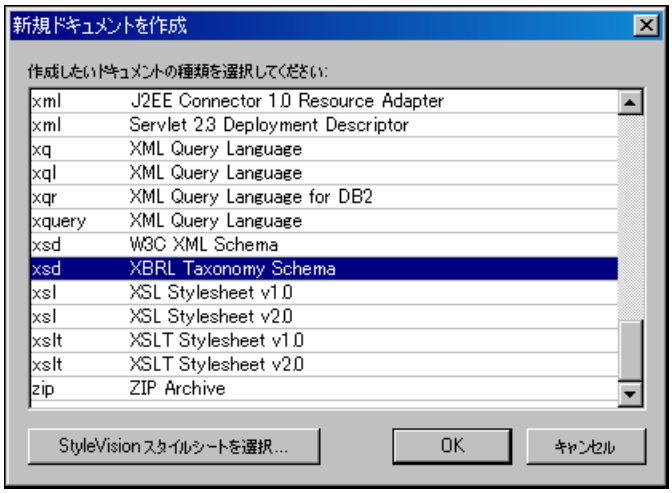
13.1.3 新規タクソノミの作成

通常新規タクソノミは1つまたは複数の標準タクソノミを拡張する形で作成されます。標準タクソノミ または既存のタクソノミを元に新規タクソノミを構築するのであれば、既存のタクソノミをインポートする必要があります。新規タクソノミは何も無い状態から作成することもできます。XMLSpy のXBRL ビューでは、新規タクソノミウィザードを使ってUS-GAAP やIFRS タクソノミを簡単にインポートすることができます。タクソノミはXBRL ビューのグラフィカルなインターフェースを使って編集することができます。

新規タクソノミを作成するにあたり最初にすることは、XML スキーマファイル（.xsd）であるコンセプト定義ファイルを作成することです。コンセプト定義の他にも、このファイルには新規タクソノミの名前空間定義、宣言が含まれており、インポートするタクソノミやタクソノミのリレーションシップファイルの場所を特定し、インポートされたタクソノミの名前空間や、その他使用される名前空間の宣言を行ないます。

コンセプト定義ファイルの作成

新規 XBRL タクソノミを作成するには、メニューコマンドから **「ファイル | 新規作成」** を選択します。新規ドキュメントを作成ウィンドウが表示されます。



.xsd XBRL Taxonomy Schema を選択し、 **「OK」** で確定します。この操作により、タクソノミウィザードの最初の画面が表示されます。

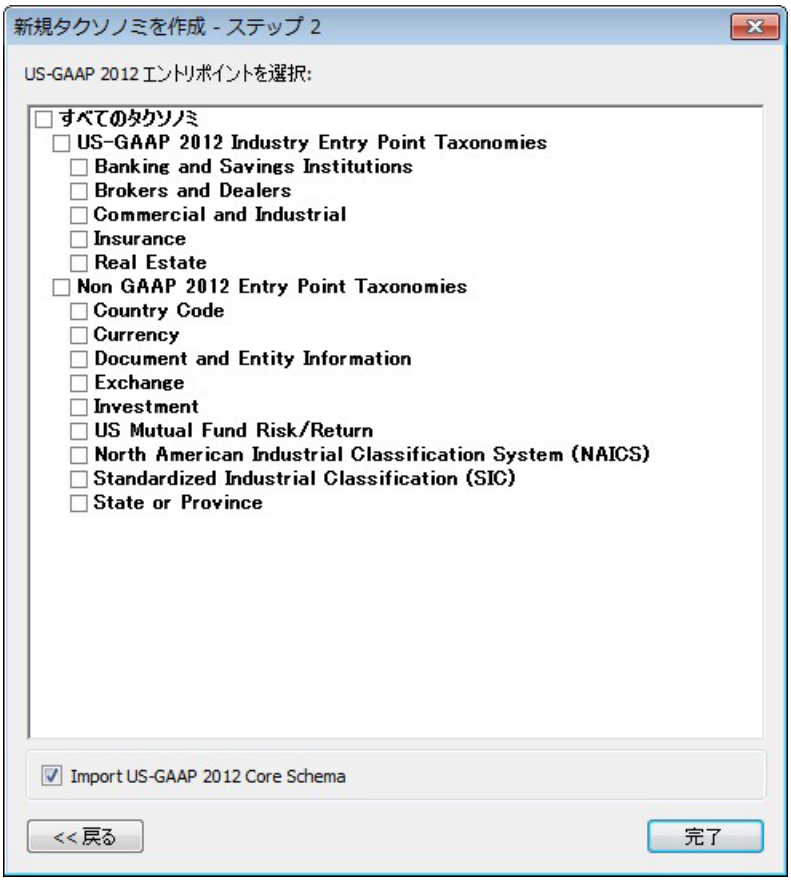
新規*タクソノミ*を作成ウィザードによりUS-GAAP やIFRS 要件に準拠したXBRL *タクソノミ*を作成することができます。実質的には、指定した*タクソノミ*がインポートされ、その後、必要に応じてファイルの編集を行うことができます。

ウィザードの最初のダイアログでは、以下の入力を行ないます：

- *タクソノミ*に使用する会社のティッカーシンボル (または名前) を作成します。入力する会社名は通常ユニークなもので、ここで入力されるティッカーシンボルをベースに重要な*タクソノミ*プロパティやコンポーネントの名前が作成されます。例えば、*ティッカー名* フィールドに入力された値は*タクソノミ*名前空間や*タクソノミ*のリレーションシップファイルを自動生成する際に使用されます (上のスクリーンショットを参照)。会社がティッカーシンボルを持っていない場合、このフィールドには*タクソノミ*が対象とする会社の名前を入力することができます。
- *タクソノミベース*で指定できるオプションは：*US-GAAP タクソノミ*、*IFRS*、そして*無し*です。このコンボボックスにて選択した*タクソノミ*により、あなたの*タクソノミ*が形づくられます。次に表示されるウィザードのダイアログの内容は、ここで選択した*タクソノミベース*により変わってきます。*タクソノミベース*については、次に表示されるウィザードの画面で説明されます。
- 日付の値は直接入力するか、コンボボックスのドロップダウンからアクセスすることのできる日付選択画面から選択することができます。ティッカーシンボルと同様、日付も*タクソノミ*名前空間や*タクソノミ*のリレーションシップファイルの生成時に使用されます。
- 保存先フォルダーはメイン*タクソノミ*ならびに関連するファイルが保存される場所になります。
- *タクソノミ*名前空間 (`xsd` 拡張子を持つメイン*タクソノミ*コンセプトファイルのターゲット名前空間) とコンセプト定義ファイル (スキーマファイル)、そしてリレーションシップファイルは、(i) ティッカーシンボル (または会社名) と、(ii) 日付から自動的に作成されます。これらの値を変更しようとすると、関連する*タクソノミ*に準拠しなくなるという旨の警告が表示されます。

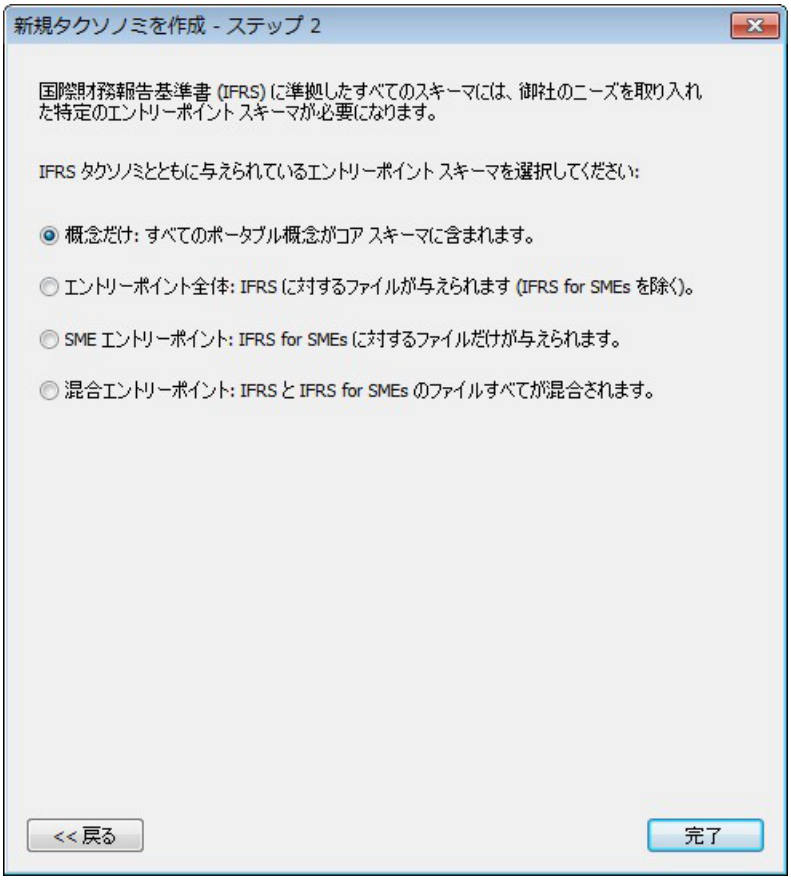
入力を完了後、**次へ**をクリックします。次に表示されるダイアログは、*タクソノミ*のベースに選択した内容により変化します。3つのケースが考えられます：

- US-GAAP タクソノミを選択した場合、(以下のスクリーンショットにあるように)新規タクソノミを作成ウィザードのダイアログにはUS-GAAP に特化した内容が表示されます。タクソノミへインクルードしたいエントリポイントを選択したり、ダイアログの下部にあるチェックボックスにて US-GAAP コアスキーマをインポートするかどうか指定することができます。



完了ボタンをクリックすると、選択したエントリポイントをインポートした新規コンセプト定義ファイル (.xsd) が、ウィザードにて指定した保存先フォルダーに作成されます。XBRL ビューで開かれるタクソノミを編集することもできます。

- タクソノミのベースとしてIFRS が選択された場合、(以下のスクリーンショットにあるように)新規タクソノミを作成ウィザードにはIFRS に特化した内容が表示されます。目的のエントリー ポイント スキーマ オプションを選択し、[**完了**] をクリックしてください。



完了ボタンをクリックすると、選択したエントリポイントをインポートした新規コンセプト定義ファイル (.xsd) が、ウィザードにて指定した保存先フォルダーに作成されます。XBRL ビューで開かれるタクソノミを編集することもできます。

- 無し (つまりベースとするタクソノミが無い) を選択すると、新規コンセプト定義ファイル (.xsd) が、ウィザードで指定した保存先フォルダーに作成され、XBRL ビューにて開かれます。2つの要素 (xbrldt:hypercube と xbrldt:dimension) が自動的に生成され、全てのタクソノミに必要となる基本的なスキーマが既にインポートされていることに注目してください (概要入力ヘルパーを参照ください)。

ベースタクソノミの作成後、次のステップは: (i) タクソノミの名前空間 (及びスキーマのターゲット名前空間) の作成と、(ii) 新規タクソノミを構築する土台となるタクソノミのインポートとなります。新規タクソノミウィザードを使用したのであれば、デフォルトのタクソノミ名前空間が作成され、関連するタクソノミがインポートされます。

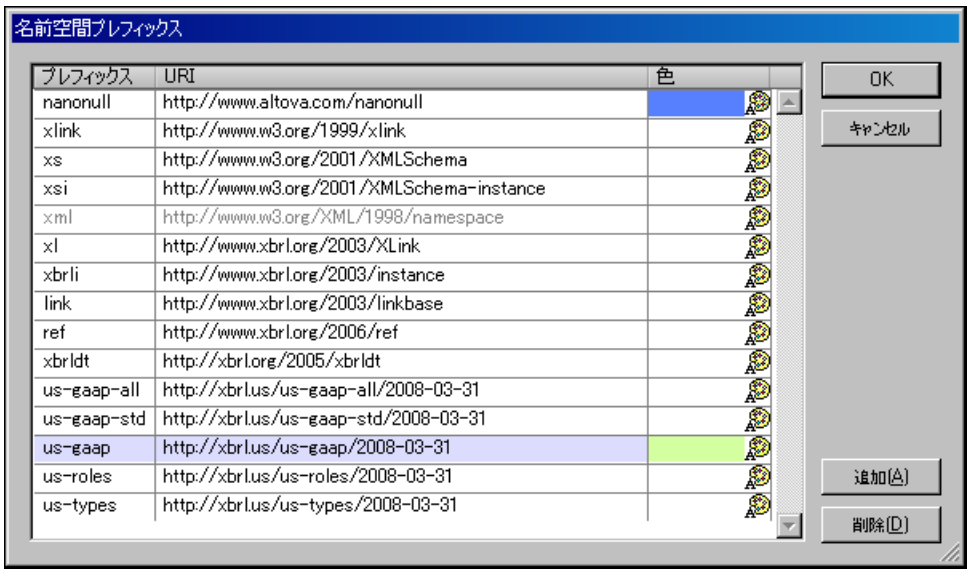
例題 : ステップ1

上の記述にある様に、新規タクソノミドキュメントを作成ウィザードを使って、新規タクソノミドキュメントを作成します (US-GAAP 2011 をベースタクソノミとして使用し、US-GAAP 2011 コアスキーマをインポート)。名前を入力し、適切な場所へ保存します。このファイルがメインタクソノミファイル、またはコンセプト定義ファイルとなり、XML スキーマファイルであるため、.xsd 拡張子を持つことになります。この例では作成するファイルを Nanonull.xsd とします。これは C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017Examples\XBRLExamples\Nanonull にある例と同じ名前で、リレーションシップファイルの名前をシンプルにするために、名前の日付部分を省略しています。

次のステップでは、新規タクソノミウィザードにより自動的に作成されたタクソノミのターゲット名前空間を修正します。

13.1.4 タクソノミ内の名前空間

タクソノミ名前空間は、メニューコマンド **[XBRL | 名前空間プレフィックス]** からアクセスすることができる (以下のスクリーンショットの) 名前空間ダイアログにより管理することができます。このダイアログでは名前空間ならびにプレフィックスの宣言や、各名前空間に対応した背景色を設定することができます。このダイアログでなされた変更により、タクソノミの名前空間宣言が修正されます。



名前空間プレフィックスダイアログにはタクソノミ内にある全ての名前空間が表示されます。

- 名前空間の追加や削除を行うには、それぞれ **[追加]** または **[削除]** ボタンを使用します。名前空間を追加した後は、対応するフィールドをダブルクリックして値を入力することで、デフォルトのプレフィックスとデフォルトの URI を修正します。
- 名前空間のカラーパレットにより、名前空間に色を割り当てることができます。色が名前空間に割り当てられると、その名前空間のコンポーネントは全て、XBRL ビューのメインウィンドウならびに入力ヘルパーにて、その色を背景に表示されます。特定の名前空間に割り当てられた色の設定は、タクソノミドキュメントを超えて、XBRL ビューで開かれているドキュメント全ての名前空間に適用されることにご注意してください。

名前空間ダイアログでの編集を終了後、**「OK」** ボタンをクリックして変更を確定してください。

ターゲット名前空間

タクソノミのターゲット名前空間は、タクソノミの `xs:schema` 要素にある `xs:targetNamespace` 属性により**定義されます** (以下のコードを参照)。`xs:schema` 要素はコンセプト定義ファイルのドキュメント要素です。

```
<xs:schema targetNamespace="http://www.altova.com/XBRL/Taxonomies">
    ...
</xs:schema>
```

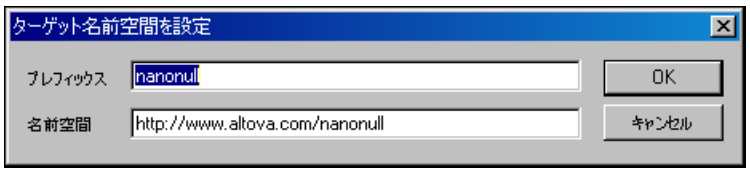
ターゲット名前空間を定義 (指定) するのに加えて、ターゲット名前空間は `xs:schema` 要素にて**宣言**されなければならず、これによりドキュメント全体をスコープに収めることができます。以下のコードはターゲット名前空間である名前空間の宣言を行っています。

```
<xs:schema targetNamespace="http://www.altova.com/XBRL/Taxonomies"
xmlns:ns1="http://www.altova.com/XBRL/Taxonomies" >
    ...
</xs:schema>
```

上のコードでは、名前空間は xs:schema 要素にて宣言され、nsl というプレフィックスが与えられています。

ターゲット名前空間の設定

新規タクソノミウィザードにより新しいタクソノミが作成されると、デフォルトのターゲット名前空間とプレフィックスが、タクソノミに対して自動的に作成されます。デフォルトのターゲット名前空間は、新規タクソノミウィザードで入力された値をベースに作成されます。デフォルトで作成されるターゲット名前空間のプレフィックスは nX となります（X は整数）。デフォルトターゲット名前空間の宣言とプレフィックスは、**[XBRL | ターゲット名前空間の設定]** コマンドによりアクセスできる（ターゲット名前空間ダイアログを使って編集することができます（以下のスクリーンショットを参照）。これらの値を編集することにより、ターゲット名前空間の**定義**（targetNamespace 属性の値）だけでなく、ターゲット名前空間の**宣言**も変更されます。



（ターゲット名前空間の定義を除いて）ターゲット名前空間の宣言だけを修正、または任意の名前空間の宣言だけを修正するには、名前空間プレフィックスダイアログ（**[XBRL | 名前空間プレフィックス]** コマンド）から、名前空間と、そのプレフィックスの値を編集してください。

例題：ステップ2

[XBRL | ターゲット名前空間の設定] コマンドを使用して、ターゲット名前空間を設定ダイアログを開きます。デフォルトのターゲット名前空間は http://xbrl.nanonull.com/2011 で、プレフィックスは n1 となっています。名前空間、そしてプレフィックスのフィールドをダブルクリックして、値の修正を行ないます。（上のスクリーンショットにあるように）名前空間には http://www.altova.com/nanonull を、そしてプレフィックスには nanonull を指定します。ダイアログにある［「OK」］ボタンをクリックすると、ターゲット名前空間が定義され、割り当てたプレフィックスと共に宣言されます。この例では、ターゲット名前空間が http://www.altova.com/nanonull で、プレフィックスが nanonull となります。

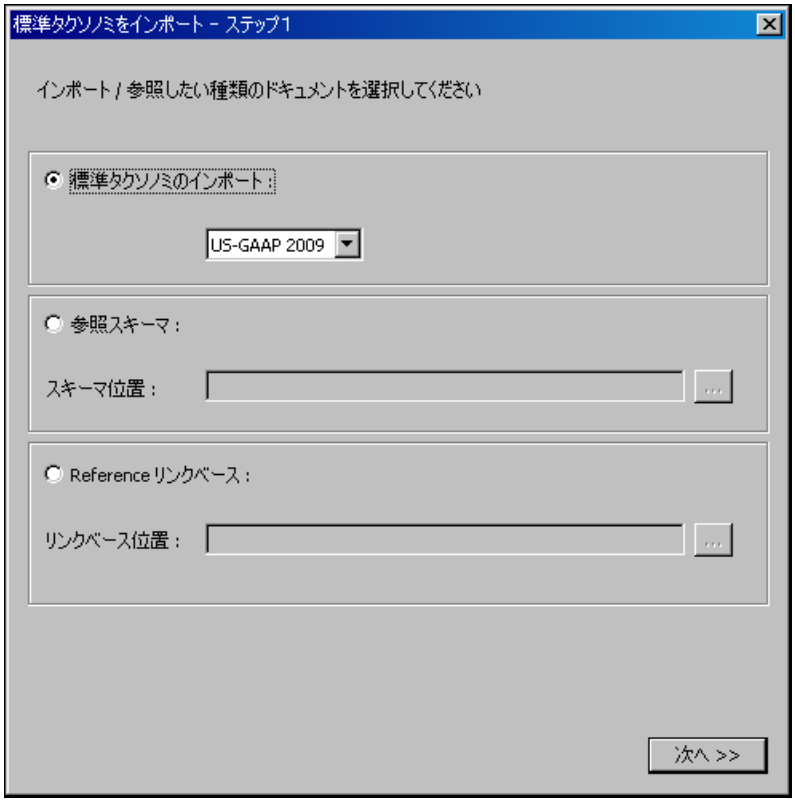
次のステップでは、XBRL ビューのインポートメカニズムについて詳しく見ていくことにします。

13.1.5 タクソノミのインポート

新規タクソノミが、既存のタクソノミの上に構築される場合、拡張したタクソノミに既存のタクソノミをインポートする必要があります。タクソノミを新規タクソノミウィザードを使って作成する場合、US-GAAP や IFRS をベースにしたタクソノミを、作成時にインポートすることができます。タクソノミのインポートはその後いつでも行うことができます。

以下の操作によりタクソノミをインポートすることができます：

1. XBRL ビューの概要入力ヘルパーを右クリックして、メニューコマンドから**[インポート参照]** を選択します。
2. 標準スキーマのインポートダイアログが表示され、インポートするタクソノミや参照するリンクベースを指定することができます（以下のスクリーンショットを参照）。選択されたオプションにより、ダイアログの名前が変化します。



インポート参照には、(i) 標準タクソノミ (US-GAAP または IFRS)、(i) その他のタクソノミ (参照スキーマ)、そして (ii) リンクベースという 3 つのオプションがあります。非標準タクソノミをインストールしている場合、参照スキーマラジオボタンを選択し、スキーマ位置のテキストフィールドにある **[参照]** ボタンをクリックして、目的のタクソノミを選択してください。

3. 選択を行った後に **[完了]** ボタンをクリックします。選択されたタクソノミがインポートされ、その要素とリレーションシップが XBRL ビューに表示されます。

以下の点に注意してください：

- 概要入力ヘルパーには、インポートされたタクソノミがインポートするタクソノミと、インポートされたタクソノミが使用するリンクベースが表示されます。
- グローバル要素入力ヘルパーでは、インポートされたタクソノミ内のコンセプトが表示されます。
- デザインウィンドウならびに詳細入力ヘルパーでは、インポートされたコンセプトはグレーで表されます。
- インポートされたタクソノミを削除するには、概要入力ヘルパーにてタクソノミを右クリックし、**[削除]** を選択します。

タクソノミをインポートした後は、必要に応じてタクソノミを拡張することができます。

メモ US-GAAP の様な巨大なタクソノミにより編集のスピードが落ちてしまう感じるのであれば、メインウィンドウのフィルタリングを使って、表示対象を新規タクソノミに制限することができます。この操作によりスピードが大幅に向上します。

インポートメカニズム

上で述べたような標準的なインポートの追加により、新規タクソノミファイルに `xs:import` 要素が追加されます。`xs:import` 要素によりインポートされたタクソノミの名前空間と場所が指定されます。

```
<xs:import namespace="http://fasb.org/us-gaap/2011-01-31"
           schemaLocation="http://xbrl.fasb.org/us-gaap/2011/elts/us-gaap-2011-01-31.xsd"/>
```

上の設定では、schemaLocation 属性によりタクソノミがインターネット上からロードされるよう指定されています。しかしこのURI はXMLSpy のカタログメカニズムにより、XMLSpy パッケージに同封されるUS-GAAP タクソノミのローカルコピーへマップされます。

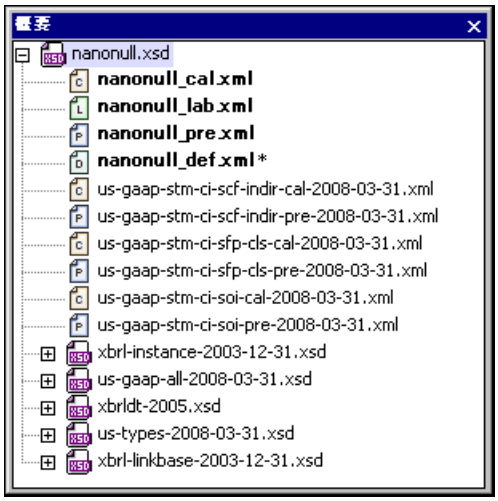
ローカルに保存されているタクソノミの位置を確認するには、ローカルのアドレスを直接使用することができます。その他にも、カタログファイルを使ってウェブアドレスからローカルのアドレスを発見することができます。

例題：ステップ3
この例題で作成しているタクソノミはUS-GAAP 2011 タクソノミを使用しています。このタクソノミは、タクソノミの作成時に既にインポートされています。概要入力ヘルパーにて、インポートされたタクソノミ参照されているリンクベースを良く見てください。テキストビューへ切り替え、xs:import 要素を探してみてください。XBRL ビューのメインウィンドウでは、インポートされたコンセプトがグレーのフォントカラーで表示されていることに注目してください。また、概要入力ヘルパーにて、US-GAAP タクソノミのリンクベースとインポートされたスキーマが表示されていることにも注目してください。

次のステップでは、リンクベースファイルと、参照メカニズムに関する解説を行ないます。

13.1.6 タクソノミファイルの設定

概要入力ヘルパーには、タクソノミを構成するファイルが、以下のスクリーンショットで示されるようにツリー構造で表示されます。ツリーのルートには、メインタクソノミファイル（コンセプト定義ファイル）があり、これが現在アクティブなファイルになります。その下のレベルにあるファイルには2種類あります：(i) タクソノミ内の様々なリレーションシップを指定するリンクベースファイル（これらは色付のアイコンで示されます）と、(ii) インポートされたスキーマ（xsd ファイル）です。



タクソノミのインポートセクションでは、概要入力ヘルパーからタクソノミをインポートする方法が記述されました。インポートされたタクソノミは、インポートされたスキーマと共に、概要入力ヘルパーに表示されます。

このセクションでは、概要入力ヘルパーを使ってリンクベースファイルを管理する方法について説明します。概要入力ヘルパーのコンテキストメニューからは、リンクベースを管理するための、4つのオペレーションにアクセスすることができます：

- 新規リンクベースの追加とタクソノミと共にそれらを保存。
- リンクベースの種類の設定。リンクベースの種類（計算、定義、表示、名称、そして参照）をXMLSpy が特定できなかった場合、リンクベースの種類を明示的に指定することができます。

- 特定の種類のリレーションシップリンクベースに対して、任意のリンクベースをデフォルトリンクベースとして設定します。特定の種類のリンクベース 例えばラベルリレーションシップに対して、2つ以上のリンクベースがある場合、タクソノミ内で新たに作成されるラベルは、デフォルトラベルリンクベースで作成されます。
- リンクベースの削除

メモ リレーションシップには5つの種類があります：(i) 定義、(ii) Calculation、(iii) Presentation、(iv) ラベル、(v) 参照。各リレーションシップの種類に対して別々のリンクベースファイルを作成することができます。

新規リンクベースの追加

新規リンクベースを追加するには：

1. 概要入力ヘルパーを右クリックして、**[新規リンクベースの追加 | <リレーションシップの種類>]** を選択します。選択されたリレーションシップの種類のリンクベースファイルが、デフォルトの名前で概要入力ヘルパーに作成されます。作成された新規リンクベースは、そのリレーションシップの種類のデフォルトリンクベースになることに注意して下さい（ファイル名が太字で表示されます）。
2. デフォルト名を右クリックして、**[名前の変更]** を選択し、名前を修正します。
3. 新たに作成されたリンクベースファイル名は、メインタクソノミファイルが保存される際に、物理的にファイルシステムで保存されます。詳細については以下を参照下さい。

リンクベースファイルの保存

リンクベースファイルが保存されていない場合、リンクベースファイルの名前の右側にアスタリスク (*)が表示されます。メインタクソノミファイルの保存は、以下のように行われます：

1. リンクベースのパスを確認するダイアログが表示されます。このダイアログでは、新たに追加されたものも含めて、タクソノミ内にある全リンクベースの名前と場所が表示されます。保存されていないリンクベースファイルのパスには、メインタクソノミファイルが保存されているフォルダーへのパスが与えられます。リンクベースを別の場所に保存したい場合は、各リンクベースファイルのパスを編集することができます。ファイルの名前も変更することができます。
2. 完了後、**「OK」** をクリックします。指定された場所にリンクベースファイルが保存されます。

リンクベースの種類のセット

このコマンドを使うと、ファイルのリンクベースの種類をセットすることができます。リンクベースの種類を変更したいファイルを右クリックして、コンテキストメニューから**[リンクベースの種類を設定 | <リンクベースの種類>]** を選択します。**[すべて]** オプションにより、リンクベースファイルには2つ以上のリレーションシップが含まれていると指定することができます。

デフォルトリンクベースのセット

各リレーションシップの種類に対して、デフォルトリンクベースファイルをセットすることができます。その種類のリレーションシップがタクソノミエディターにてセットされると、リレーションシップがその種類のデフォルトリンクベースファイルに保存されます。リンクベースファイルをデフォルトリンクベースとしてセットするには、リレーションシップを右クリックして、**[デフォルトリンクベースの設定 | <リンクベースの種類>]** を選択します。デフォルトリンクベースの名前は太字で表示されます。

リンクベースの削除

リンクベースを右クリックし、**削除**を選択することで、タクソノミからそのリンクベースを削除することができます。

例題：ステップ4

作成中のタクソノミファイルには、既にリンクベースファイルが含まれています。これらのリンクベースファイルは新規タクソノミの作成ウィザードにより作成されたものです。サンプルファイルでは、デフォルトで与えられている名前が以下のように変更されています：

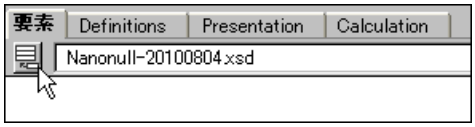
- Calculation リンクベース：`nanonull_cal.xml`
- 定義 リンクベース：`nanonull_def.xml`
- ラベルリンクベース：`nanonull_lab.xml`
- プレゼンテーション リンクベース：`nanonull_pre.xml`

このセクションで解説されているコマンドを試してみるには、**[新しいリンクベースの追加]** コマンドを使って、お好きな種類のリンクベースファイルを作成します。上記にあるように、名前の変更をします。新たに作成されたリンクベースが、その種類のデフォルトリンクベースとなっていることに注意してください（名前が太字で表示されます）。そのリンクベースを選択し、**[リンクベースの種類を設定]** コマンドを使って、他のリレーションシップの種類に変更します。ファイルが新たなリンクベースの種類のデフォルトリンクベースには**ならない**ことに注目してください。**[削除]** コマンドにより、リンクベースを削除します。オリジナルのリンクベースファイルは既にデフォルトリンクベースではないので、そのファイルをデフォルトリンクベースに設定しなおします。

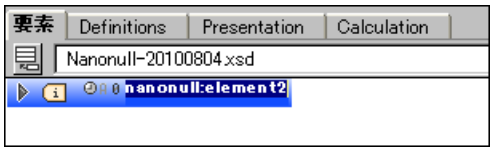
次のステップでは、メインタクソノミファイルへの新たな要素の追加について説明します。

13.1.7 タクソノミへ要素の追加

タクソノミへ要素（コンセプト）を追加するには、メインウィンドウにある **「新しい要素の追加」** アイコンをクリックします（以下のスクリーンショットを参照）。

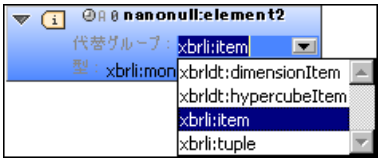


`xbrli:item` の代替グループを持つ新規要素がデフォルトの名前で、画面上の要素リストに追加されます（以下のスクリーンショットを参照）。



要素ボックスについては、メインウィンドウ| 要素タブのセクションを参照ください。以下の方法により、メインウィンドウにて要素のプロパティを編集することができます。

- 要素のデフォルト名をダブルクリックすることで、要素の名前を変更することができます。名前には適切な名前空間プレフィックスも入力する必要があるということに注意してください。
- プラス（+）アイコンをクリックして要素ボックスを展開し、以下のスクリーンショットにあるように、フィールドのドロップダウンリストから目的の代替グループを選択することで、要素の代替グループを変更することができます。



- Balance、Period、Abstract、またはNillable プロパティを変更するには、要素名の左に配置されている対応するアイコンをクリックし、そこで表示されるボックスに表示されるオプションを選択します。
- 要素にラベルリンクロールを追加するには、要素ボックスを右クリックし、**「ラベルリンクロールの追加」**コマンドを選択します。ラベルリンクロールの行が追加され、この行にラベルリンクロールを入力したり、コンボボックスからオプションを選択することができます。タクソノミにラベルリンクロールが割り当てられていない場合、ここでラベルが作成され、概要入力ヘルパーにて表示されます。
- ラベルリンクロールを右クリックして、**「ラベルの追加」**コマンドを選択することで、ラベルリンクロールにラベルを追加することができます。ラベルの詳細を入力するには、フィールドをダブルクリックして新しい値を入力するか、対応するコンボボックスから新しい値を選択します。ラベルに加えられた変更は、メインタクソノミファイルの保存時にラベルリンクベースに保存されます。
- 参照はラベルと同様の方法により参照リンクベースに追加されます。参照リンクベースはまず要素に対して追加され、特定の参照リンクロールに対して追加されます。

要素プロパティは詳細入力ヘルパーで編集することができます。編集方法についてはXBRL ビューの入力ヘルパーのセクションを参照ください。

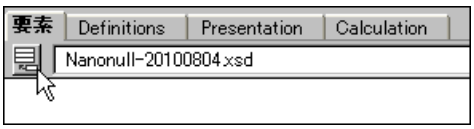
例題：ステップ5

このセクションでは、新しい要素を作成してUS-GAAP タクソノミの拡張を行ないます。

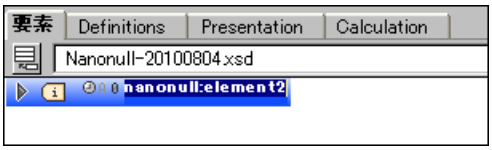
最初に作成する要素は`nanonull:OnboardAndOther` で、Nanonull 社の客船での売上による歳入を表しています。これほど詳細な歳入項目はUS-GAAP タクソノミ内で定義されておらず、US-GAAP の拡張を作成する必要があります。Nanonull 社に特化したタクソノミとして、タクソノミはNanonull 名前空間（`http://www.altova.com/nanonull`）内に作成されなければなりません。名前空間は`naanonull` という名前空間プレフィックスにより既に宣言されています。このプレフィックスを持つ要素を作成すると、要素はNanonull 名前空間内に配置されます。

要素を作成するには：

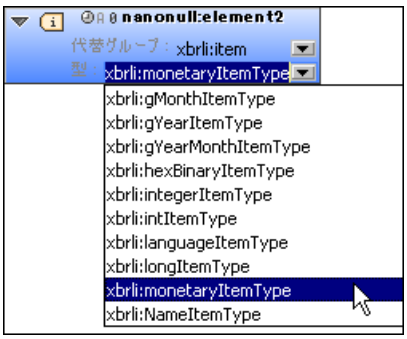
1. メインウィンドウにある、**「新しい要素の追加」**アイコンをクリックする



`xbrli:item` という代替グループと、デフォルトの名前を持った新しい要素が、要素のリスト内に作成されます。



2. 要素名をダブルクリックして、`nanonull:OnboardAndOther` を入力します。この操作により`Nanonull` 名前空間内に`OnboardAndOther` 要素が作成されます。
3. 要素ボックスを展開します。要素には金額が入るので、以下のスクリーンショットにあるように`Type` 属性を`xbrli:monetaryItemType` に変更します。

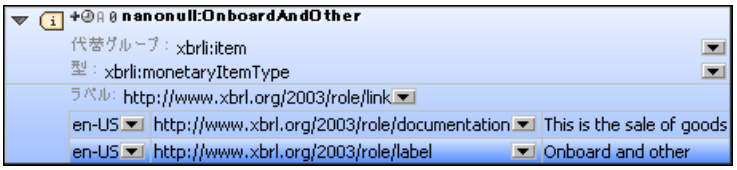


4. 時計アイコンの左の空間をクリックし、ポップアップから`credit` を選択します (以下のスクリーンショットを参照)。

これで`xbrli:balance` 属性の値が`credit` になりました。

5. 時計アイコン、`A` アイコン、そして`0` アイコンをクリックして、それぞれ*duration*、*非 Abstract*、*Nillable* にセットします。xsd ファイル内部では、`xbrli:duration`、`xs:abstract`、`xs:nillable` の値が、それぞれ`credit`、`duration false`、そして`true` となります。
6. 要素ボックスを右クリックして、表示されたメニューから **「ラベルリンクロールの追加」** を選択します。この操作により要素ボックスの下部にラベルリンクロールの行が追加されます。

7. XBRL リンクURI を選択します。
8. ラベルリンクロールの行を右クリックして、メニューから **「ラベルの追加」** を選択します。この操作により、ラベルリンクロール以下に新たにラベルの行が作成されます。
9. 新たに作成されたラベル行の (一番左にある) 言語フィールドをダブルクリックして、`en-US` を入力します (以下のスクリーンショット参照)。リンクロールを指定する次のフィールドでは、ドロップダウンリストからドキュメント を選択します。ラベルフィールドにはドキュメンテーションに表示されるテキストを入力します。その後、同様の操作を`label` リンクロールに対しても行ないます。要素の表示が (左側にある矢印をクリックすることで) 展開されると、名称の右側にあるプラスやマイナスシンボルをクリックすることで、`label` ロールの表示を切り替えることができます (名称の表示を切り替え)。



これで要素`nanonull:OnboardAndOther` が作成されました。

`OnboardAndOther` には`credit` (貸方) の`xbrli:balance` 値があることに注意してください。これはこのアイテムが収益のアイテムであるためです。当然売られる品物にも仕入れのためにコストがかかるため、対応する`nanonull:CostOfOnboardAndOther` という要素を借方側に作成します。nanonull:OnboardAndOther を作成したのと同様の方法でこの要素を作成します。この要素では、`xbrli:balance` の値を`credit` ではなく`debit` にセットしてください。

コストの中にはエージェントに対するコミッションも含めなければなりません。従って、`xbrli:balance` はdebit の`nanonull:CruiseCommissionsTransportationAndOther` という要素を作成します。`nanonull:CostOfOnboardAndOther` をと全く同じような方法でこの要素を作成してください。

最後に、地域ごとのグループ化を行うために、Asia、Europe、そしてUSA という3つの要素を作成します。これら要素はグループ化の為だけに使用され、要素そのものは値を持たず、このような要素はAbstract 要素と呼ばれます。その要素がどのような型を持つかは重要ではありません。Abstract 要素には、それら要素が使用される状況に最も適した型を与えるのが良いでしょう。例えば、サンプルファイルでは、これら3つの要素にstringItemType 型が与えられています。今までに作成した要素と同様に、nanonull:Asia、nanonull:Europe、そしてnanonull:USA を作成します。これら要素に対しては、Abstract 属性を*Abstract* にセットします（xsd ファイル内に記述される実際の属性値はtrue になります）。xbrli:balance 属性の値はセットしません。

メモ Abstract 要素にてxbrli:balance 属性許されているのはmonetaryItemType の要素だけで、そうでない場合、タクソノミが不正となってしまいます。Abstract 要素からは、オプションのxbrli:balance 属性をセットしないようにした方が賢明です。

次のステップでは、新規タクソノミのリンクロールを指定します。リレーションシップの作成時にこれらリンクロールが必要になります。

13.1.8 リレーションシップとリンクロール

リレーションシップのセットが作成される時、これらリレーションシップは要素内にて作成されます。例えば、Definition リレーションシップが作成されるとき、これら定義 リレーションシップを定義するもの（ロケータと定義 アーク）はすべて、以下にあるようなdefinitionLink 要素内にて作成されます。

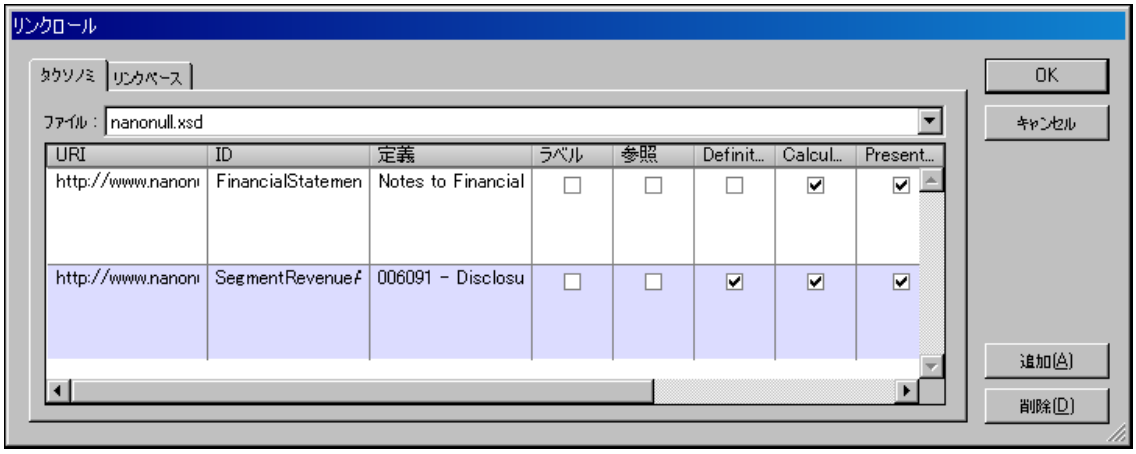
```
<link:definitionLink xlink:type="extended"
xlink:role="http://www.nanonull.com/taxonomy/role/SegmentRevenueAndOperatingIncome">
```

definitionLink にあるxlink:role 属性の値は、Definition リレーションシップで使われるリンクロールのroleURI 属性の値で無ければなりません（以下のコードを参照）。以下のコードに見られるように、リンクロールはタクソノミのappinfo 要素内に含まれています。

```
<xs:appinfo>
  <link:roleType id="SegmentRevenueAndOperatingIncome"
  roleURI="http://www.nanonull.com/taxonomy/role/SegmentRevenueAndOperatingIncome">
    <link:definition>006091 - Disclosure - Segment Revenue and Operating Income</link:definition>
    <link:usedOn>link:calculationLink</link:usedOn>
    <link:usedOn>link:definitionLink</link:usedOn>
    <link:usedOn>link:presentationLink</link:usedOn>
  </link:roleType>
</xs:appinfo>
```

リンクロールは他のリレーションシップの種類を含んだ（例えばcalculationLink やpresentationLink 要素といったdefinitionLink 要素以外の）要素にて使用されます。上のコードにおいて、このリンクロールが使用されても良いリレーションシップの種類を指定するusedOn 要素があることに注目してください。

リンクロールを作成するには、XBRL ビューにて、メニューコマンドから**[XBRL | リンクロール]** を選択してください。以下のスクリーンショットに示されるリンクロールダイアログが表示されます。



タクソノミタブにおいて、ファイル コンボ ボックスのドロップダウン リストからタクソノミを選択し、**[追加]** をクリックしてリンクロールを追加します。リンクロールのURI とID (上のコードを参照) を指定します。このリンクロールが使用されても良いリレーションシップの種類を選択します。目的のリレーションシップの種類があるチェックボックスにチェックを入れてください。

例題 : ステップ6
リンクロールダイアログ (**XBRL | リンクロール**) を使って、上のスクリーンショットにあるように2つのリンクロールを作成します：

1. `id="SegmentRevenueAndOperatingIncome" URI="http://www.nanonull.com/taxonomy/role/SegmentRevenueAndOperatingIncome"` (定義、計算 そして プレゼンテーション リレーションシップで利用可)。
2. `id="FinancialStatements" URI="http://www.nanonull.com/taxonomy/role/FinancialStatements"` (計算 ならびに プレゼンテーション リレーションシップで利用可)。

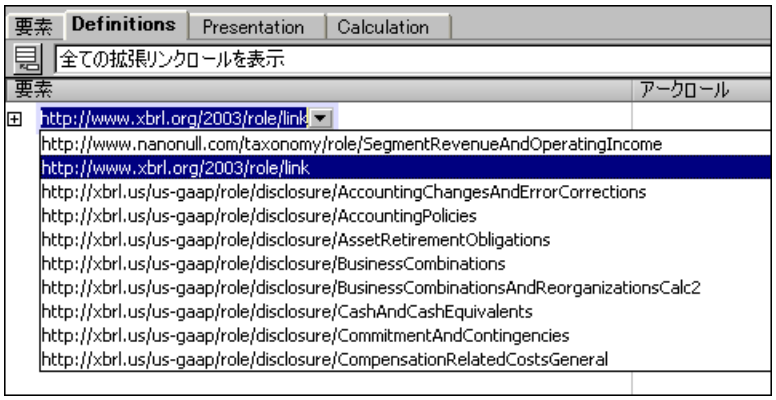
次のステップでは、新規タクソノミのリレーションシップの作成について説明します。

13.1.9 リレーションシップの作成 : パート1

リレーションシップは、それぞれ対応する各タブにて作成されます (Definisions、Presentation、Calculation)。これら3種類のリレーションシップの作成は、Definition リレーションシップがアークロールを持ち、Presentation リレーションシップと Calculation リレーションシップは持たないという点を除いてほとんど同じです。このセクションでは 定義 リレーションシップを例にしてリレーションシップの作成方法を説明します。次のセクションでは、Presentation ならびに Calculation リレーションシップの違いや、リレーションシップに関するその他の機能について解説します。

以下の説明を読み進めるにあたり、XBRL ビューにて既に完成されたドキュメントを開くことをお勧めします。Nanonull タクソノミはフォルダー `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017Examples\XBRLExamples\Nanonull` にて見つけることができます。

リンクロールの追加
メインウィンドウにて目的のタブ (Definitions、Calculation、Presentation) をクリックします。メインウィンドウを選択し、**拡張リンクロールの追加 /リンクロールの管理** ボタンから **拡張リンクロールの追加** コマンドを選択します。この操作により、以下のスクリーンショットにあるようにデフォルトリンクロールのURI が、含まれる行が追加されます。行の右側にあるドロップダウンの矢印をクリックすると、選択可能なリンクロールのリストが表示されます。

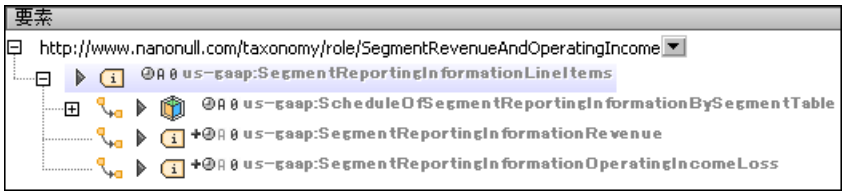


目的のリンクロールが表示されていない場合、それはタクソノミまたは現在のリレーションシップの種類にて、そのリンクロールが定義されていないためです。リンクロールとその作成方法に関する詳細は[リレーションシップとリンクロール]のセクションを参照ください。

追加できるリンクロールの数に制限はありません。

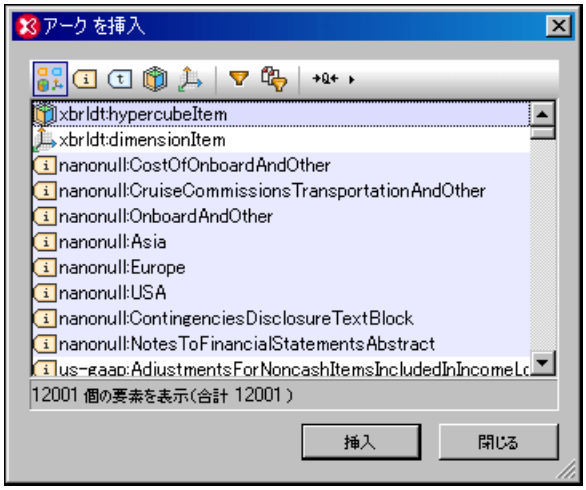
リンクロールへの要素参照やアークの挿入

リンクロール内で最初で作成される要素**から**、他の要素へのリレーションシップが作成されます（以下のスクリーンショットを参照）。この要素は、通常 Abstract 要素となり、バランスシートの要素のように他の要素をグループ化するために使用されます。この要素**から**アークが作成されるため、アークロール列へのエントリは存在しません。アークロールは、アークを送られる側の要素にて表示されます。

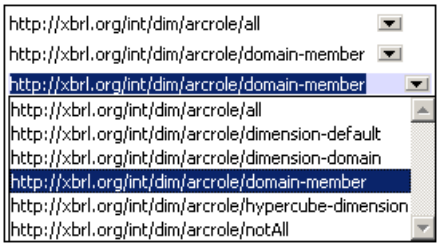


上のスクリーンショットでハイライトされている要素は、挿入された要素参照です。3つのアークがあり、1つがハイパーキューブ要素へ、2つがアイテム要素へのものです。各アークには、それぞれアークロールがあり、それらアークロールは「**from-to**」リレーションシップを定義し、これら3つの要素は各アークの「**to**」側の要素となります。

アークを要素参照または要素に挿入するには、「**from**」要素を右クリックし、コンテキストメニューから「**アークの挿入**」を選択します。この操作によりアークを挿入ダイアログが表示されます。アークの「**to**」側に作成する要素を選択します。このダイアログに表示されている内容を絞り込むには、フィルターの条件を指定してください（フィルタリングについては[XBRL ビュー内の入力ヘルパー]に関する記述を参照ください）。



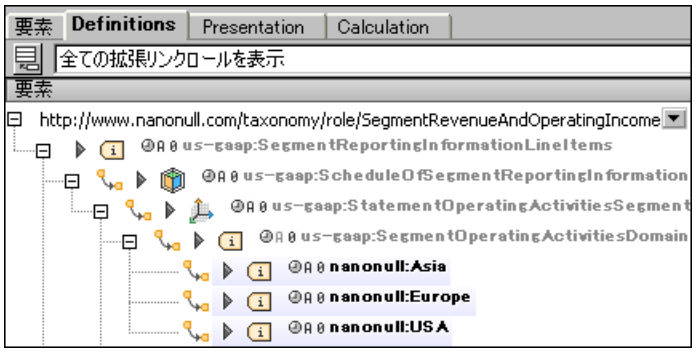
デフォルトのアークロールと共に要素が挿入されます。以下のスクリーンショットにあるように、アークロールのドロップダウンリストから別のアークロールに変更することもできます。



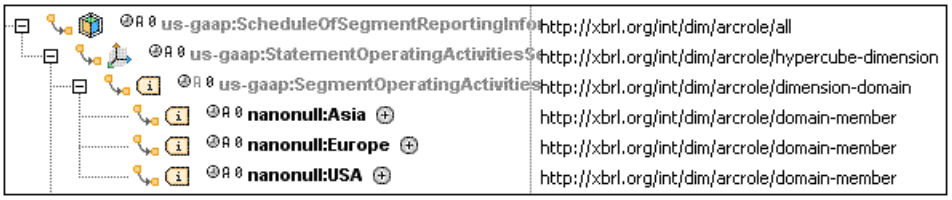
メモ 要素はアークと共にグローバル要素入力ヘルパーからドラッグすることで追加することもできます。

例題：ステップ7

以下のスクリーンショットにあるように定義リレーションシップを作成します。



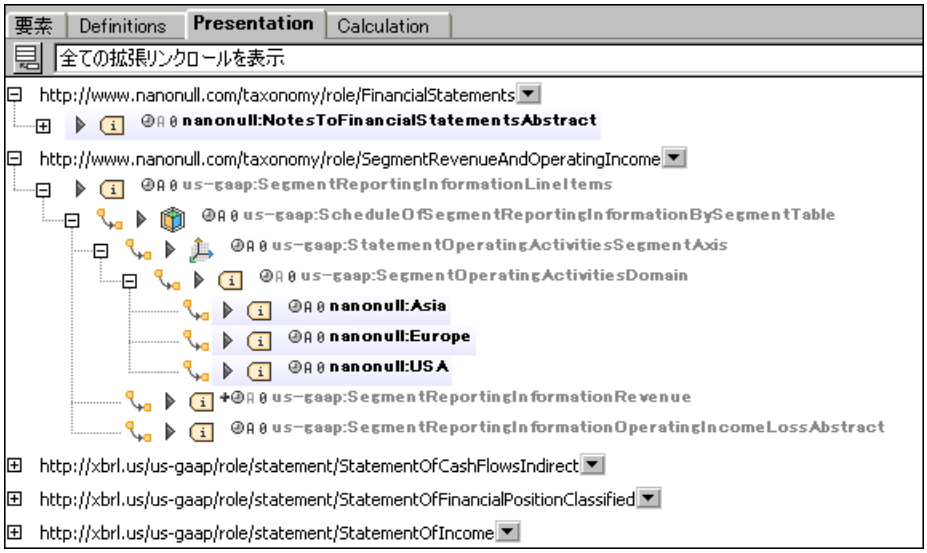
上のスクリーンショットにはアークと共に追加される要素が示されます。以下のスクリーンショットには新たに追加された要素のアークロールを示します。



作成したタクソノミとXMLSpy パッケージに含まれているものを比較することもできます。提供されるタクソノミ (nanonull.xsd) は`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017Examples\XBRLExamples\Nanonull` 以下のフォルダーに含まれます。

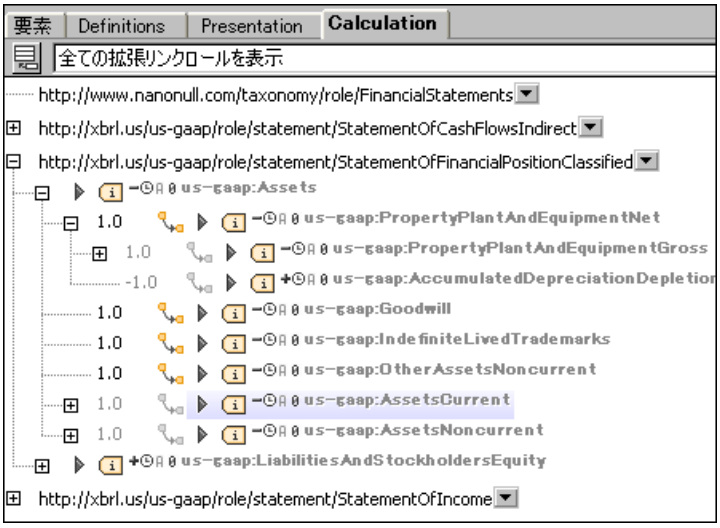
13.1.10 リレーションシップの作成 :パート2

以前のセクションのリレーションシップの作成 :パート1では、Definitions リレーションシップの作成方法がデモされました。Presentation リレーションシップ (以下のスクリーンショット) ならびにCalculation リレーションシップも同じような方法で作成されます。唯一の違いは、これらのリレーションシップにはアークロール列が存在しないことです。



以下の点に注意してください:

- プレゼンテーションならびに計算 リレーションシップは、親子関係のような2つの要素間の単純なアークとして見ることができます。アークアイコンにより2つの関係が強調されます。要素にアークを挿入することは、グラフィカルに子要素を作成するのと同様な操作と考えることができます。従って、アークを使用することで階層構造を構築することができます。
- グローバル要素入力ヘルパーから直接ツリーに要素をドラッグすることもできます。これら要素は常にアークの **「to」** 側に配置されます。要素がドロップされると、その位置に矢印が表示されます。
- Calculation アークには、アークにより **「to」** 要素の値が合計されるときに使用される`weight` 属性があります (以下のスクリーンショット参照)。例えば、`weight` の値が`+1.0` であれば、要素の`100%` の値が**[from]** または`summation`) 要素へ加えられます。属性の値が`-1.0` であれば、`100%` の値が`summation` 要素から引かれます。`weight` 属性をダブルクリックすることで、異なる値を入力することができるようになります。



`weight` 属性は、以下のスクリーンショットにあるように、詳細入力ヘルパーからも変更することができます。

アーク使用の禁止

それが定義、Presentation、Calculation であるかに関わらず、すべてのアークには、`optional` または `prohibited` の値を持たせることができます。`prohibited` の値が使用されると、アークは無効になります。

表示色とコンテキストメニュー

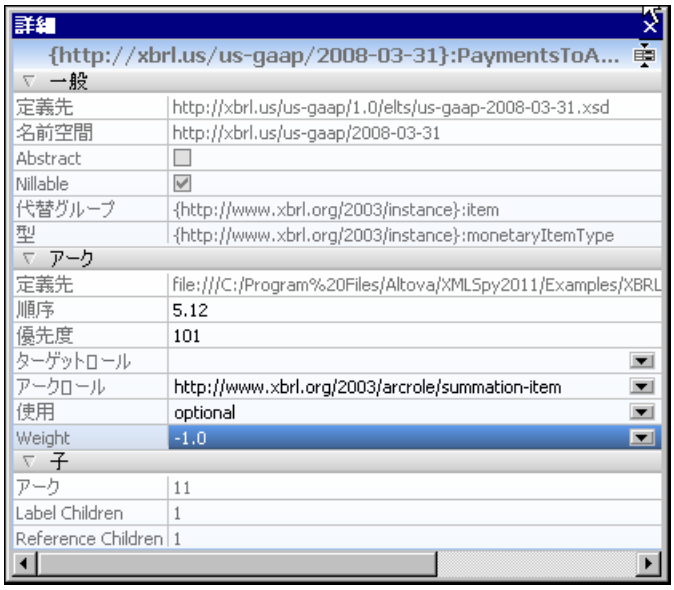
現在のタクソノミ内に要素が作成され、編集することができるのであれば、要素は黒色で表示されます。そうでない場合（それら要素がインポートされたタクソノミ内にあり、編集することができない場合）、要素はグレーで表示されます。

リレーションシップタブのコンテキストメニューには以下のエントリが表示されます。

- *要素参照の挿入*：拡張リンクロールで利用可能。リンクロール以下に、アークの **[from]** 側となる要素を作成します。
- *要素参照の削除*：リンクロール直下の要素参照にて利用可能。
- *アークの挿入*：要素にて利用可能。アークの **to** 側となる要素を選択するためのダイアログが表示されます。
- *ターゲットロールの設定*：選択された要素にターゲットロールをセット。
- *ラベルリンクロールの追加*：選択された要素にラベルリンクロールを追加。
- *参照リンクロールの追加*：選択された要素に対して参照リンクロールを追加。
- *アークのオーバーライド*：アークの `use` 属性の値を `optional` から `prohibited` に置き換え、アークを無視します。
- *アークの削除*：選択されたアークを削除。
- *グローバル要素で表示*：選択された要素をグローバル要素入力ヘルパーにてハイライトします。

詳細入力ヘルパー

リレーションシップ内の要素が選択状態であれば、詳細入力ヘルパーを使用してアーク属性を編集することができます（以下のスクリーンショットを参照）。

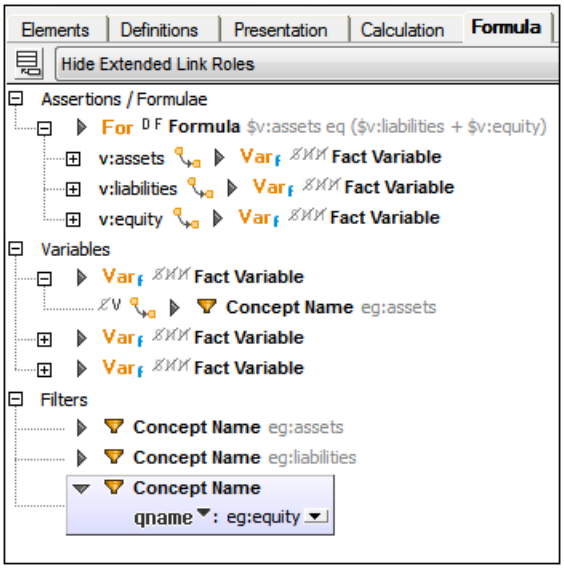


`order` や`priority` といったメインウィンドウのグラフィカルな表示からは編集できない属性は、詳細入力ヘルパーにて編集を行うことができます。

13.2 XBRL フォーミュラエディター

XBRL フォーミュラ変数とフィルター仕様は、XBRL ビジネス報告書内のデータから新しいファクト値を取得するために使用されることのできる、式ルールのための構文を提供します。ジェネリックなラベルおよび参照仕様は、異なる構成のXBRL をラベルするすべての方法をサポートします。 XBRL フォーミュラのコンテキストでは、ラベルと参照は、ヒューマンドキュメントとフォーミュラを関連付けるために使用することができます。変数は、XBRL ビジネス報告書内のどのファクトがフォーミュラ内の評価の使用方法のために変数により選択されるかをフィルターします。 検証と3つのアサーション仕様は、ビジネス報告書の予期されるコンテンツについてのルールの構文を、変数および変数とフォーミュラのセットにより定義します。XBRL フォーミュラの構文とセマンティクスの概要については、Working Draft of XBRL Formula Overview 1.0 で確認することができます。

XMLSpy のXBRL フォーミュラエディターは、アプリケーションのXBRL タクソノミエディターの一部として実装されます。XBRL ビューのフォーミュラタブで使用することができます (*下のスクリーンショット参照*)。



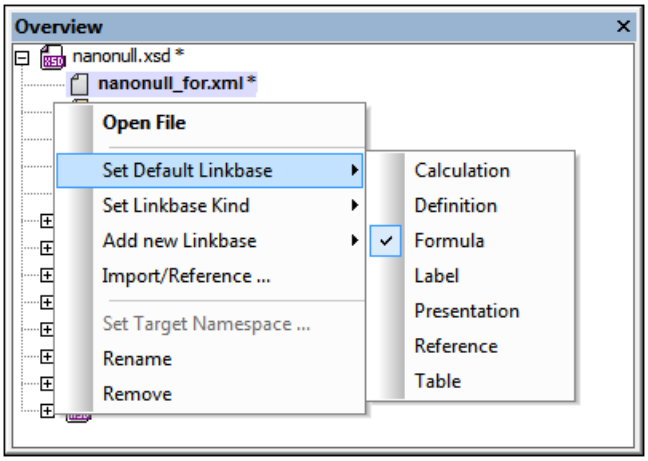
フォーミュラタブ概要入力ヘルパーおよび詳細入力ヘルパーとともに使用され、フォーミュラを作成および編集します。概要入力ヘルパーは、 XBRL フォーミュラのデフォルトのリンクベースを設定するために使用されます (フォーミュラがデフォルトで保存されるファイルです)。詳細入力ヘルパーは、フォーミュラコンポーネントのコンテンツを編集するために使用されます。 (このような編集は、 フォーミュラタブで直接行うこともできます)。

13.2.1 フォーミュラ リンクベース および リンクロール

標準のXBRL リンクベース (定義、プレゼンテーション、計算) は、コンセプト間のリレーションをロケーターと標準拡張リンク内の標準アークで定義しますが、フォーミュラ リンクベースは、フォーミュラ コンポーネント (フォーミュラ、変数、フィルター、アサーションなど) 間のリレーションを定義します。これらの定義は、リソースとジェネリックな拡張リンク内のジェネリックなアークにより指定されます。

フォーミュラ リンクベースの追加

概要入力ヘルパー内 (*下のスクリーンショット*) タクソノミファイルまたは、既存の リンクベースを右クリックして、**[新規リンクベースの追加 | フォーミュラ]** を選択します。追加されたリンクベースは、デフォルトのフォーミュラ リンクベースファイルになります。他のフォーミュラ リンクベースファイルをデフォルトのフォーミュラ リンクベースにする場合は、右クリックして、**[デフォルトのリンクベースの設定 | フォーミュラ]** を選択します (*下のスクリーンショット*)。



デフォルトのリンクベースは、太字で表示されていて、保存されていない変更されたリンクベースは、アスタリスクでマークされていることに注意してください。

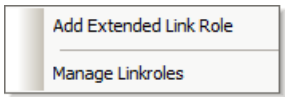
フォーミュラリンクベース フォーミュラタブは、フォーミュラタブで表示されています。

メモ: フォーミュラリンクベースが存在しない場合にフォーミュラコンポーネントがタクソノミに追加された場合、フォーミュラリンクベースは自動的に作成されます。

リンクロール

標準拡張リンクのケースと同様（定義、プレゼンテーション、計算）、ジェネリックなリンクは、同じ型のリレーションシップを区切り、結合が解除されたネットワークにするリンクロールの値を定義する必要があります。同じリンクを持つすべてのジェネリック名拡張リンクは、異なるリンクベースファイル内に存在する場合でも、フォーミュラタブ内のダイアグラム内で1つのリンクロールノードの下で結合されています。

ジェネリックなリンクロールは、背景エリアのコンテキストメニューを介したダイアグラム内で作成することができます。（下のスクリーンショット）。ですが、このコンテキストメニューは、フォーミュラタブ内のビューオプションコンボボックスがすべての拡張リンクロールを表示に切り替えられている場合のみ表示されることに注意してください。

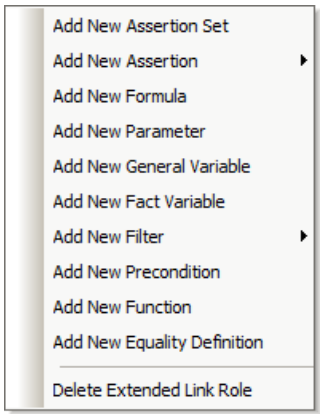


このメニューは、ツールバーアイコン 拡張リンクの追加 /リンクロールの管理 を介して使用することができます。リレーションシップネットワークは、フォーミュラリンクベースにとって、リレーションシップネットワークはそれほど重要ではないため、デフォルトのフォーミュラタブのデフォルトの表示は、リンクロールを隠し、フォーミュラコンポーネントをリンクロール無しで表示する拡張リンクロールを隠すです。

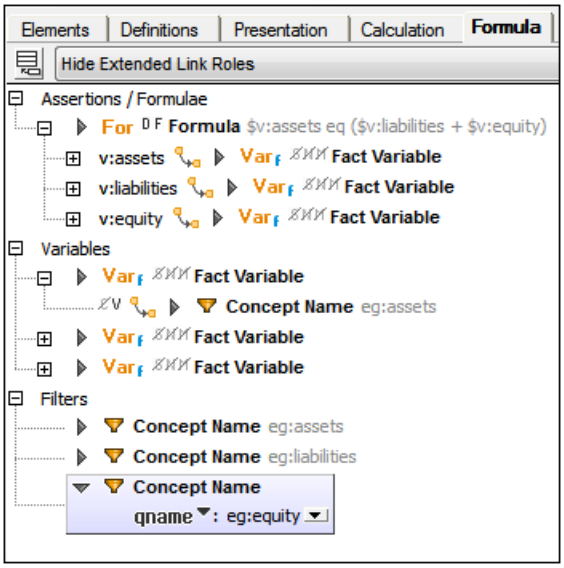
デフォルトのフォーミュラリンクベース ファイルがない場合、デフォルトのフォーミュラリンクベースが自動的に作成されます。また、リンクロールが作成された時点でデフォルトのリンクベースファイルにリンクロールが存在しない場合、デフォルトのリンクベースファイルが作成されます。

13.2.2 フォーミュラ コンポーネント

新しいフォーミュラ コンポーネントは、リンクロールノードのコンテキストメニューを介して作成されます。 (*下のスクリーンショット*)。または、 ツールバーアイコン *新しいフォーミュラコンポーネントの追加*を介して、*拡張リンクロールを隠す*に設定されたビューにより作成されます。

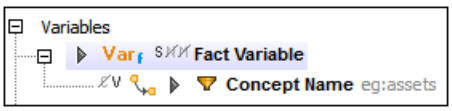


複数のコンポーネントを含むメカニズムについては、このセクションのサブセクションで説明されています。フォーミュラ コンポーネントが追加された後、フォーミュラ タブ内のダイアグラムに表示されます (*下のスクリーンショット参照*)。



意味を明確にするために、フォーミュラ コンポーネントは、リレーションシップと他のコンポーネント (アーク)のセクションに分割されており、ツリー構造内に表示されています (*上のスクリーンショット参照*)。

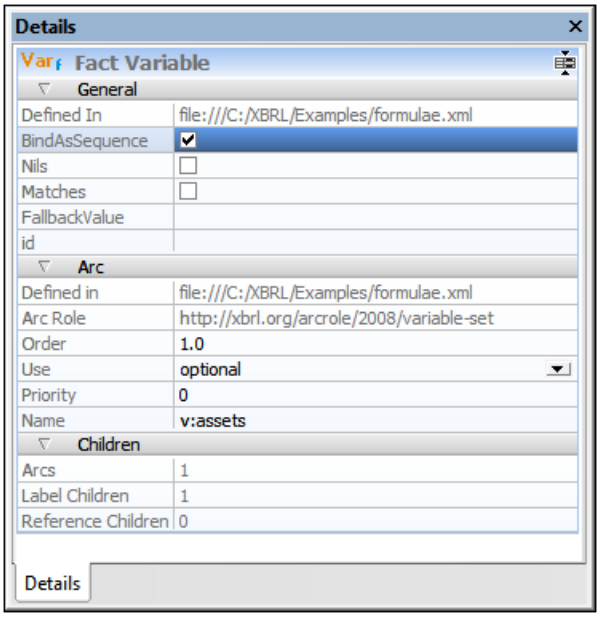
コンポーネントのプロパティおよびリレーションシップ (アーク)は、アイコン コンポーネントの または アークの左側のアイコンとして、ダイアグラム内 に表示されています。 (*下のスクリーンショット参照*).



例えば、上のスクリーンショット内では、ファクト変数 コンポーネントには3つのプロパティがあります :(S アイコンにより表示されている)BindAsSequence、Nils (N アイコン)および、 Matches (M アイコン)。 これらは、すべてboolean プロパ

ティです。ダイアグラム内でアイコンからのラインを持たない (bindAsSequence) は、true の値を持ちます。他の2つのプロパティは、(それぞれラインを介して表示された) false の値を持ちます。 (変数の下の) アークは、2つの プロパティを持ち、最初プロパティは、boolean false、二番目のプロパティは boolean true です。

ファクト変数の詳細入力ヘルパー (下のスクリーンショット) では、変数の プロパティは、全般的なセクションの下にリストされています。boolean プロパティの値は、true の場合はチェックボックスをチェック、false の場合はチェックを解除することにより指定します。



詳細入力ヘルパー内のアークのプロパティを確認するには、ダイアグラム内の to (destination) コンポーネントを選択します。アークの プロパティはアークセクションにリストされています。

フォーミュラ エディター内のコンテキストメニュー

フォーミュラ コンポーネントのコンテキストメニューは、コンポーネントの型により異なります。メニュー アイテムは、以下のセクションに整理されています :

- コンテンツの修正 (フォーミュラ フィルターの一部、カスタム関数のため) :例えば、*追加 追加 アスペクトルール*
- リレーションの修正 (サブ-アイテム のみのため) : *オーバーライド 削除 アーク*
- *ラベル 参照の追加*
- 新しい子コンポーネントの作成 (リレーションシップを含む) :例えば、*新しい フィルターの追加*
- コンポーネントの削除 (リレーションシップを含む)
- (コンポーネントの) *次の 前の発生 を検索*

メモ: コンテキストメニューにより作成または削除することのできる*コンセプト アスペクトルール*などのコンテンツアイテムは、追加セクション内の詳細入力ヘルパーに表示されます。

アサーション と アサーションセット

3つの型のアサーションがあります :

- 値アサーション

- 存在アサーション
- 整合性 アサーション

値アサーション

値アサーションは、フォーミュラリンクベース機能で頻繁に使用され、入力 XBRL インスタンス ファクトを XPath 式に対して、確認する方法を提供します。プロパティ *アスペクト モデル* そして *明示的なフィルター* をアイコンとして提供します。プロパティ `test` の値は、XPath 式です。

存在アサーション

存在アサーションは、書式の型、会社の ID、ファイルの ID などの静的な存在をチェックするために役に立ちます。 これは、プロパティ *アスペクト モデル* および *明示的なフィルター* をアイコンとして提供します。プロパティ `test` の値は、XPath 式です。

整合性アサーション

整合性 アサーションは、関連するフォーミュラにより作成された出力ファクトが、入力インスタンス内のすべてのアスペクト一致したファクトに対して整合性を持つかを指定します。 これは、Boolean プロパティ `strict` をアイコンとして提供します。*絶対的アクセスを受け入れる半径* および *プロポーショナル 受け入れ半径* のプロパティの値は XPath 式です。

アサーションが成立した/成立していないメッセージ

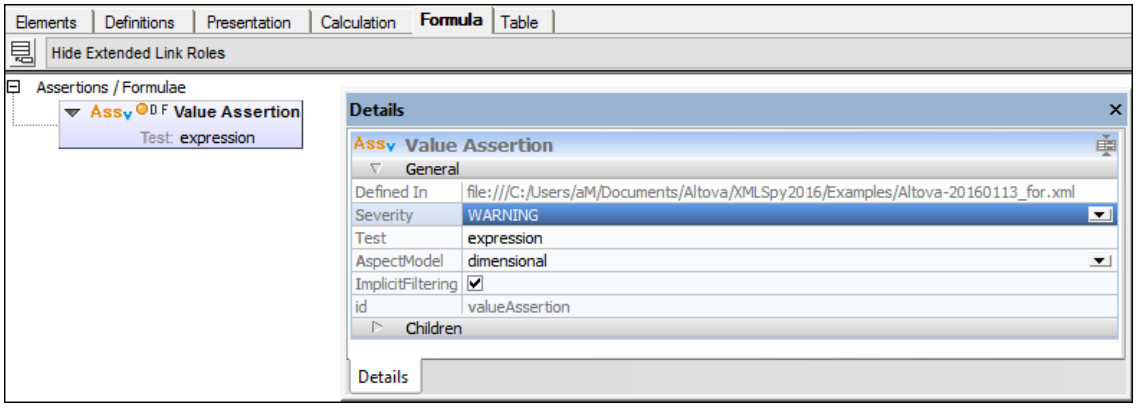
アサーションのこれらのサブコンポーネントは、アサーション評価をもつメッセージの関連を有効化します :成立した メッセージは、成功した評価が与えられ、成立していないメッセージには失敗したメッセージが与えられます。これらの メッセージは、各アサーションのコンテキストメニューを介して追加されます。

アサーション 条件が満たされない 重要度 リレーションシップ

アサーションは条件が満たされるか、満たされないかです。ですが、アサーションには異なる重要度のレベルのルールがあり、満たされないアサーションはその特定の満たされないレベルの重要度に従い、満たされないアサーションは分類されます。3つの標準の重要度のレベルがあります :エラー、警告、`OK`。デフォルトの重要度はエラーです。重要度は、アサーションが定義された重要度に関連しない場合、呼び出されます。

アサーション 重要度が満たされない 重要度 リレーションシップは、アサーションと定義された重要度リソースの1つの間のリレーションシップです。以下を持つ XLink アークと共に表示されます : (i) アークロールの値 `http://xbrl.org/arcrole/PR/2015-11-180/assertion-unsatisfied-severity`、(ii) 開始リソースとしてのアサーション、および (ii) 終了リソースとしての重要度リソース。

タクソノミーエディター内で、重要度リレーションシップはダイアグラム内のアサーションコンポーネントの 重要度 アイコンをクリックすることで指定することができます (*下のスクリーンショット参照*)。そして、表示されるポップアップから重要度のレベルを選択します。または、アサーションの詳細入力ヘルパー内で選択することもできます (*スクリーンショット参照*)。



アサーションセット

アサーションセットには、1つまたは複数のアサーションが含まれます。アサーションセットのコンテキストメニューは、アサーションセットへの各アサーションの追加を許可します。

フォーミュラ

フォーミュラは、出力 XBRL ファクトを構築するためのルールのセットを、フォーミュラの変数セット内で評価された変数に値を変換することで示します。変数の値は、入力 XBRL インスタンスおよびサポートされるDTS または フォーミュラを処理するアプリケーションからから取得されます。

値ルールは、ファクトに割り当てられる値を取得するXPath 式です。コンスタントなどの、シンプルな式 であることができます。または、変数セット、他の変数セットからの値のチェーン、および/または カスタムおよび内蔵された関数から計算された値の変数および パラメーターを参照する用語を含むことができます。

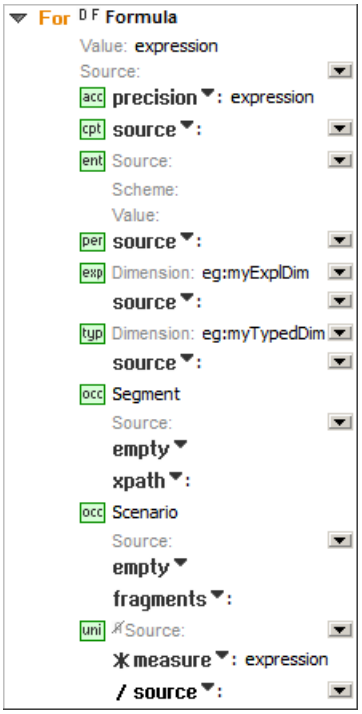
XBRL 内では、非 分数 数値 ファクトは、正確性 /小数点 属性のかたちで正確性についての情報と共に報告されます。ですから、フォーミュラは、出力ファクトのために アサーとされる正確性の決定を左右する正確性 ルールを含む可能性があります。

出力ファクト値と正確性を決定するルールと共に、フォーミュラは出力値を解釈するために必要なすべての出力アスペクトのために アスペクトルールを指定または暗黙に示します。出力コンセプト、出力コンテキストと 数値ファクトのための 単位の出力ユニットはすべて異なる型のアスペクトルーツです。ト

(部分的または全体の)アスペクトが、ソースを指定することにより、評価のバインドされた変数から取得される可能性があります。ソースを指定されることができます。ソースはルールにより指定される、またはフォーミュラのフォーミュ(または タプル)要素のソースから継承されることができます。on a ルール または 複数の ソースが存在する場合、アスペクトルールに近いものが優先されます。

フォーミュラが挿入されると、正確性 または アスペクトルールを持ちません (下のスクリーンショット)。

正確性 とアスペクトルールは、フォーミュラの コンテンツおよび追加された (または削除された)コンテキストメニューを介して定義されます。下のスクリーンショットは、可能性のあるすべての正確性とアスペクトルールを持つフォーミュラを表示します。



追加セクション内の詳細入力ヘルパーに正確性とアスペクトルールは表示されています。

正確性 ルール
型 :precision または decimals
値 :XPath 式

アスペクトルール
アスペクトルールは、型によりグループ化されています。

コンセプト ルール
型 :qname、expr、または source
値 :コンセプトの QName、XPath 式、または ソース変数 (またはカバーされていない QName)

エンティティ識別子 ルール
ソース:ソース変数 (またはカバーされていない QName)
スキーム値 :XPath 式

期間 ルール
型 :instant、duration、forever または source
値 :値の XPath 式 ,開始 終了 /ソース,値無し または ソース変数 (またはカバーされていない QName)

明示的なディメンション ルール
ディメンション : 明示的なディメンション ルールに影響されたディメンションの QName。
型 :qname、exp、omit または source
値 :メンバーの QName, メンバーの XPath 式 ,値無し または ソース変数 (またはカバーされていない QName)

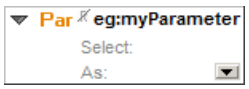
型指定された ディメンション ルール
ディメンション :明示的なディメンション ルールに影響されたディメンションの QName。
型 :xpath、value、omit または source
値 :XPath 式 ,XML 要素 ,値無し または ソース変数 (またはカバーされていない QName)

オープンコンテキスト コンポーネント ルール
OCC ルールは、型によりセグメント OCC ルールおよびシナリオ OCC ルールなどにグループ化されています。
ソース:ソース変数は、最初の OCC ルール内で定義されています。
For each OCC ルール:
型 :empty、fragments、または xpath
値 :値なし、XML 要素、または XPath 式

ユニット ルール
Boolean *フラグ引数* は、ソースアスペクト値が使用されるかを指定します。
For each ユニット 乗算 除算 ルール:
型 : *measure、 /measure または *source、/source
値 :メジャーの XPath 式 または ソース変数 (または カバーされていない QName)

パラメーター

パラメーターは、XPath 式内で参照されることができます。これは、*必要とされる フラグ*を提供します。設定されると、パラメーターは必須です。値は、処理アプリケーションにより与えられる必要があります。パラメータが必須ではない場合、また処理アプリケーションより値が与えられない場合、与えられた値 は、プロパティ*Select* 内で与えられた XPath 式 を使用して計算される可能性があります。オプショナル プロパティ*As* は、パラメータにより必要とされるデータ型を指定します。

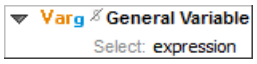


変数

変数は、入力データのバインドの方法、通常はファクトアイテムを、変数名により参照される、アサーションまたは フォーミュラ式などの名前に対して宣言します。 入力 ファクト アイテムをバインドする変数は、 ファクト変数であり、フィルターを使用して入力内で何がバインドされるかを宣言します。一般 変数は、中間の式の結果と他の種類の処理に使用されます。

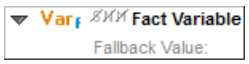
一般 変数

一般 変数は、Boolean プロパティのシーケンスとしてバインドをアイコンとして提供します。プロパティの選択の値は、XPath 式です。



ファクト変数

ファクト変数は、Boolean プロパティのシーケンスとしてバインド、およびアイコンとして一致を Nils として提供します。プロパティ フォールバック値 の値は、XPath 式です。

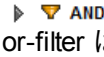


フィルター

フィルターは、入力 XBRL インスタンス内の変数が評価される際に使用されるXBRL インスタンスであるファクトのための選択条件を定義します。フィルターは、入力 ファクトに適用される条件を示します。 フィルターの一部には、サブラインに表示されたXML コンテンツを持ちます。

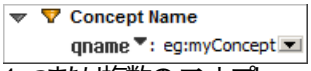
Aspect Cover
これらの フィルター は、フィルタリングを行わず、明示的なXPath 式を持ちません。これらは、(コンセプト および ディメンション などの)他のフィルタの後に処理または適用され、他のフィルターのアプリケーションにより引き起こされるアスペクトのカバー状態をオーバーライドします。
1 つまたは複数のアスペクトアイテム
型 :aspect, dim-qname/excl-dim-qname または dim-exp/excl-dim-exp
値 :アスペクト型 (enum), ディメンションのQName または XPath 式
アイテムは、入力ヘルパー詳細追加セクションに表示されています。

Boolean Filter
Boolean フィルター は、サブ-フィルターに関連付けられています。
and-filter は、サブフィルターにより表された条件をベースにしたファクトに一致します。



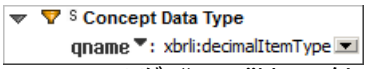
or-filter は、サブフィルターにより表された条件をベースにしたファクトに一致します。

Concept Name
Concept Name フィルター は、コンセプトの名前をベースにしたファクトに一致します。



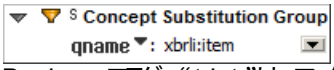
1 つまたは複数の コンセプト :
型 :qname または exp
値 :コンセプトのQName または XPath 式
コンセプトは、入力ヘルパー詳細追加セクションに表示されています。

Concept Data Type
Concept Data Type フィルター は、スキーマデータ型をベースにしたファクトに一致するために使用されます。



Boolean フラグ : 'strict ''は、ファクトのデータ型が割り出されるかを指定します。
型 :qname または exp
値 :データ型のQName または XPath 式

Concept Substitution Group
Concept Substitution Group フィルター は、XML スキーマ代替 グループをベースにしたファクトに一致するために使用されます。

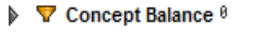


Boolean フラグ : 'strict ''は、ファクトのコンセプトが自身の@substitutionGroup 属性内の要素を直接指定するかを

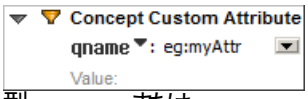
指定します。
型 :qname または exp
値 :代替 グループの QName または XPath 式

Concept Period Type
Concept Period Type フィルターは、@xbrli:periodType 属性により決定されたとおり、期間型またはインスタント型コンセプトのための値を報告するかどうかをベースにしてファクトに一致するために使用されます。

Concept Balance
Concept Balance フィルターは、@xbrli:balance 属性 を持つかどうか、また出金および入金の値があるかをベースにしたファクトに一致するために使用されます。

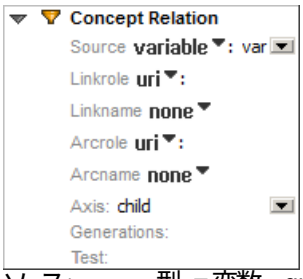


Concept Custom Attribute
Concept Custom Attribute フィルターは、存在または各コンセプトの宣言内の属性の値をベースにしたファクトに一致するために使用されます。



型 :qname または exp
値 :属性の QName または XPath 式

Concept Relation
Concept Relation フィルターは、効果的なリレーションシップの指定されたリンクロール URI 内ネットワーク内の、指定されたアークロール URI の、指定された軸の、指定された生成を含む、オプションのテスト式を満たす、ソースコンセプトに対してのコンセプトの効果的なリレーションシップをベースにした、ファクトに一致するために使用されます。



ソース: 型 =変数、qname または exp
リンクロール: 型 =uri または exp
リンク名 : 型 =none、qname または exp
アークロール: 型 =uri または exp
アーク名 : 型 =none、qname または exp

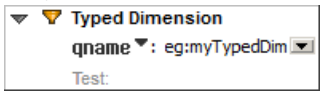
Explicit Dimension
フィルターディメンションの有効なメンバーのすべてのドメインの集合体内のすべてのドメインメンバーのセットとして与えられた

DTS のコンテキスト内で明示的なディメンションのドメインが定義されています。 Explicit Dimension フィルターは、明示的ディメンション値を持つ明示的ディメンションのドメイン内のドメインメンバーを持つファクトに一致するために使用することができます。

ディメンション型 :qname または exp
1 つまたは複数のメンバー :
型 :変数 ,qname または exp
メンバーは、入力ヘルパー詳細追加セクションに表示されています。

Typed Dimension

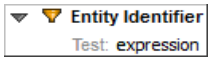
Typed Dimension フィルターは、型指定されたディメンションのための値をベースにしたファクトに一致するために使用されます。



ディメンション型 :qname または exp

Entity Identifier

Entity Identifier フィルターは、エンティティ識別スキームおよび/または エンティティ識別値の特性をベースにするファクトに一致するために使用されます。

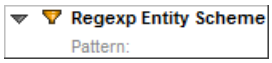


Specific Entity Scheme

Specific Entity Scheme フィルターは、スキーム識別フィルターのための報告値をベースにするファクトに一致するために使用されます。

Regular Expression Entity Scheme

Regular Expression Entity Scheme フィルターは、正規パターンエンティティ内のスキームをベースにしたファクトに一致するために使用されます。

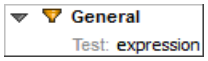


Specific Entity Identifier

Specific Entity Identifier フィルターは、フィルターにより与えられたエンティティ識別し、値を使用する報告値をベースにするファクトに一致するために使用されます。

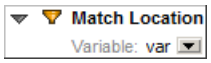
Regular Expression Entitiy Identifier
Regular Expression Entity Identifier フィルターは、エンティティ識別子の値のテキスト内の正規パターンをベースにしたファクトに一致するために使用することができます。

General
General フィルターはアスペクトをカバーしません。



Match Concept
Concept Matching フィルターに一致するコンセプトは、同じコンセプトのレポート値を持つファクトを選択するために使用されます。

Match Location
Location Matching フィルターは、同じ親の要素を持つファクトに一致するために使用されます。



Match Unit
Unit Matching フィルターは、同じユニットを持つファクトに一致するために使用されます。

Match Entitiy Identifier
Entity Matching フィルターは、同じエンティティ識別子とともにファクトを選択するために使用されます。

Match Period
Period Matching フィルターは、同じ期間を持つファクトを選択するために使用されます。

Match Dimension
Dimension Matching フィルターは、指定されたXBRL ディメンションのための同じ値を持つファクトを選択するために使用されます。

Match Complete Segment

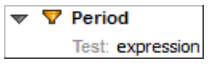
Complete Segment Matching フィルターは、 XBRL ディメンション仕様をベースにして解釈されないセグメントのコンテンツがある同じセグメントを持つファクトを選択するために使用されます。

Match Non-XDT Segment
Non-XDT Segment Matching フィルターは、比較からXBRL ディメンション仕様 コンテンツを除いた後の同じセグメントを持つファクトを選択するために使用されます。

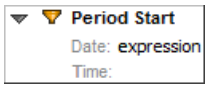
Match Complete Scenario
Complete Scenario Matching フィルターは、をベースにせずに解釈されたシナリオのコンテンツがある同じシナリオを持つ、ファクトを選択するために使用されます。

Match Non-XDT Scenario
Non-XDT Scenario Matching フィルターは、比較からXBRL ディメンション仕様 コンテンツを除いた後の同じセグメントを持つファクトを選択するために使用されます。

Period
Period フィルターは、計られたまたは期間に関連する条件の広い範囲をベースにした、ファクトに一致するために使用されます。

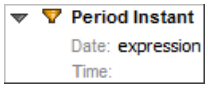


Period Start
Period Start フィルターは、計られたまたは期間の開始をベースにしたファクトに一致するために使用されます。



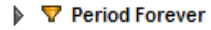
Period End
Period End フィルターは、計られたまたは期間の終了をベースにしたファクトに一致するために使用されます。

Period Instant
Period Instant フィルターは、計られたインスタントをベースにしたファクトに一致するために使用されます。

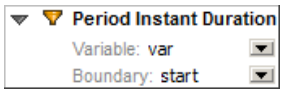


Period Forever

Forever フィルターは、繰り返し期間とともに報告されるファクトに一致するために使用されます。

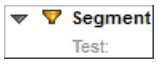


Period Instant Duration
Instant Duration フィルターは、他のファクトが報告された期間の開始または終了に一致するインスタントのあるインスタントでレポートされるファクトに一致するために使用されます。

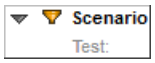


Relative
相対的なフィルターは、相対的なフィルターによりカバーされたアスペクトのためにファクトを選択するために使用されます。相対的なフィルタに一致するファクトは、評価される変数セット内の他のファクト変数の評価の結果である必要があります。

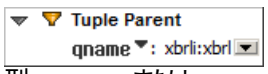
Segment
Segment フィルターは、指定された制約を満たす非 XDT コンテンツを持つ、ファクトに一致するために使用されます。非 XDT コンテンツは、XBRL ディメンション使用で定義されている明示的なまたは型指定されたディメンションをベースにしないセグメントコンテンツを参照します。



Scenario
Scenario フィルターは、指定された制約を満たす、非 XDT コンテンツを持つファクトに一致するために使用することができます。非 XDT コンテンツを参照します。ディメンション仕様で定義されていない、明示的なまたは型指定されたディメンションをベースとしないシナリオコンテンツを参照します。

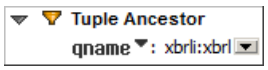


Tuple Parent
Parent フィルターは、指定された親要素を持つファクトを選択するために使用することができます。



型 :qname または exp

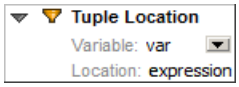
Tuple Ancestor
Ancestor フィルターは、指定された祖先要素を持つファクトを選択するために使用することができます。



型 :qname または exp

Tuple Sibling
Sibling フィルターは、他のファクトの兄弟であるファクトを選択するために使用することができます。

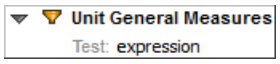
Tuple Location
Location フィルターは、他のファクトのロケーションに相対的な指定されたロケーションを持つ、ファクトを選択するために使用されます。



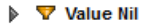
Unit Single Measure
Unit Single Measure フィルターは、単一の方法により指定されるユニットと共にレポートされているファクトに一致するために使用することができます。

型 :qname または exp

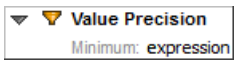
Unit General Measure
Unit General Measure フィルターは、ユニットメジャーの数値を含む条件をベースにした、ファクトを選択するために使用することができます。



Value nil
Nil フィルターは、nil とレポートされているファクトに一致するために使用することができます。

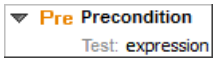


Value Precision
Precision フィルターは、最小の実際の正確性を持つことをベースにし、@decimal 属性の値により示されることのできるファクトに一致するために使用することができます。正確性フィルターは、フィルターが永続の最小の必須の正確性を示す場合、ファクトを選択しないことに注意してください。フィルターは、非数値または nil 値と共にレポートされるファクトを選択しません。



事前条件

事前条件は、バインドされた変数のセットが、フォーミュラ値 および出力ファクト または アサーション値テストまたは存在数をアクティブ化を決定する方法を提供します。

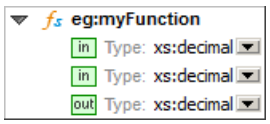


関数

カスタム関数は、XPath および XQuery 関数仕様、または、XBRL 関数 レジストリで定義されていない XPath 関数です。 カスタム関数は、XPath 式内で使用される可能性があります。

関数署名

関数署名 は、以下のスクリーンショットに表示されているとおりです。

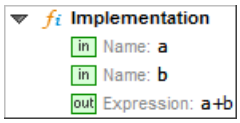


カスタム関数署名の子要素がある場合は、カスタム関数の 入力パラメーターのデータ型を指定します。カスタム関数の 入力パラメーターの順序は、カスタム関数署名の子要素のドキュメントの順序に一致します。

入力は、追加セクションの詳細入力ヘルパー内で表示されます。

関数の実装

関数の実装 は、以下のスクリーンショットに表示されているとおりです。



カスタム関数の実装 (CFI) は、カスタム関数の実装を構成するXPath 式 を示し、カスタム関数の出力を定義するため、関数入力のための名前を定義する子要素のシーケンスを含みます。

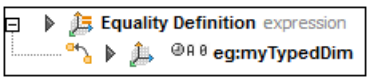
関数実装 リレーションシップは、カスタム関数署名 およびカスタム関数の実装間のリレーションシップです。関数の実装は、関数実装 リレーションシップのターゲットである必要があるため、対応する関数署名の下に表示されます。リレーションシップが存在しない場合、 (または署名異なるリンクロールの下で定義されている場合) 実装は、*関数* セクションの下に直接表示されます。

入力とステップは、追加セクション内の詳細入力ヘルパーに表示されています。

等値定義

等値定義は、型指定されたディメンションドメイン内の2つの値の等値の定義です。型指定されたディメンションドメイン定義は、型指定されたディメンションを宣言する XML スキーマ要素上の `@xbrldt:typedDomainRef` 属性などにより識別された、型指定されたディメンションのためのコンテンツモデルを定義するXML スキーマ内の要素です。型指定さ

れたディメンションドメイン定義および等値定義間のリレーションシップである、等値 定義 リレーションシップが、等値定義および対応する型指定されたディメンション間の逆 リレーションとして、表示されます。

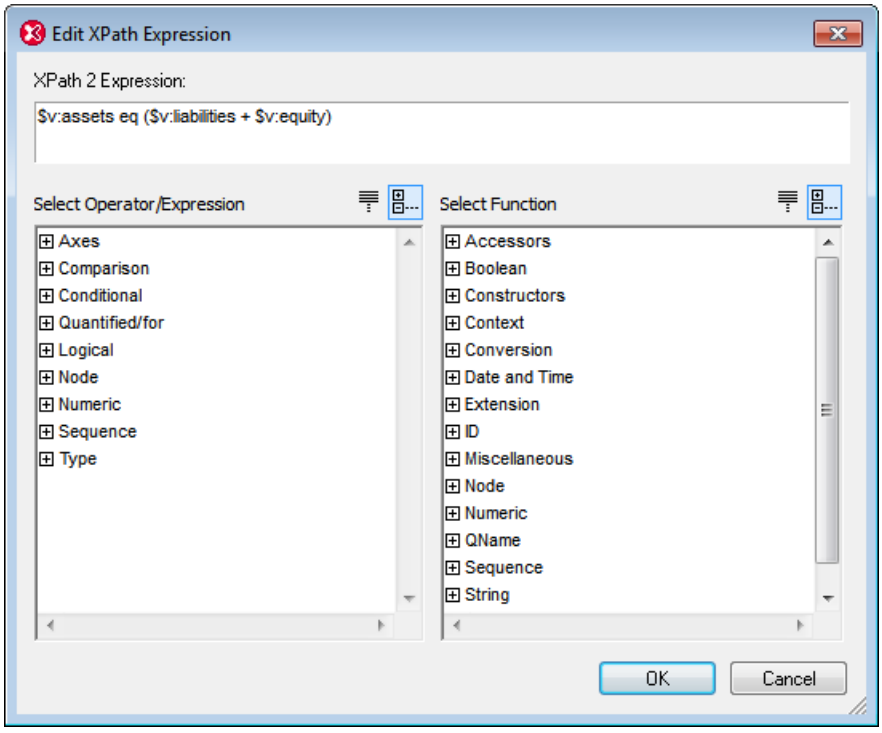


等値 定義 リレーションシップは、型指定されたディメンションをグローバル要素 入力ヘルパーから等値定義 コンポーネントへドラッグすることで構築することができます。等値定義または、まだ既存の等値 定義 リレーションシップに含まれていない可能性があることに注意してください。

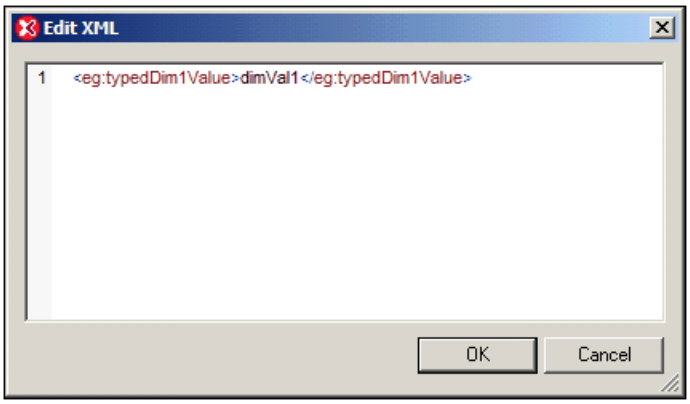
13.2.3 コンポーネントプロパティとコンテンツの編集

フォーミュラコンポーネンツのプロパティは、ダイアグラム内で、または詳細入力ヘルパー内で直接編集することができます。

ダイアグラム内で、コンポーネントに存在する場合は名前を入力すると適切な値のデフォルトのプロパティがプロパティ詳細テキストの横に灰色で表示されます。 コンポーネントをダブルクリックすると、展開します。プロパティをダブルクリックすると、プロパティを編集モードにします。プロパティまたはコンテンツがXPath 式を含む場合、XPath 式の編集がポップアップします。



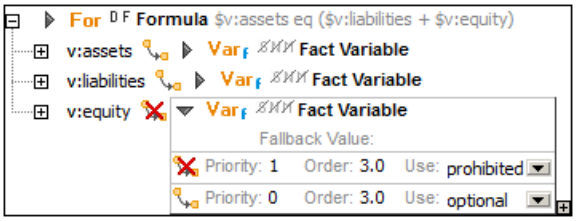
フォーミュラの型指定されたディメンション アスペクトルールの内の*値のコンテンツ*または フォーミュラの OCC アスペクトルール内の*フラグメントのコンテンツ*などのXML コンテンツの編集中、XML の編集ダイアログがポップアップします (*下のスクリーンショット*) 。(*フォーミュラを右クリックしてアスペクトルールを追加します。*)



XML テキストを入力して、「**OK**」をクリックすると、XML テキストは、プロパティ内のコンテキストとして入力されます。XML テキストが整形式でない場合、この効果へのメッセージがポップアップしますが、テキストは許可されません。

13.2.4 フォーミュラコンポーネントリレーション

2 つのフォーミュラコンポーネント間のリレーションシップは、フォーミュラコンポーネントを他のコンポーネントにドラッグアンドドロップしてリンクすることにより行います。リレーションシップはアークとともにダイアグラム内で、表示されています。(*下のスクリーンショット参照*)。



次の表示および編集の可能性があります：

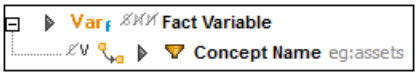
- コンポーネントの子の順序は、ドラッグアンドドロップにより子を移動させることにより変更することのできるアークプロパティの *順序* の値によります。(*上のスクリーンショット参照*).
- (プロパティを含む)子コンポーネントリレーションをコピーまたは移動するために、コンポーネントをの上または下にドラッグします。
- 既存の(親)コンポーネントのコンテキストメニューを使用して、新しいコンポーネントを作成するには、(アークである)リレーションシップは自動的に生成されます。
- 子コンポーネントのコンテキストメニュー内のコマンド**[アークのオーバーライドおよびアークの削除]**は、コンポーネントとその親の間のリレーションをオーバーライドまたは削除します。
- コンセプトリレーションに関しては、オーバーライドされたリレーションの複数のアークがサブラインに表示されます(*上のスクリーンショット参照*)。

メモ： フォーミュラコンポーネントリレーションのアークロールを変更することはできません。

変数セットリレーションシップ
変数セットリレーションシップは、(i) 変数セットリソース(値アサーション、存在アサーション、またはフォーミュラ)、(ii) 変数(ファクト変数または一般変数)またはパラメーター間のリレーションシップです。変数の名前またはパラメーターがアークアイコンの前に表示されます(下のスクリーンショット)。

変数フィルター リレーションシップ

変数フィルター リレーションシップは、ファクト変数とフィルター間のリレーションシップです。Boolean *プラグ コンプリメント*(ダイアグラム内の、c アイコン)が設定され、リレーションシップは、変数フィルター リレーションシップにより補修されます。Boolean *プラグ カバー*(ダイアグラム内の、v アイコン)が設定されると、リレーションシップは、カバーする変数フィルター リレーションシップとなります (*下のスクリーンショットに表示*)。この場合は、フィルターは、フィルターされるファクトのアスペクトをカバーします。



変数セット-フィルター リレーションシップ

variable-set-filter リレーションシップ (*下のスクリーンショット参照*) は、変数セット リソースとフィルター間のリレーションシップです。variable-set-filter リレーションシップに含まれるフィルターは、基本的に、変数セット内のファクト関数に関連しており、関連するリソースにより定義されています。Boolean *プラグ コンプリメント*は、変数がフィルター コンプリメントを使用するか指定します。variable-set-filter リレーションシップによりファクト変数と関連する場合、通常アスペクトをカバーしません。

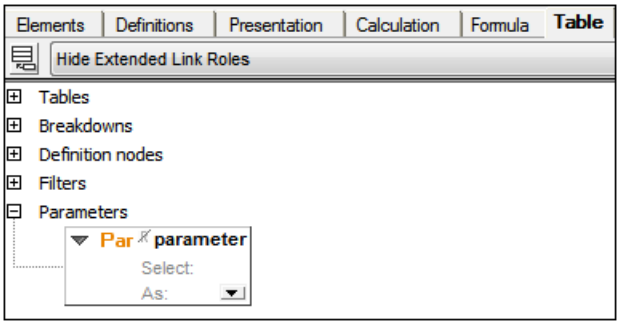
テーブル レイアウトプレビュー内で視覚的にフォーミュラをバインドする

XBRL タクソノミの開発者は、XBRL フォーミュラをポイントしてクリックすることにより作成するXBRL テーブル プレビューを活用することができます。 この機能については、テーブル レイアウトプレビュー内でフォーミュラをバインドするセクションで説明されています。

13.2.5 フォーミュラ パラメーター

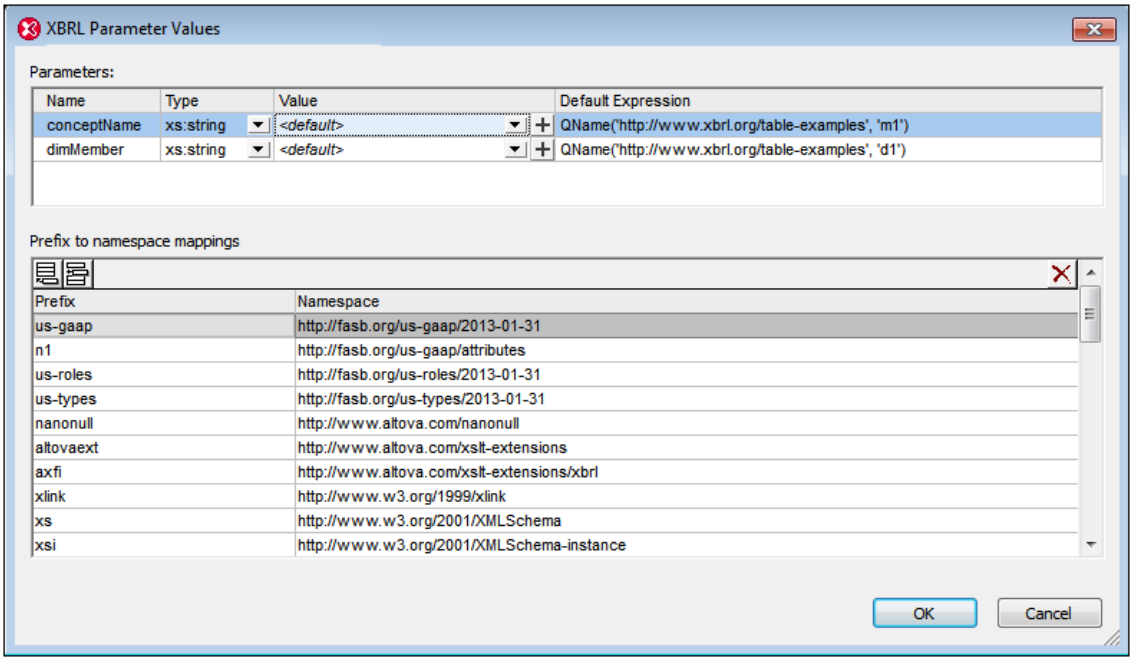
XBRL パラメーターは、フォーミュラおよびテーブル定義内のXPath 式 で使用することができます。フォーミュラ パラメーターとして使用される、(フォーミュラ リンクベース内に存在する)パラメーターは、 フォーミュラ タブで作成されます。 (テーブル リンクベース内に存在する)テーブル パラメーターはテーブル タブ内で作成されます。フォーミュラ パラメーター およびテーブル パラメーターは、ローカル または グローバルであることができます。ローカル パラメーター は、基本的には、作成時に対応するコンポーネント (フォーミュラ または テーブル)にリンクされたグローバル パラメーターです。 ローカル パラメーターは、コンポーネント (フォーミュラ または テーブル)を右クリックして、**[新しいパラメーターの追加]** を選択することにより、作成されますが、グローバル パラメーターは、対応するタブの空白のエリアを右クリックして **[新しいパラメーターの追加]** を選択して作成されます。これにより、パラメーターという名前の新しいパラメーターがダイアグラム内に追加されます (*下のスクリーンショット グロー*

バル パラメーターを表示しています)。パラメーター名を変更するには、名前をダブルクリックして、編集します。



すべてのパラメーターには、必須のフラグがあります。設定される場合、パラメーターは、必須で、処理アプリケーションにより値が提供される必要があります。パラメーターが必須ではなく、に処理アプリケーションより値が提供されていない場合、与えられた値は、プロパティ内 *Select* で与えられている XPath 式 を使用して計算される可能性があります。XPath 式を入力するには、*Select* フィールド内をダブルクリックします。この値がパラメーターのデフォルトの値になります。オプショナルプロパティ*As* は、パラメーターにより必要とされるデータ型を指定します。コンボボックスのドロップダウンリストからデータ型を選択します。

テーブルパラメーターとして使用されるパラメーターの場合、パラメーターのデータ型を編集し、デフォルトの値をオーバーライドするパラメーター値 を提供することができます。これを行うには、**[XBRL | パラメーターの値]** をクリックします。そして、表示されるダイアグラム内 (*下のスクリーンショット*) にパラメーター値を入力します。この値がデフォルトの値をオーバーライドします。 テーブルパラメーターとして使用される、パラメーターは、複数の値をとることができ、パラメーターのための追加のパラメーターの値を*値* 列内の**[+]** アイコンをクリックすることにより追加できます。



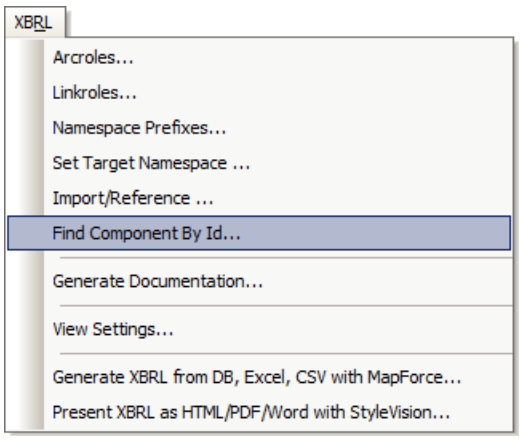
このダイアログで割り当てられたグローバルパラメーターの値は、テーブルパラメーターのみのために評価されます。 フォーミュラで使用されるパラメーターの値は、このダイアログで編集されることはできません。

13.2.6 フォーミュラ コンポーネントの検索

フォーミュラ コンポーネントは、ID を使用して、または、ドキュメント内のコンポーネントの発生をナビゲートすることにより検索することができます。

ID によりフォーミュラ コンポーネントを検索する

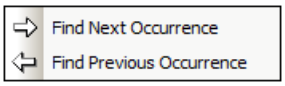
同じ型の複数のコンポーネントを含む大きなフォーミュラ リンクベースを持つタクソノミは、ID によりコンポーネントを検索する際に役に立ちます。メニューコマンド **[XBRL | ID によるフォーミュラ コンポーネントの検索]** は ID による検索を有効化します。



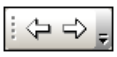
コマンドをクリックすると検索する ID を問うダイアログがポップします。

コンポーネントの発生を検索する

フォーミュラ コンポーネントのほとんどは、フォーミュラ リンクベース ダイアグラム内に複数回表示されています : (i) 適切な セクション ノードの下に直接存在する定義、(ii) コンポーネントへのすべての参照 (*リレーションシップを介して*)。コンポーネントの コンテキスト メニュー内のコマンド **[次の発生を検索]** および **[前の発生を検索]** は、フォーミュラ コンポーネントが参照されているすべての箇所にナビゲートします (*下のスクリーンショット*)。



これらのコマンドは、をツールバー アイコン介してアクセスすることができます。 (*下のスクリーンショット*).



コンポーネントの定義に達成すると、その効果が表示されます。

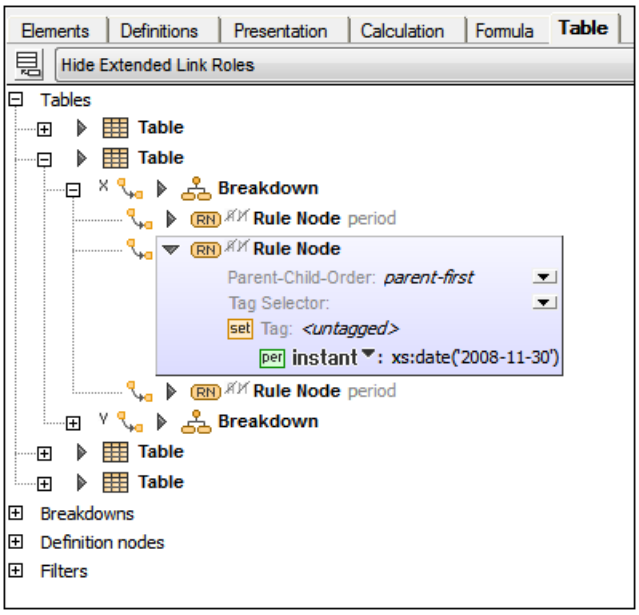
13.3 XBRL テーブル定義エディター

XBRL 仕様はリンクベースの表示を補完するテーブルリンクベースを提供します。テーブルは、XBRL タクソノミ内で定義されるコンセプトのビューを定義する他の方法を与えます。プレゼンテーションリンクベースがするように、コンセプトを階層構造で表示するのではなく、複数の軸を使用してテーブルを定義します。軸のコンポーネントは、個別のアイテムに制限されておらず、ディメンション、期間、単位、エンティティ、タクソノミにより表されることのできる金融ファクトとして使用することのできる他のプロパティの組み合わせとして定義されることもできます。XBRL テーブルリンクベースの構文とセマンティックの概要情報は、XBRL テーブルリンクベース概要 1.0 および2014 年 3月 18日 版テーブルリンクベース 1.0勧告により見つけることができます。

XMLSpy は、2014 年 3月 18日 版テーブルリンクベース 1.0勧告に従い、名前空間 http://xbrl.org/2014/テーブル を使用します

標準のXBRL リンクベース (プレゼンテーション、計算、定義)は、標準の拡張されたリンク内のロケーターと標準のアークを介して、コンセプト間のリレーションを定義しますが、テーブルリンクベースは、コンポーネント (テーブル、ブレークダウン、定義ノード、など)を含み、ジェネリックな拡張されたリンク内の リソースおよびジェネリックな アークを介してその関係を定義します。テーブルリンクベース仕様は、3つのモデルのシーケンスを定義し、各モデルの次のモデルへの変換を処理します。3つの モデル:定義 モデル、構造モデルおよびレイアウト (またはレンダリング)モデル。定義モデルは、テーブルリンクベースのセマンティックコンテンツのモデルです。 テーブルは、軸により定義されます。また、軸定義は、代わりに定義ノードのツリーにより作成されます。

XMLSpy のXBRL テーブル定義 エディターは、アプリケーションのXBRL タクソノミエディターの一部として実装されます。XBRL ビューの**テーブル タブ** で使用することができます (*下のスクリーンショット参照*)。

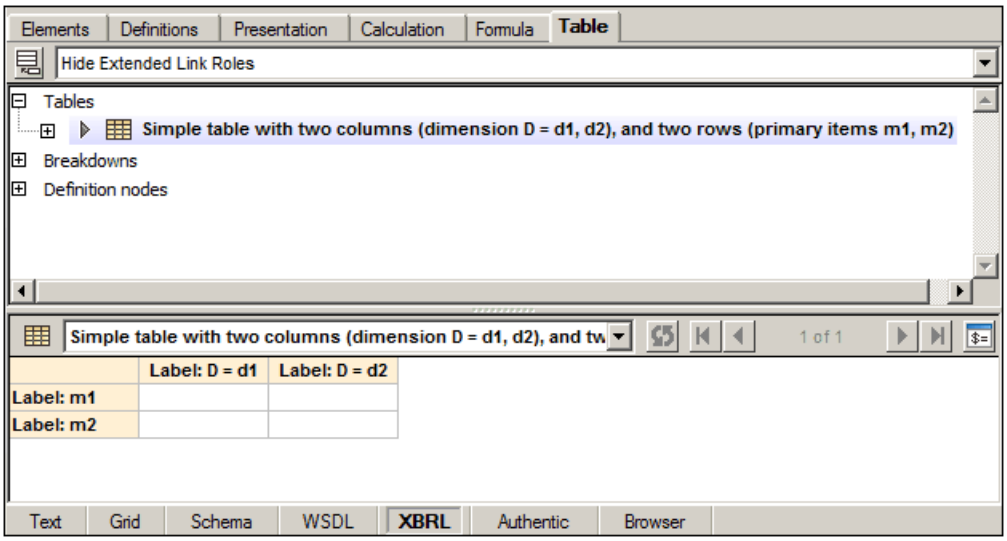


テーブル タブは、概要 入力ヘルパーと詳細入力ヘルパーと共に、テーブル定義を編集するために使用されます。詳細入力ヘルパーは、テーブルコンポーネントのプロパティとコンテンツを編集するために使用されますが、テーブル概要 入力ヘルパーは、XBRL テーブルのためのデフォルトのリンクベースを設定するために使用されます (テーブル定義が保存されるファイルはデフォルトで保存されます)。 タブ自身がテーブル定義の直接の編集を有効化することができます。

XBRL テーブル レイアウトプレビュー

テーブル定義のレイアウトをプレビューするために、XBRL タクソノミエディターは、XBRL ビューのテーブルタブ内にXBRL テ

ーブルレイアウトプレビューペインを与えます (*下のスクリーンショット参照*)。ダイアグラム内で、テーブル または テーブルコンポーネントが選択されると、テーブルのプレビュー が、ダイアグラム下のテーブルレイアウトプレビューペインに表示されます (*下のスクリーンショット参照*)。代わりに、プレビューペインの コンボボックスのドロップダウンリストから、テーブルを選択することができます。これは、テーブルリンクベース内のテーブルのリストです。



プレビューの機能の詳細に関しては、次のセクションを参照してください:

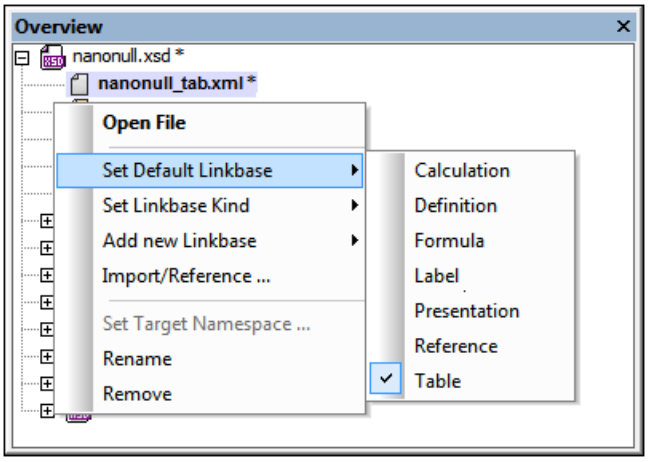
- テーブル構造
- テーブルパラメーター
- テーブル レイアウトプレビュー

13.3.1 テーブル リンクベース および リンクロール

標準のXBRL リンクベース (定義、プレゼンテーション、計算) は、標準の拡張されたリンク内のロケーターと標準のアークを介してコンセプト間のリレーションシップを定義しますが、テーブル リンクベースは、テーブルコンポーネント (テーブル、ブレークダウン、定義ノードなど) およびそのリレーションシップを定義します。これらの定義は、ジェネリックな 拡張されたリンク内のリソースおよびジェネリックな アークを介して 指定されることができます。

テーブル リンクベースの追加

概要入力ヘルパー内 (*下のスクリーンショット*) のタクソノミファイル または既存の リンクベースを右クリックして、**新規リンクベースの追加 | テーブル** を選択します。追加されたリンクベースは、デフォルトテーブル リンクベース ファイルとなります。デフォルトテーブル リンクベース ファイルは、タクソノミファイル が保存される際に保存される新しいテーブル定義が入るファイルです。他の テーブル リンクベース ファイル をデフォルトテーブル リンクベースにするには、右クリックして **デフォルトリンクベースの設定 | テーブル** を選択します (*下のスクリーンショット参照*)。



デフォルトリンクベースは、太字で表示され、変更されているが保存されていないリンクベースは、アスタリスクと共にマークされています。

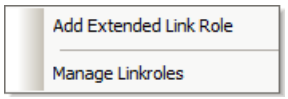
テーブルリンクベースは、テーブルタブ内に表示されます。

メモ: テーブルリンクベースが存在しない場合に[テーブルコンポーネントがタクソノミに追加](#)されると、テーブルリンクベースは自動的に作成されます。

リンクロール

標準の拡張されたリンクのケースと同様（定義、プレゼンテーション、計算のための）ジェネリックなリンクは、ネットワークを区切る同じタイプのリレーションシップを区切る拡張されたリンクロール値を定義する必要があります。同じリンクロールを持つすべてのジェネリックな拡張されたリンクは、*異なるリンクベース ファイルにぞんざいする場合*でも、リンクロールノード テーブルタブ内のダイアグラムにこの下に結合されます。

ジェネリックな リンクロールは、背景エリアのダイアグラム コンテキストメニューを介して作成することができます (*下のスクリーンショット*)。しかしながら、この コンテキスト メニューは、テーブル タブの表示 オプション コンボボックスが、*すべての拡張されたリンクロールの表示*に切り替えられている場合のみ表示されます。場合のみ表示されます。



この メニューは、 ツールバーアイコン、*拡張リンクの追加 /リンクロールの管理*、でも使用することができます。リレーションシップ ネットワークは、テーブル リンクベースにとってそれほど重要ではなく、テーブル タブのデフォルトビューは、ンクロールを非表示にする、*拡張リンクロールを隠す*です。代わりに、リンクロール無しのテーブルコンポーネントを表示します。

拡張されたリンクロールが作成された際にデフォルトテーブル リンクベース ファイルが存在しない場合、デフォルトテーブル リンクベース ファイルは自動的に作成されます。リンクロールの作成時にデフォルトリンクベースファイル内にリンクロールが存在しない場合、デフォルトリンクベース ファイル内にンクロールは自動的に作成されます。

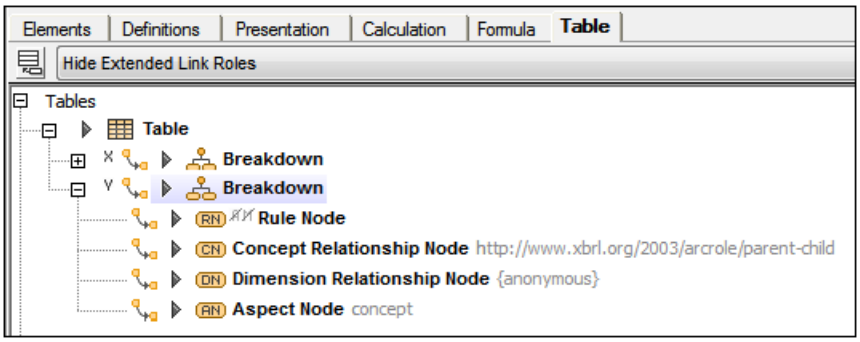
13.3.2 テーブル構造

テーブル定義内のテーブルの構造がテーブルの軸 (X、Y、Z) により定義されている場合、それぞれ 1 つまたは複数の ブレークダウン コンポーネントに対応します (*下のスクリーンショット参照*)。

- X とY 軸 は、それぞれ生成されたテーブルの列と行に対応します。 X とY 軸 セクションで説明されています。
- Z 軸 が定義されている場合、個別のテーブルとして表示されています。Z 軸のセクションを参照してください。
- 各ブレークダウンコンポーネントは、複数の テーブル定義ノードを含むことができます (下のスクリーンショット参照)。これらは、定義ノードの異なる型のです ：

  – ルールノード (RN 下のスクリーンショットのアイコン)
  – コンセプトリレーションシップノード (CN)
  – ディメンションリレーションシップノード (DN) および
  – アスペクトノード (AN)。

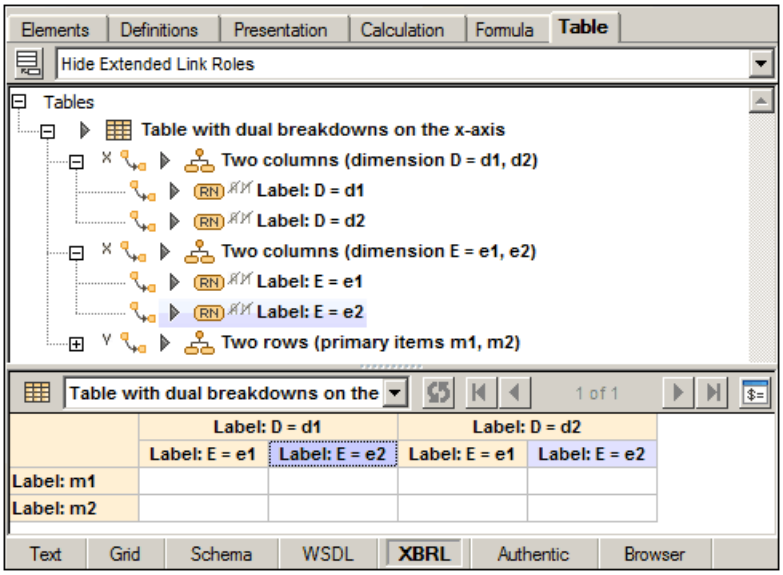
  定義ノードの構造 プロパティの詳細に関しては、定義ノードのセクションを参照してください。



複数のブレークダウンのためのプロジェクション

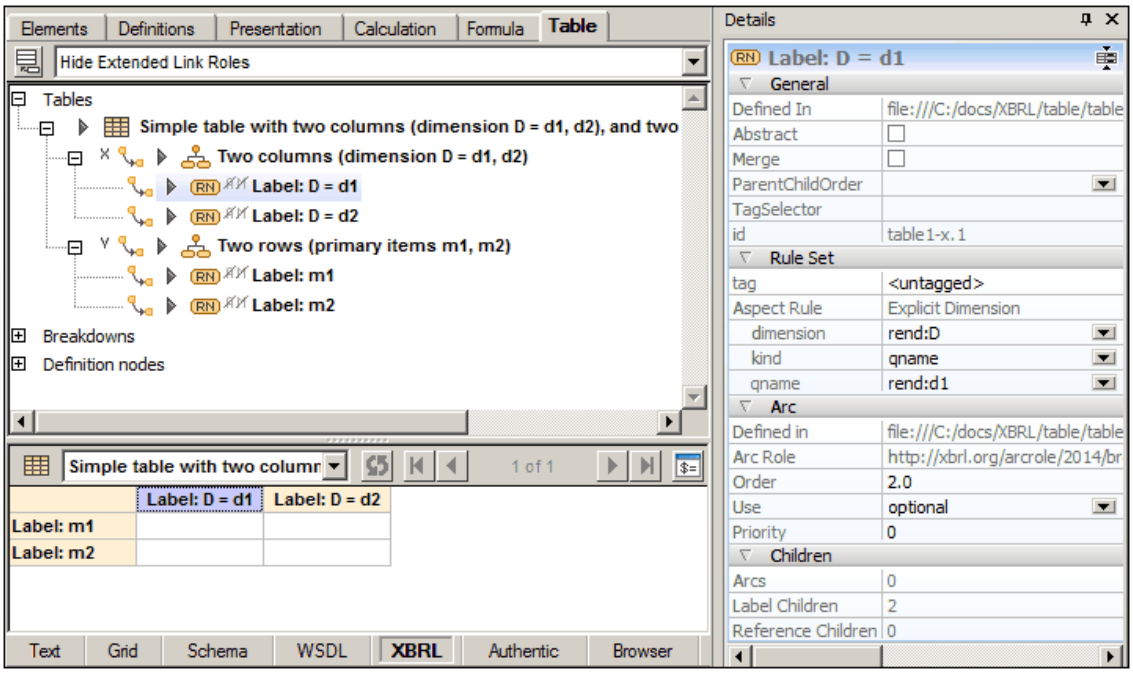
複数の独立したブレークダウンは、単一の テーブル軸に関連している可能性があります。複数のブレークダウンを単一の"効果的な" ブレークダウンに組み合わせるためのメカニズムは、プロジェクションと呼ばれます。 単一の軸のための複数のブレークダウンの相対的な優先順序は、各ブレークダウンの `@order` 属性 により決定されます。ブレークダウンは、ツリーとしてビジュアル化されます。最初の ブレークダウンの各リーフのため、 2番目の ブレークダウン全体が添付され、再帰的に行われます。

下のスクリーンショットでは、例えば , X 軸のために 2つの ブレークダウンがあります :**`dimension D`** は、**`dimension E`** よりも高い優先順位でオーダーされます。ですから、 **`dimension D`** (**`d1`** および **`d2`**) の各リーフには、**`dimension E`** のツリー全体が添付されます。X 軸 は、列を生成するため、これらのブレークダウンはテーブルの列構造のためのプロジェクションを作成します。下のスクリーンショットのテーブルレイアウトプレビューを参照してください。



X とY 軸

X とY 軸 は、*それぞれテーブルの列と行を定義します。それぞれの軸で、1つまたは複数の階層的ブレークダウンが定義されます。(下のスクリーンショット参照)。ブレークダウン対応する単一の軸は、単一の"効果的な" ブレークダウンに解決されます。軸のためのブレークダウンが1つしか存在しない場合、このブレークダウンは、効果的な ブレークダウンとなります。軸のために複数のブレークダウンが定義されている場合、解決 メソッドについては、さらに詳しく複数のブレークダウンのためのプロジェクション*で説明されています。



次の軸 に関連するプロパティ編集機能に注意してください:

- テーブル定義内で、X 軸 は、生成されたテーブルの列に対応しますが、Y 軸は、テーブルの行に対応します (*上の*

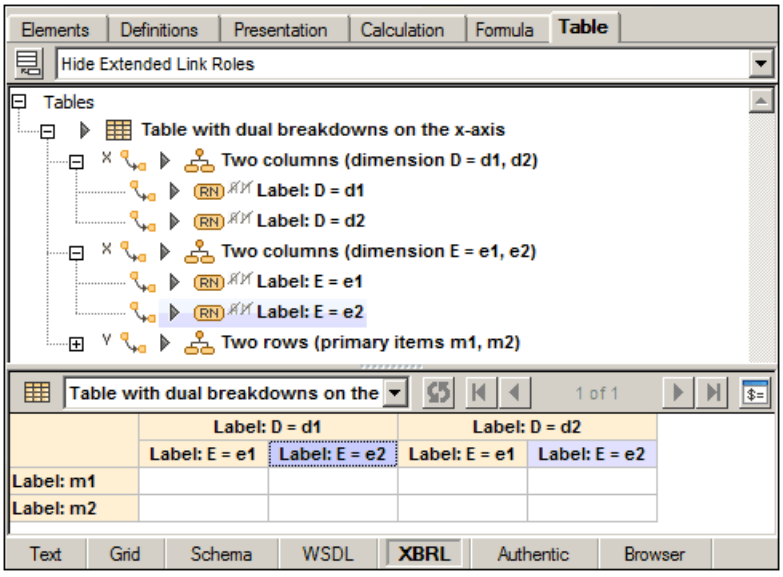
テーブル レイアウトプレビューのスクリーンショットを参照してください)。

- それぞれの軸 は、 1つまたは複数の ブレークダウンを持つことができます (複数のブレークダウンのためのプロジェクション を参照)。
- 生成されたテーブルの各セルの背景色は、オレンジ黄色です。テーブル定義内で、 テーブル定義内で セルは、軸のブレークダウン内の定義ノードに対応します。
- セルが選択されると、 対応する定義ノードも選択されます。またその逆もあります。 選択された コンポーネントのセルの背景色は、紫色です。
- コンポーネントが選択されると、詳細入力ヘルパーにプロパティが表示され、そこで編集されることができます。 (*上のスクリーンショット参照*).
- データセルには、背景色はありません。タクソノミ自身がファクトを含まないため、これらは常に空白です。
- セル制約 は、軸 (存在する場合はタグセレクターを使用して) から、計算され、詳細入力ヘルパーの制約タブに表示されます。セクション内のスクリーンショット Z 軸 を参照してください。

複数のブレークダウンのためのプロジェクション

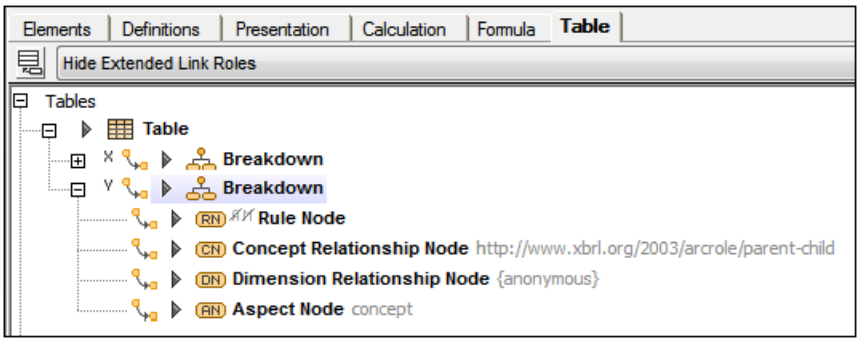
複数の独立した ブレークダウンは、単一の テーブル軸に関連している可能性があります。複数のブレークダウンを単一の効果的な ブレークダウンに組み合わせるためのメカニズムは、プロジェクション と呼ばれます。 単一の軸のための複数のブレークダウンの相対的な優先順序は、各ブレークダウンの `@order` 属性 により決定されます。ブレークダウンは、ツリーとしてビジュアル化されます。最初の ブレークダウンの各リーフのため、 2番目の ブレークダウン全体が添付され、再帰的に行われます。

下のスクリーンショットでは、例えば, X 軸のために 2つの ブレークダウンがあります :**`dimension D`** は、**`dimension E`** よりも高い優先順位でオーダーされます。ですから、 **`dimension D`** (**`d1`** および **`d2`**) の各リーフには、**`dimension E`** のツリー全体が添付されます。X 軸 は、列を生成するため、これらのブレークダウンはテーブルの列構造のためのプロジェクションを作成します。下のスクリーンショットのテーブルレイアウトプレビューを参照してください。



定義ノード

各ブレークダウン コンポーネントは、複数の テーブル定義 ノードを含むことができます (*下のスクリーンショット参照*)。

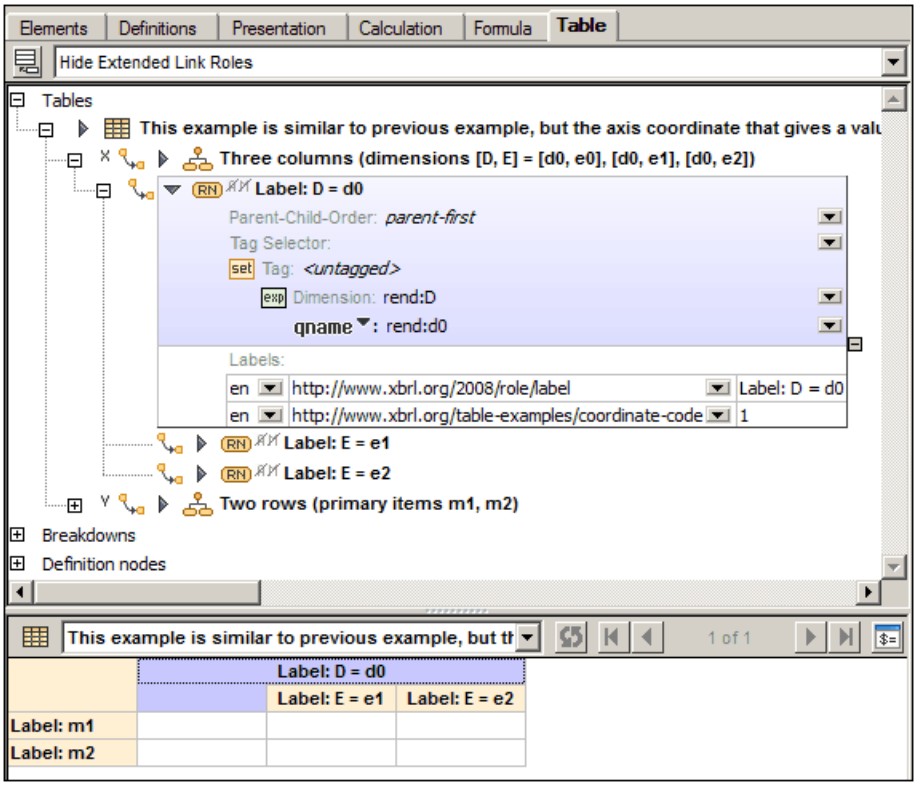


定義ノードの異なる型があります：

- ルールノード(RN 上のスクリーンショット内のアイコン)
- コンセプトリレーションシップノード(CN)
- ディメンショナルリレーションシップノード(DN)
- アスペクトノード(AN)

ルールノード

ルールノードは、 1つまたは複数の アスペクトのための アスペクトルールを定義します。:コンセプト、期間、ユニット、 エンティティ識別子、ディメンション、 またはオープン コンテンツアスペクト。定義 ツリー内のコンポーネントは、ルールノードが抽象的または子のない場合、レイアウト内の1つのセルに対応します。その他の場合は、 レイアウトは、 ルールノードのプロパティ `parentChildOrder` の効果的な値により位置が決定される、追加ロールアップセルを含みます。

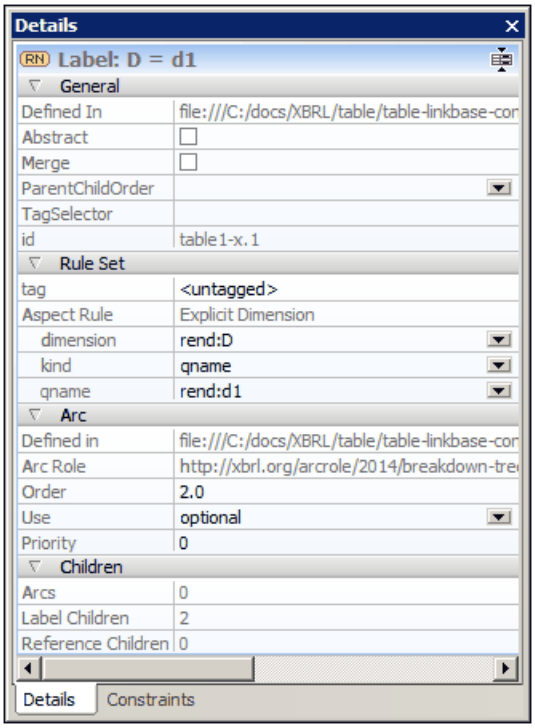


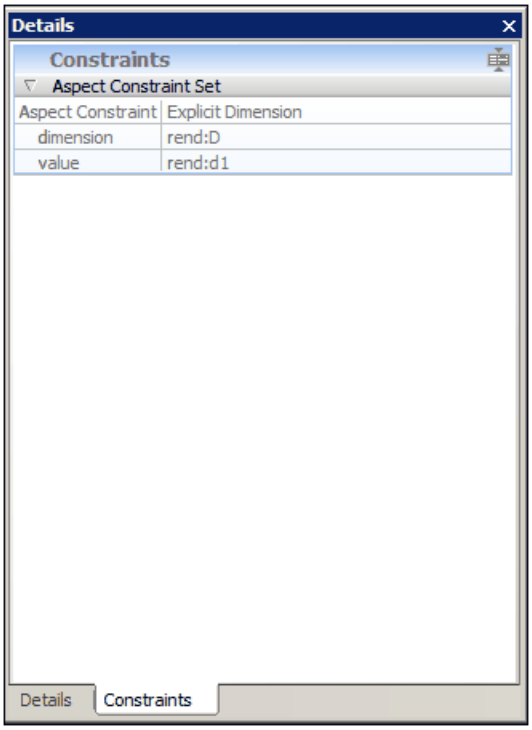
レイアウトセルのヘッダーは、 ルールノードから次のように計算されます：

- ノードがユーザー定義 ラベルに関連付けられている場合、このラベルのテキストが表示されます。
- 単一の アスペクト制約 を定義する ラベル および ノードが存在しない場合、(コンセプト、ディメンション、ユニット、エンティティ識別子、または期間) その値が表示されます (例えば、コンセプトの修飾名)。
- その他の場合は、静的テキスト `Rule node` が使用されます。

詳細入力ヘルパー

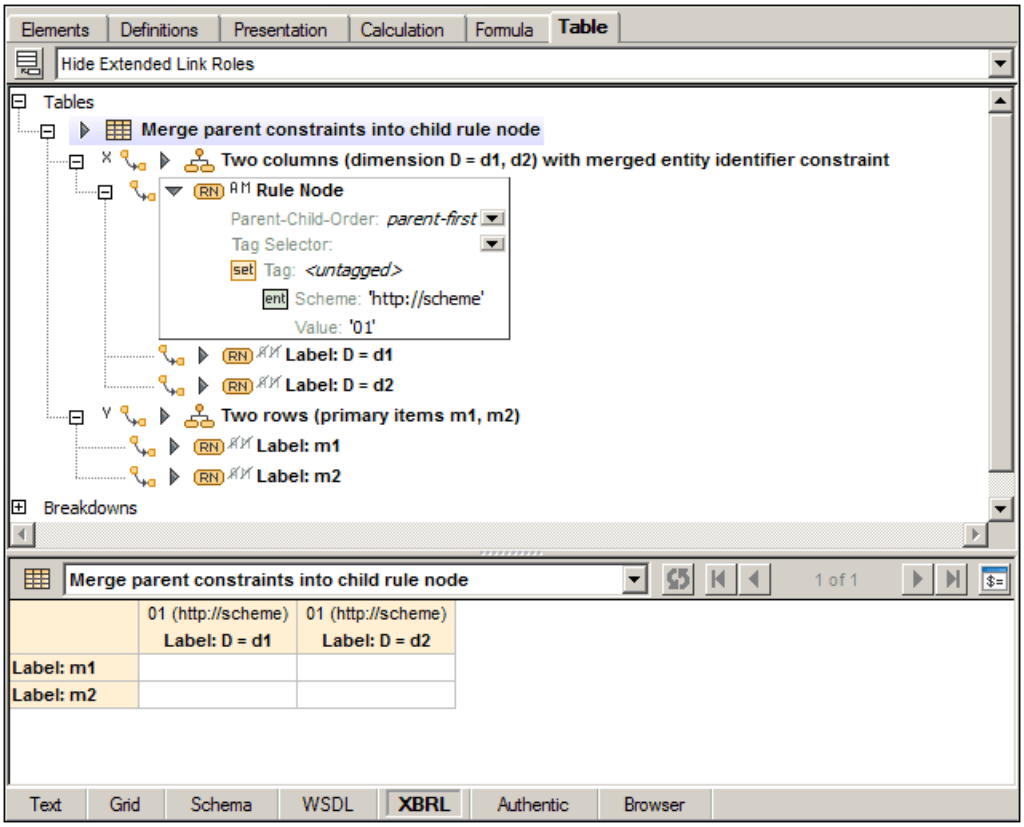
ノードのプロパティは、詳細入力ヘルパーの詳細タブに表示されています (左下のスクリーンショット)。制約タブは、ルール ノードの アスペクトルールから計算された アスペクト制約 セットの読み込み専用のビューを提供します (右下のスクリーンショット)。





マージされたルールノード

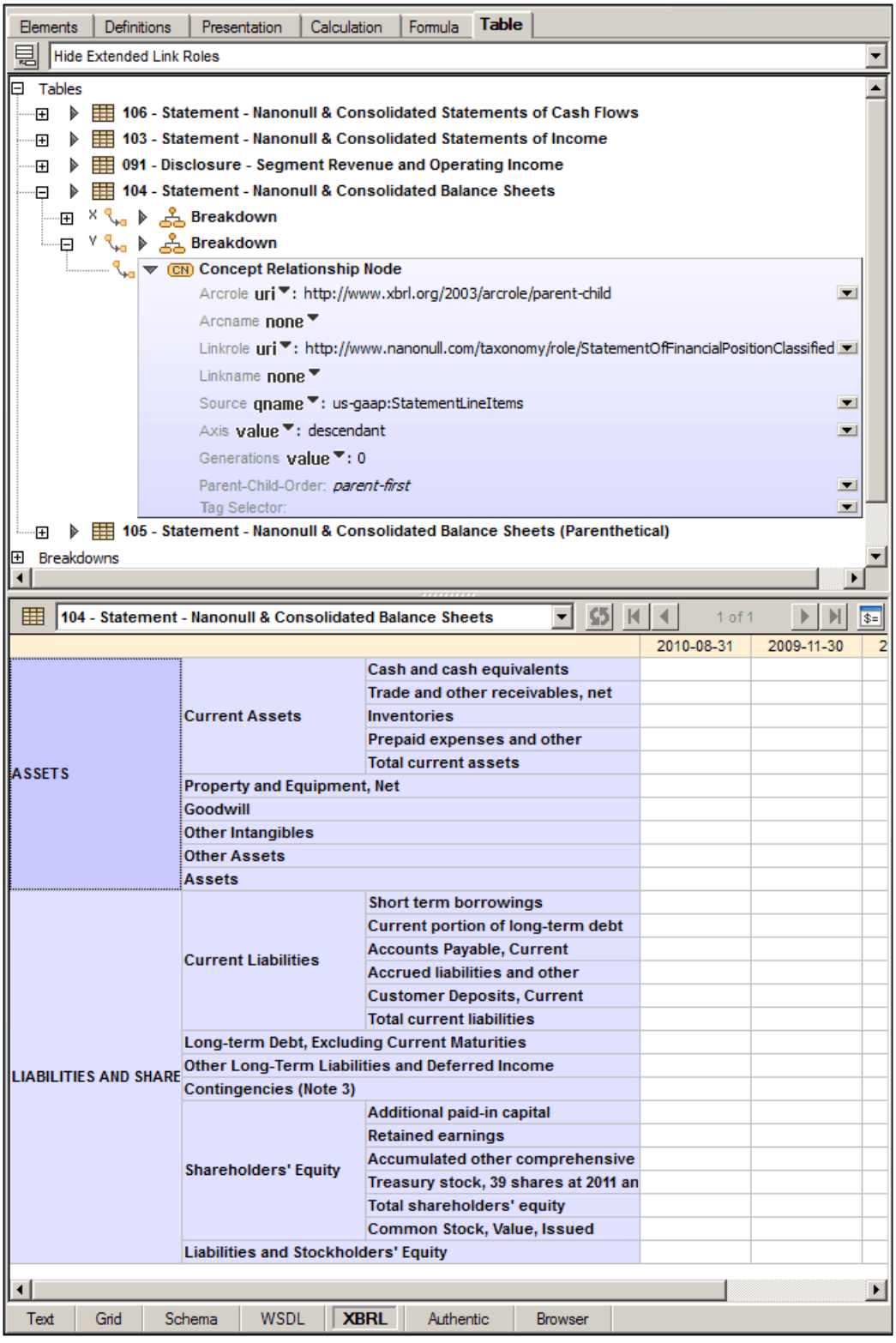
マージされたルールノードは、子により作成されたすべての制約セットの制約すべてに与えられるすべての子に適用される追加プロパティを示します。 (*下のスクリーンショット参照*)。

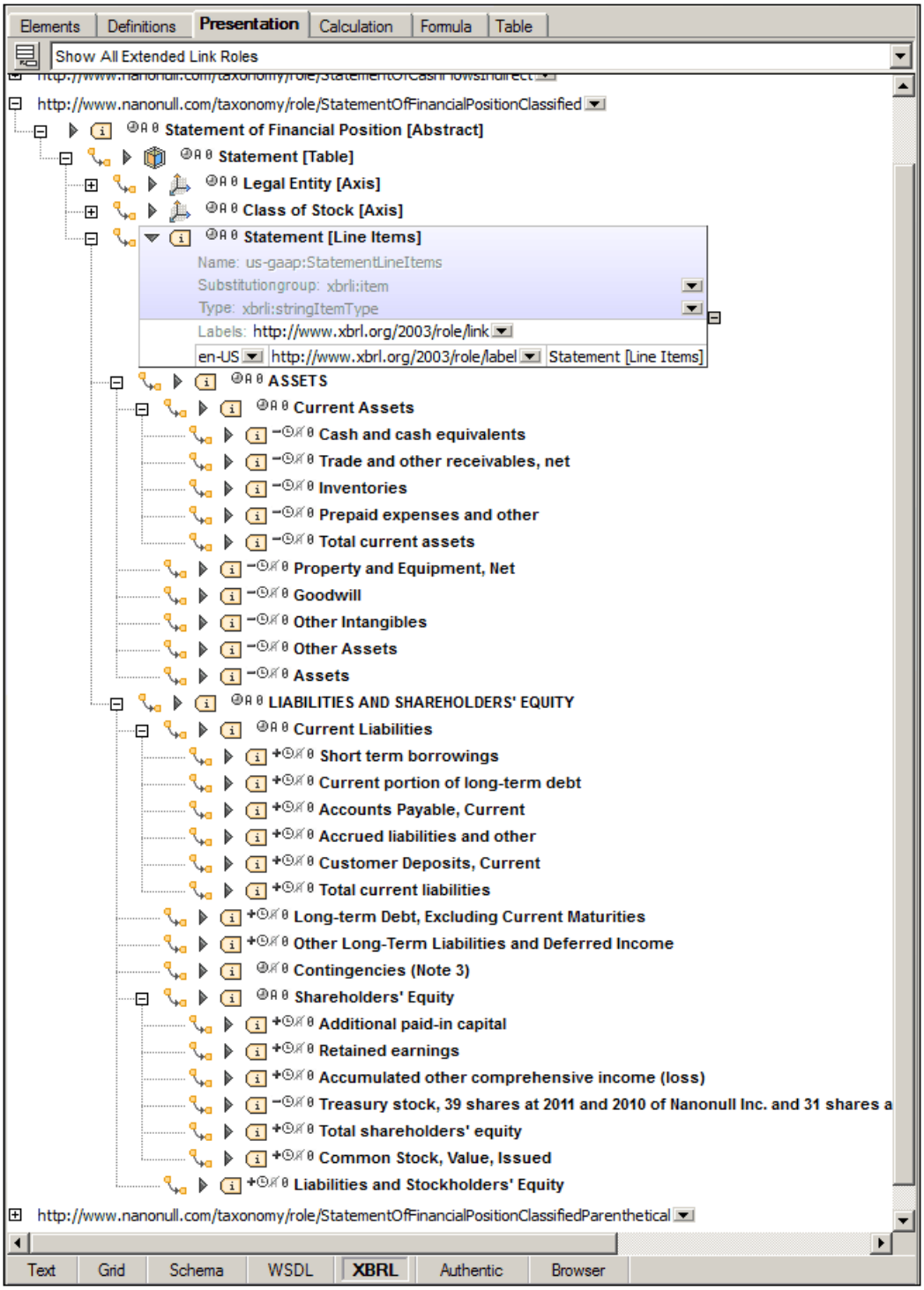


リレーションシップノード

コンセプト または ディメンション リレーションシップノードは、コンセプトのネットワーク または DTS 内の明確なディメンション メンバーにより定義された構造ノードのツリーになります。ですから、定義ツリー内のコンポーネントは、レイアウトのセルのブロックに対応します。

左のスクリーンショットは、を コンセプト リレーションシップノードを含むテーブル定義表示します。右のスクリーンショットは、コンセプトの対応するネットワークを表示します。この場合は、プレゼンテーション リンクベース内で定義されています。

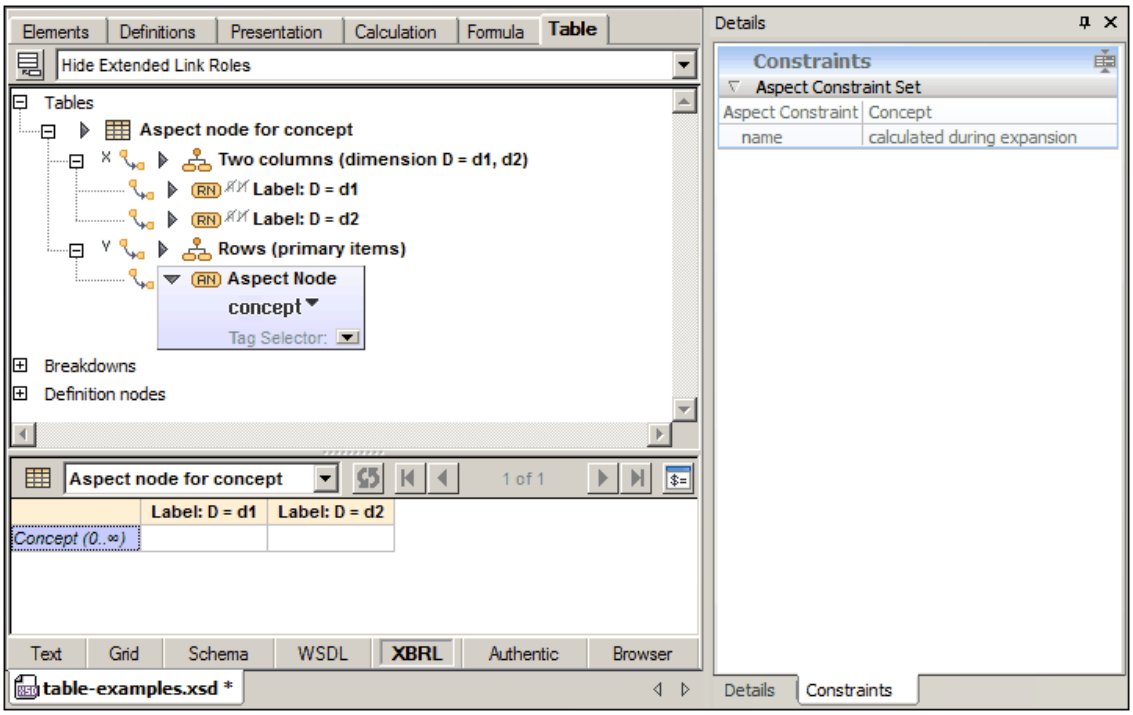

Elements
Definitions
Presentation
Calculation
Formula
Table
Hide Extended Link Roles
Tables
106 - Statement - Nanonull & Consolidated Statements of Cash Flows
103 - Statement - Nanonull & Consolidated Statements of Income
091 - Disclosure - Segment Revenue and Operating Income
104 - Statement - Nanonull & Consolidated Balance Sheets
Breakdown
Breakdown
Concept Relationship Node
Arcrole uri: http://www.xbrl.org/2003/arcrole/parent-child
Arcname none
Linkrole uri: http://www.nanonull.com/taxonomy/role/StatementOfFinancialPositionClassified
Linkname none
Source qname: us-gaap:StatementLineItems
Axis value: descendant
Generations value: 0
Parent-Child-Order: parent-first
Tag Selector:
105 - Statement - Nanonull & Consolidated Balance Sheets (Parenthetical)
Breakdowns
104 - Statement - Nanonull & Consolidated Balance Sheets
1 of 1
2010-08-31
2009-11-30
ASSETS
Current Assets
Cash and cash equivalents
Trade and other receivables, net
Inventories
Prepaid expenses and other
Total current assets
Property and Equipment, Net
Goodwill
Other Intangibles
Other Assets
Assets
LIABILITIES AND SHARE
Current Liabilities
Short term borrowings
Current portion of long-term debt
Accounts Payable, Current
Accrued liabilities and other
Customer Deposits, Current
Total current liabilities
Long-term Debt, Excluding Current Maturities
Other Long-Term Liabilities and Deferred Income
Contingencies (Note 3)
Shareholders' Equity
Additional paid-in capital
Retained earnings
Accumulated other comprehensive
Treasury stock, 39 shares at 2011 an
Total shareholders' equity
Common Stock, Value, Issued
Liabilities and Stockholders' Equity
Text
Grid
Schema
WSDL
XBRL
Authentic
Browser



リレーションシップノードには、1つのアスペクト制約があります：コンセプト または 明示的なディメンション。ですから、レイアウトセルのヘッダーは、(存在する場合) コンセプトの ラベル または その修飾名です。詳細入力ヘルパーの詳細タブは、リレーションシップノードのプロパティを表示しますが、制約タブは、レイアウト内でフォーカスされたセルにより定義されたアスペクト制約 (セット) を提供します。

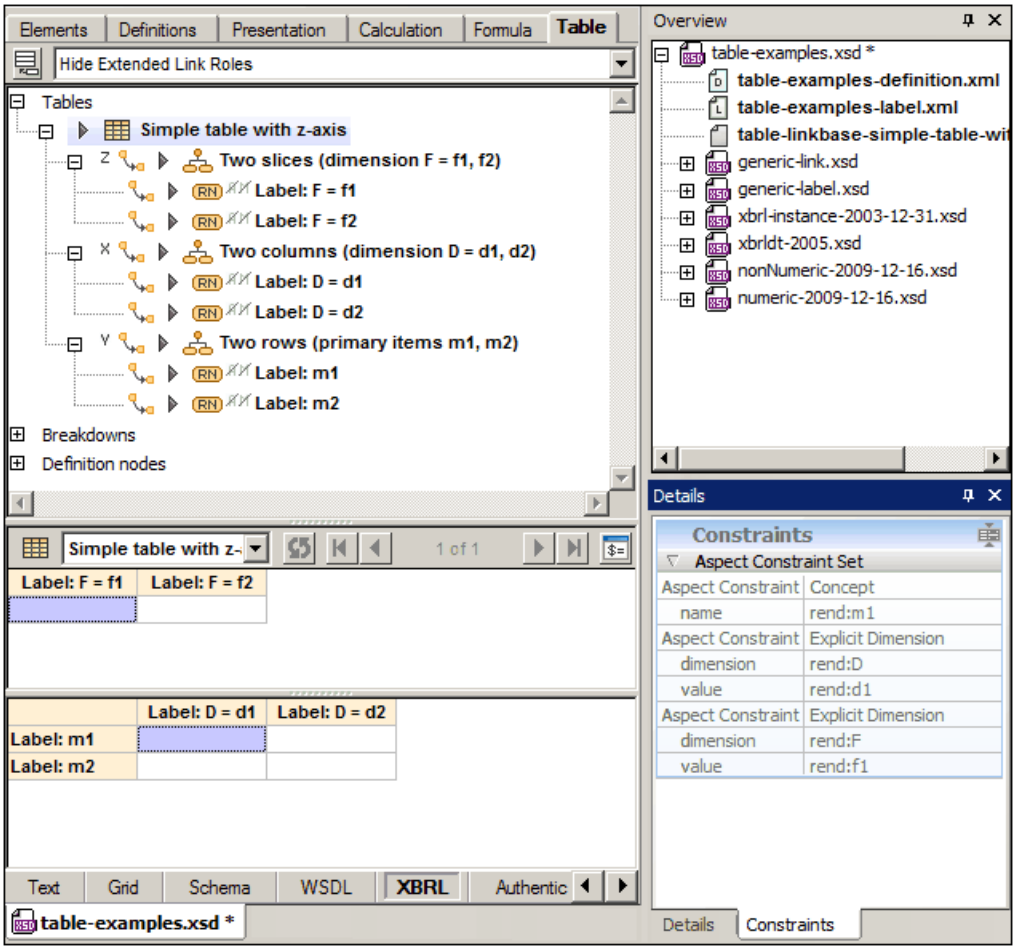
アスペクトノード

アスペクトノードは、単一の参加するアスペクトを直接指定するオープンな定義ノードです。レイアウトプロセス中、 XBRL インスタンス ファイルのファクト内に存在する、アスペクトノードは、参加するアスペクトの各明確な値のためのセルに拡張されます。アスペクト値 制約は、ノードの定義とDTS により完全に決定されないため レイアウトプレビューは、プレースホルダを表示します (*下のスクリーンショット参照*)。



Z 軸

テーブル定義がZ 軸を含む場合、この軸は、複数の 2次元のテーブルとして扱われます。テーブル レイアウトプレビュー内で、Z 軸 が個別のテーブルとしてのXY テーブル上に表示されます (*下のスクリーンショット参照*)。



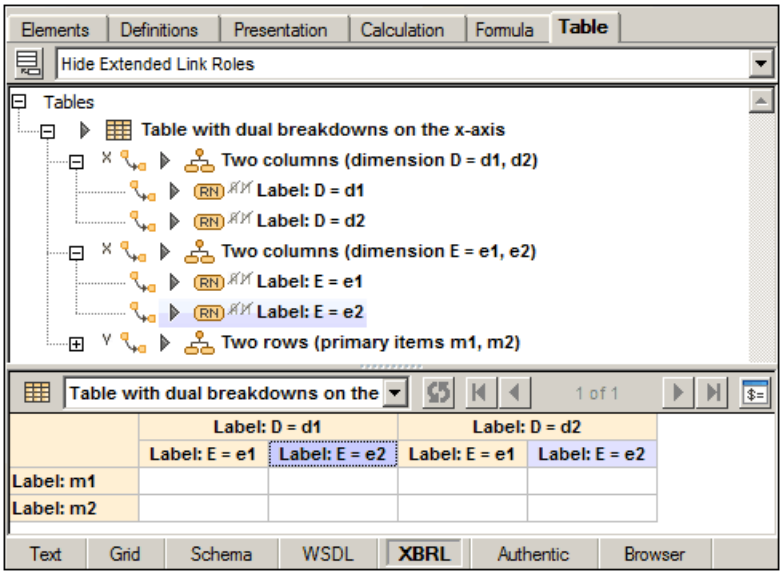
テーブル定義が、Z 軸を含むと、テーブルプレビュー内の焦点は、常に**2 つの**データセル(上のスクリーンショット参照)当てられます。 3本すべての軸のための座標は、この方法により指定されます。この方法で:XY テーブル内のX とY 座標およびZ テーブル内のZ の座標が決定されます。これは、上のスクリーンショットで表示されています。

軸に関連したプロパティおよび編集 機能は X とY 軸と同様です。 セル制約 は、存在する場合はタグセレクターを使用して)軸から計算され、詳細入力ヘルパーの制約タブ内に表示されます (*上のスクリーンショット参照*)。

複数のブレークダウンのためのプロジェクション

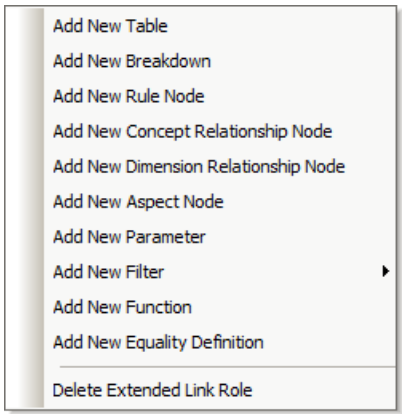
複数の独立したブレークダウンは、単一のテーブル軸に関連している可能性があります。複数のブレークダウンを単一の効果的な ブレークダウンに組み合わせるためのメカニズムは、プロジェクションと呼ばれます。 単一の軸のための複数のブレークダウンの相対的な優先順序は、各ブレークダウンの `@order` 属性 により決定されます。ブレークダウンは、ツリーとしてビジュアル化されます。最初の ブレークダウンの各リーフのため、 2番目の ブレークダウン全体が添付され、再帰的に行われます。

下のスクリーンショットでは、例えば, X 軸のために 2つの ブレークダウンがあります :**`dimension D`** は、**`dimension E`** より高い優先順位でオーダーされます。ですから、 **`dimension D`** (**`d1`** および**`d2`**) の各リーフには、**`dimension E`** のツリー全体が添付されます。X 軸 は、列を生成するため、これらのブレークダウンはテーブルの列構造のためのプロジェクションを作成します。下のスクリーンショットのテーブルレイアウトプレビューを参照してください。

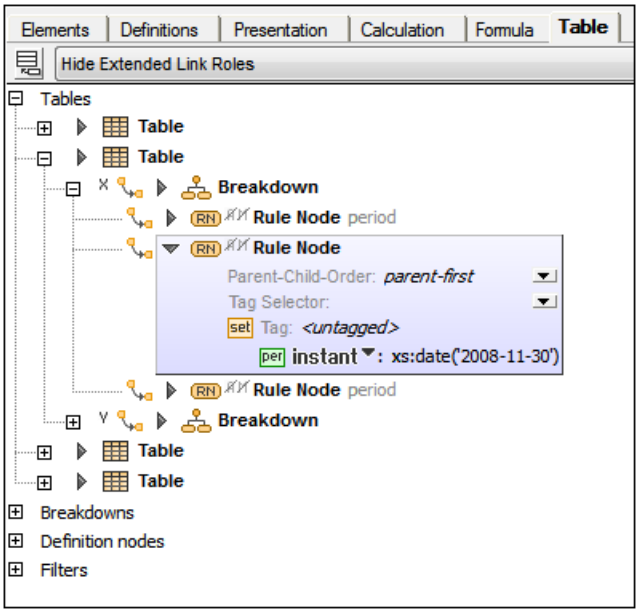


13.3.3 テーブルコンポーネント

新しいテーブルコンポーネントは、*リンクロールノード*のコンテキストメニューで作成されます (*下のスクリーンショット*)。または、*拡張リンクロールを隠す*, に設定されたビューで、をツールバーアイコン *新しいテーブルコンポーネント*介して作成されます。

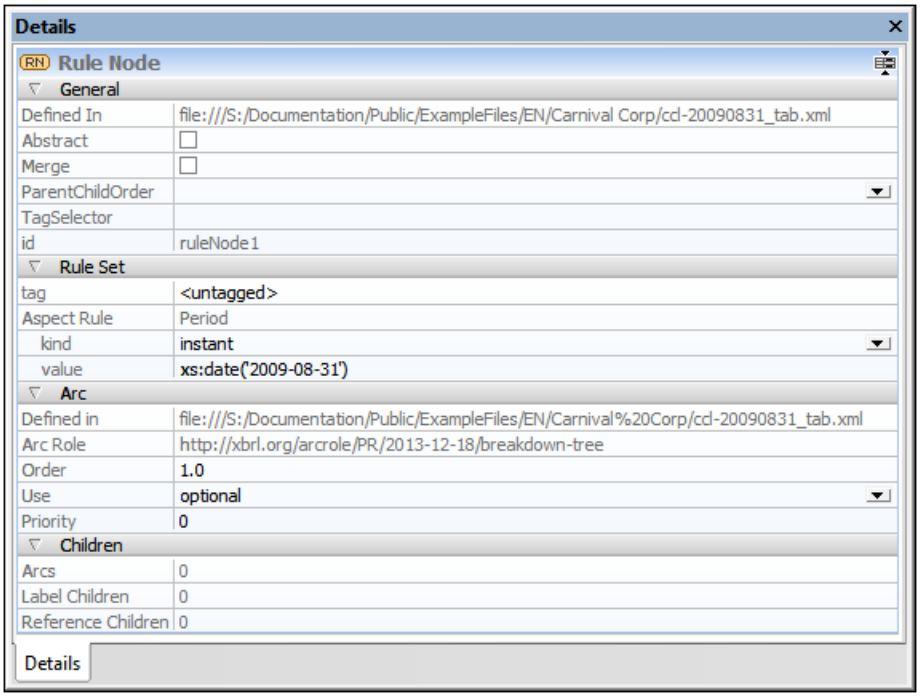


各種の コンポーネントに追加して、関係するメカニズムは、セクションのサブセクションで説明されています。テーブルコンポーネントが追加された後、テーブルタブ内のダイアグラムに表示されます (*下のスクリーンショット参照*)。



意味を明確にするために、テーブルコンポーネントは、ツリー構造内に表示されているリレーションシップto 他の コンポーネント (アーク)に分けられています (*上のスクリーンショット参照*)。コンポーネントのプロパティとリレーションシップ (アーク)のアイコンとしてダイアグラムに、コンポーネント または アークの左横に表示されています (*上のスクリーンショット参照*)。

ルールノードの詳細入力ヘルパー は、上のスクリーンショットでハイライトされています。ノードの プロパティは、*一般* セクションの下にリストされています。プロパティのboolean の値は、チェックボックスがチェックされていると、`true` チェックが解除されていると`false` で示されています。追加セクションのは、ノードに関連する他の詳細をリストします。



詳細入力ヘルパー でアークのプロパティを確認するには、ダイアグラム内から`to` (ターゲット)コンポーネントを選択します。アークの プロパティはアークセクションにリストされます。

テーブルエディター内のコンテキストメニュー

テーブルコンポーネントのコンテキストメニューは、コンポーネントの型により異なります。 メニュー アイテムは、以下のセクションに整理されています :

- リレーションの修正 (サブ-アイテムのみのため): *アークのオーバーライド*, *削除*
- コンテンツの変更 (ルールノード、リレーションシップノード):例えば、*追加*, *追加 アスペクトルール*
- *ラベル*, *参照の追加*
- 新しい子コンポーネントの作成 (リレーションシップを含む):例えば、*新しいブレークダウンの追加*
- (リレーションシップを含む)コンポーネントの削除
- (コンポーネントの)*次の*, *前の*, *発生を検索*

メモ: コンテキストメニューのより作成または削除され、*たルール セット*などのコンテンツアイテムは、追加セクション内の詳細入力ヘルパーに表示されています。

テーブル

テーブルは、プロパティ`parent-child-order` (parent-first/children-first) を提供します。オーバーライドされないテーブル内の すべての クローズされた定義ノードにより与えられたロールアップのデフォルトの位置を定義します。

ブレークダウン

ブレークダウンは、プロパティ`parent-child-order` (parent-first/children-first) を提供します。ブレークダウン内のすべての閉じられた定義ノードにより与えられた、ロールアップノードのデフォルトの位置を定義し、テーブルからの継承された値をオーバーライドします。ます。

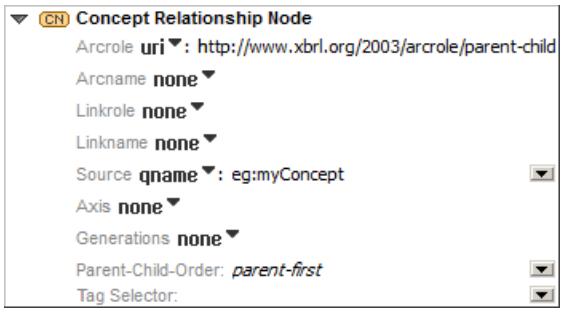
定義ノード:ルール

ルールノードは、アスペクト ルールを介して、制約を定義する閉じられた定義ノードです (フォーミュラコンポーネントを参照)。ルールノードは、ゼロまたはそれ以上のルールセットを定義します。すなわち、アスペクト ルールのセットです。各ルールセットは、タグを指定することができます。 ルールセットの1つは、タグを省略することができます。このタグ付けされていないルールセットは、常にタグ付けされたルールセットの前に表示されます。 空のタグ付けされていないルールセットは、タグ付けされたルールセットが存在する場合、表示されません。ルールノードは、以下の 2 つのBoolean プロパティをアイコンとして提供します : `abstract` および`merge`。下のスクリーンショットは、アスペクトルールなしのルールノードを表示します。

定義ノード:コンセプトリレーションシップ

コンセプトリレーションシップ ノードは、コンセプト XBRL 2.1 ネットワークのツリー ウォークを実行し、コンセプトを検索します。ツリー ウォークは、 ネットワークおよび1つまたは複数 リレーションシップソースにより一意に識別されます。コンセプトリレ

ーションシップ ノードは、単一のネットワークを識別する必要があります。多くの場合、リンクロールおよびアークロールの組み合わせで、厳密にネットワークを識別することができますが、アーク名 または 拡張されたリンクの名前などの追加の情報を指定する必要があるかもしれません。

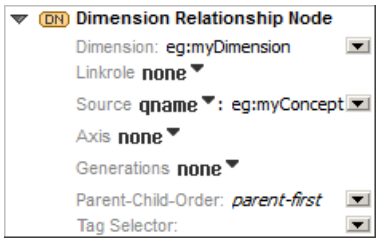



```
Arcrole:     Kind = uri | exp
Arcname:     Kind = none | qname | exp
Linkrole:    Kind = none | uri | exp
Linkname:    Kind = none | qname | exp
Source:      Kind = qname | exp
Axis:        Kind = none | value | exp
Generations: Kind = none | value | exp
```

コンセプトリレーションシップは、サブツリーを持つことができません。

定義ノード:ディメンションリレーションシップ

ディメンショナルリレーションシップノードは、明示的な ディメンション メンバーのツリーをディメンショナルなリレーションシップセット (DRS) のツリーウォークをとして説明します。この ツリー ウォークは、1つまたは複数のリレーションシップソースにより一意に識別されます。



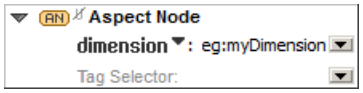

```
Linkrole:    Kind = none | uri | exp
Source:      Kind = qname | exp
Axis:        Kind = none | value | exp
Generations: Kind = none | value | exp
```

ディメンショナルリレーションシップノードは、サブツリーをもつことができません。

定義ノード:アスペクト

アスペクトノードは、1つのアスペクトを定義します。

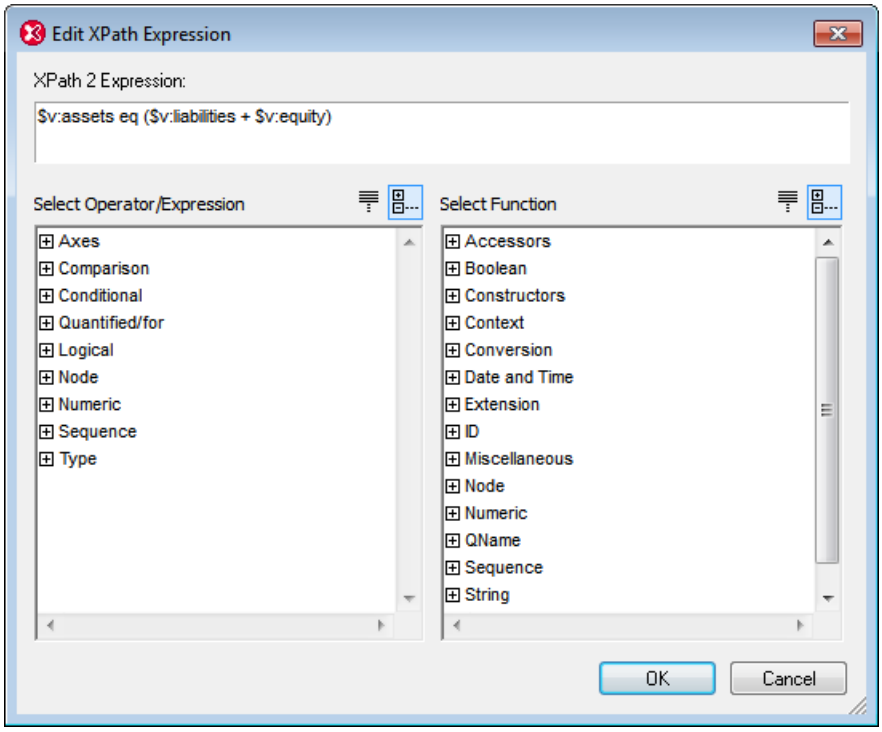
ディメンショナルなアスペクト仕様は、追加のBoolean プロパティ`include unreported value` をアイコンとしてとして与えます。



13.3.4 コンポーネントプロパティとコンテンツの編集

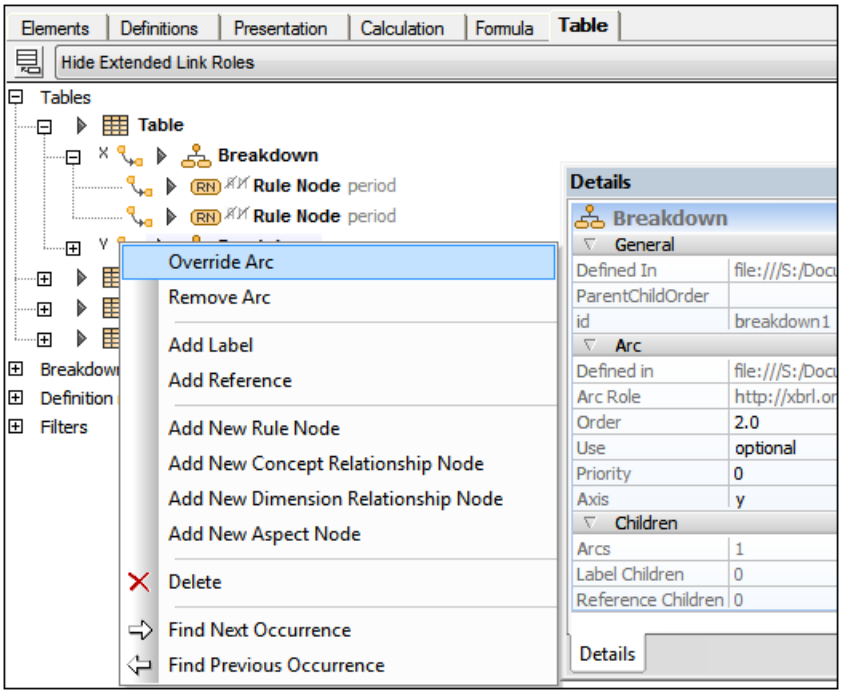
テーブルコンポーネントのプロパティは、直接ダイアグラムで、または詳細入力ヘルパーで編集することができます。

ダイアグラムで、コンポーネントが折りたたまれていると、(存在する場合は) 名前 または、適切な デフォルトのプロパティの値が、コンポーネントの説明テキストの横に、灰色で表示されます。コンポーネントをダブルクリックすると、展開することができます。プロパティをダブルクリックすると、プロパティを編集モードにします。プロパティまたは コンテンツがXPath 式を含む場合、XPath 式の編集がポップアップします。



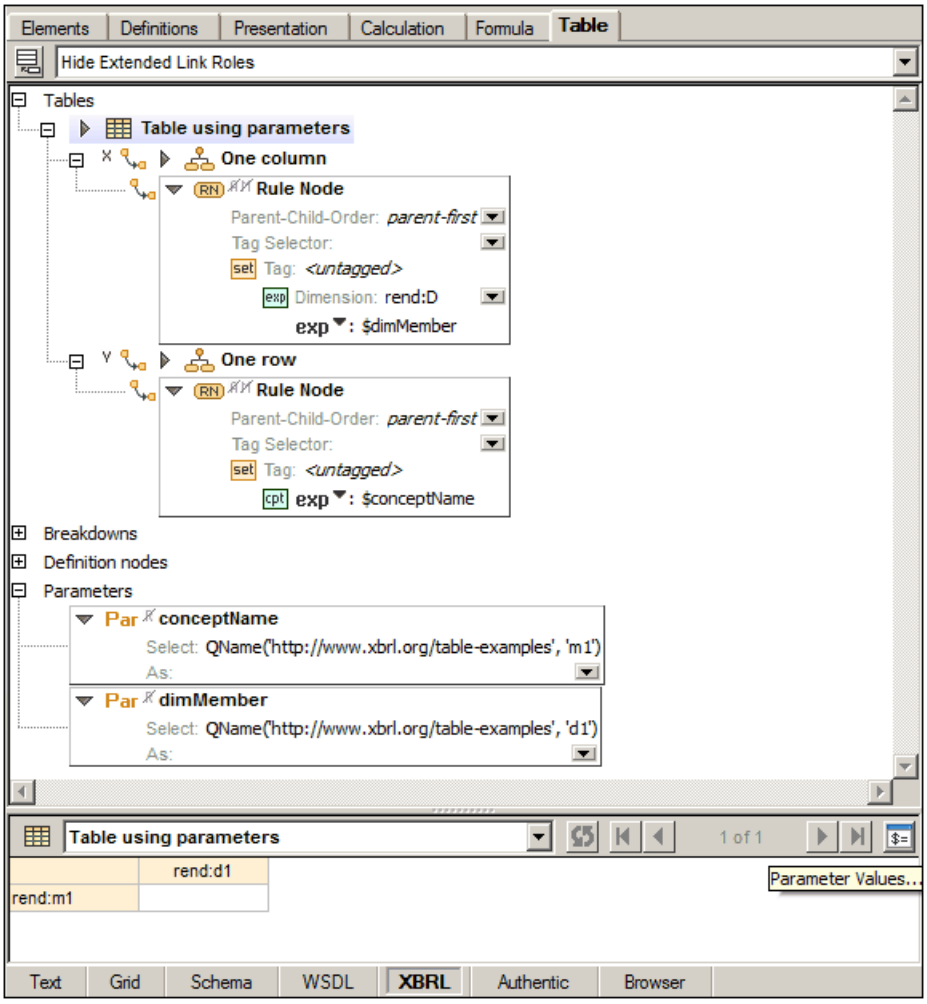
13.3.5 テーブルコンポーネントリレーションシップ

テーブルコンポーネント間のリレーションシップは、テーブルコンポーネントと他のテーブルコンポーネントにドラッグアンドドロップすることにより作成することができます。親 コンポーネントの子 の順序は、アーク-プロパティ`order` の値によります。この順序 は、子をドラッグアンドドロップして移動することにより変更することができます。(プロパティを含む)子コンポーネント リレーションをコピーまたは移動するために、コンポーネントをの上または下にドラッグします。



13.3.6 テーブルパラメーター

テーブルパラメーターは、テーブルの軸を定義するために使用することができます。例えば、下のスクリーンショット内で、選択されたテーブルのX-軸 は、パラメーター `$dimMember` により定義されます。Y-軸 は、パラメーター `$conceptName` により定義されます。 2つのパラメーターの定義は、テーブル定義の下の(グローバル)パラメーターリスト内に表示されます。テーブルレイアウトプレビュー の下のペインは、生成されるテーブルを表示します。軸は、テーブルの行と列として作成されます。

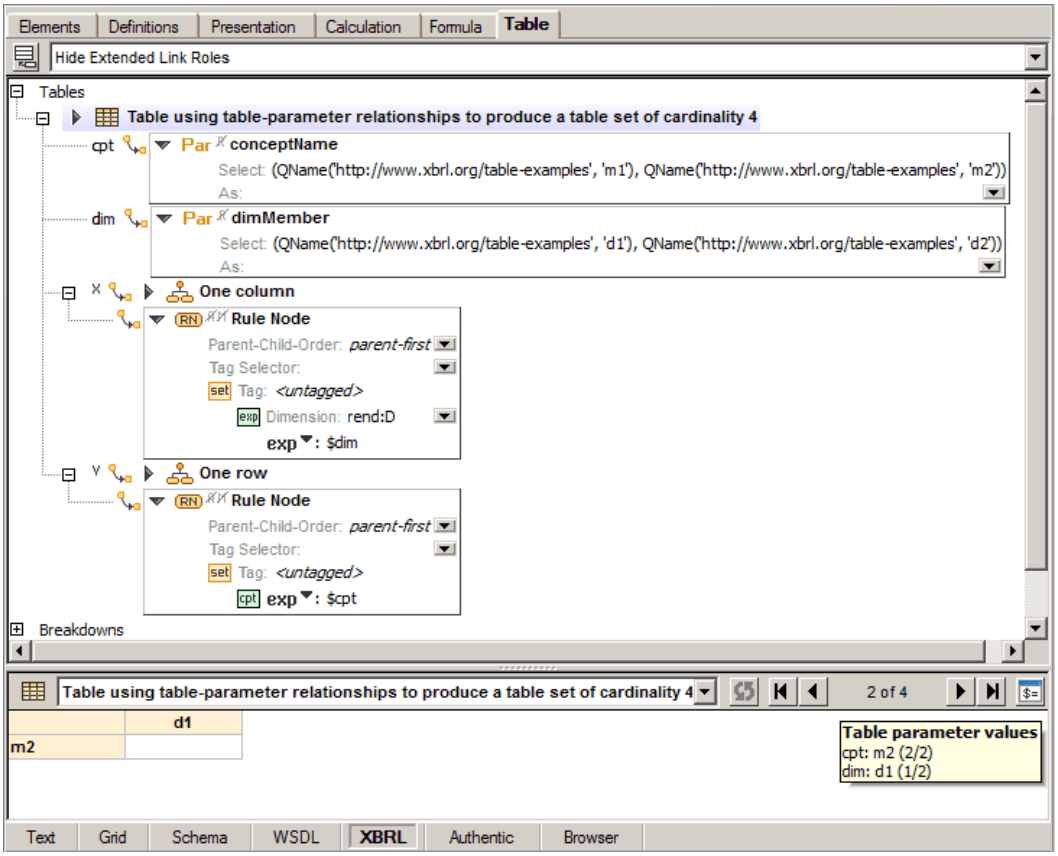


テーブルパラメーターにより単一のテーブル定義から、テーブルセットを作成する複数の関連するテーブルを作成することが許可されます。

- 単一のパラメーターが値のシーケンスを評価する場合、テーブルセットは、結果シーケンス内の各アイテムのためのテーブルを1つ含みます。
- 複数のパラメーターが値のシーケンスを評価する場合、テーブルセットは、評価パラメーターにより取得され、順序付けされた、シーケンスのデカルト製品に相対します。順序付けされたデカルト製品は次の例で表示されています：

  ```
  A × B = {1,2} × {3,4} = {(1,3), (1,4), (2,3), (2,4)}
  B × A = {3,4} × {1,2} = {(3,1), (3,2), (4,1), (4,2)}
  ```

  2 つのテーブルパラメーター（`conceptName` および `dimMember`）それぞれが 2 つの QName のシーケンスを検証します（*Select プロパティの XPath 式のを参照してください。*）

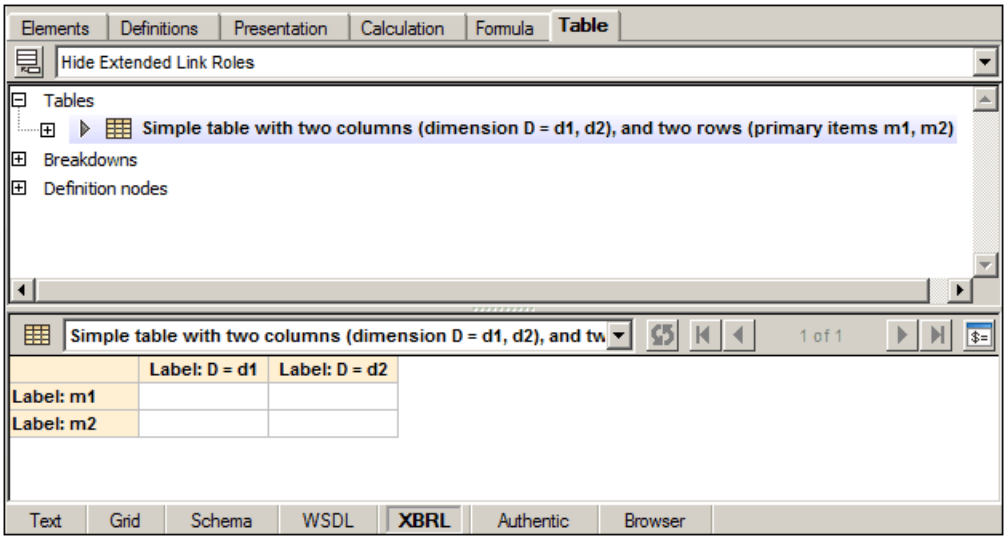


以下の点に注意してください:

- パラメーターは、ローカル パラメーターです。テーブルコンポーネントを右クリックして、 μ を選択することにより、この固有のテーブルが作成されます。これらは、最初のスクリーンショットで表示されるグローバル パラメーターではありません。
- 2 つのQNames の 2つの シーケンスの順序付けされたデカルトの製品は、4つの テーブルを作成します: `dimMember × conceptName = {d1,d2} × {m1,m2} = {(d1,m1), (d1,m2), (d2,m1), (d2,m2)}`
- テーブル セットを説明するテーブル定義が、ダイアグラム内で選択された場合、テーブル レイアウトプレビュー内のナビゲーション アイコンが有効化され、テーブルセット内のプレビューを確認することができます。現在のプレビューされているテーブルは、ツールバー内のオーダーされたテーブルセット内のインデックスにより指定されます。上のスクリーンショットでは、現在のテーブsは、 4 の 2 です。現在プレビューされているテーブルの パラメーターの値は、ポップアップで表示されます (*スクリーンショットを参照*)。
- テプレビューが現在の定義と同期されていない場合、例えば新しいコンセプトが追加された後、テーブル レイアウトプレビューの更新 ツールバー アイコンが有効化されます。
- テーブル レイアウトプレビューの パラメーターの値 ツールバー ボタンは、XBRL パラメーターの値 ダイアログを開きます。すべての テーブル パラメーターの値とデータ型 (グローバル および ローカル) を編集することができます。

13.3.7 テーブル レイアウトプレビュー

XBRL テーブル レイアウトプレビュー ペインは、テーブル定義 ツリー の下のテーブル タブ内にあります (*下のスクリーンショット参照*)。レイアウトプレビュー ペイン内のコンボボックスは、アクティブなタクソノミのテーブル リンクベース内のすべてのテーブルをリストします。テーブル のレイアウトをプレビューするには、プレビュー ペインの コンボボックスのテーブルを選択します (*下のスクリーンショット参照*)。プレビューはレイアウトのみをプレビューする点に注意してください。テーブル セルは、作成されません。これは、XBRL タクソノミ内にデータが存在しないからです。



テーブルレイアウトプレビュー により以下を行うことができます :

- テーブル レイアウトを、テーブル定義が変更されると自動的にアップデートするプレビューを使用してビジュアル化します。
- テーブル セルをクリックすることによりコンポーネントの定義に直接移動、または、テーブル定義ツリー内のコンポーネントをクリックしてテーブルセルに移動します。
- (パラメーターの値 ツールバーアイコンを介して)XBRL テーブルパラメーター ダイアログにアクセスして、テーブルパラメーターを管理します。

編集

テーブル定義とタクソノミの変更は以下のとおり処理されます :

- *テーブルの変更 :*テーブル定義の構造が変更されると、(テーブルタブ内のダイアグラム または詳細入力ヘルパーを介して、) テーブルの レイアウトプレビューはすぐにアップデートされます。パラメーター定義 または パラメーターの値の変更もこのアップデートをトリガーします。
- *DTS の変更 :*テーブル レイアウトプレビューは、Altova の XPath エンジンを定義ノード内の XPath 式 を評価するために使用します。XPath モデルは、XBRL ビューにタクソノミスキーマをロードする際に作成され、検証中にアップデートされます。基になるDTS が変更されると、(例えば、 コンセプト または リンクベースの編集による)テーブル プレビューは、変更されたDTS とはもう同期ではありません。プレビューのツールバー内のテーブルが同期されていないことを示すアイコンがこの状態を示し、ツールヒントがヒントを与えます : *プレビューは、更新 ボタンを使用して、手動で更新される必要があります。* **更新」**コマンドは、DTS の再検出を引き起こします。ですから、これは、タクソノミの検証と同等です。

	テーブル がDTS と同期されていません。
	テーブルの更新

エラーの扱い

テーブル レイアウトプレビューに関連するエラーは、以下のように扱われます：

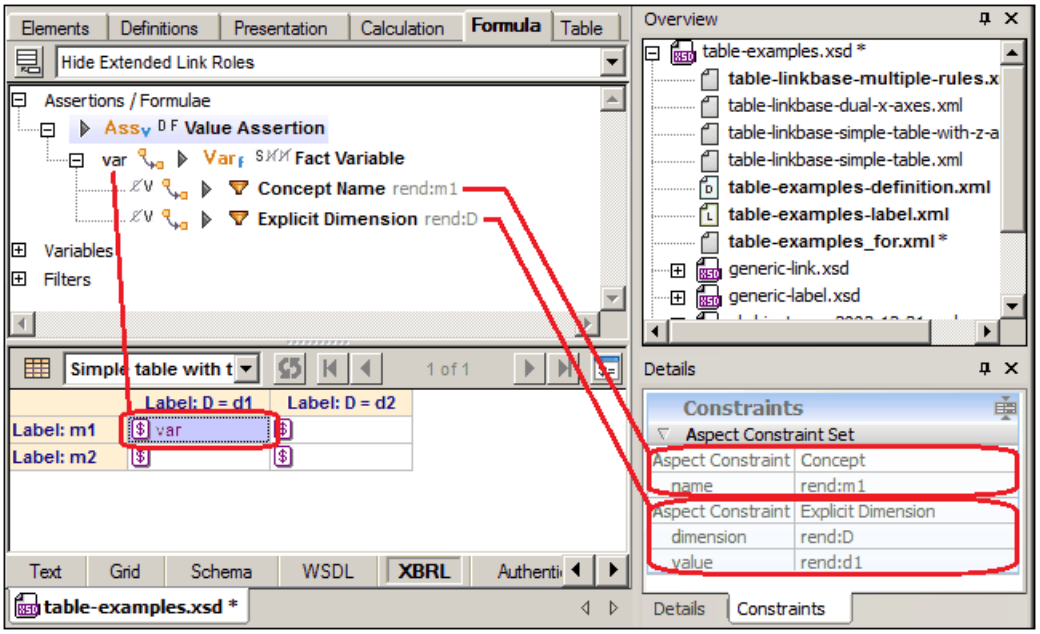
- *テーブル定義ノード内の無効な式*：テーブル定義ノードが解決することの出来ないXPath 式を含む場合、対応するレイアウト セルのヘッダーは、赤で表示されます。この場合は、無効なアスペクト制約は、詳細入力ヘルパーの制約タブ内でハイライトされて表示されます。
- *解決することのできないリレーションシップノード*：無効なプロパティまたは無効なDTS のために、リレーションシップノードが解決されない場合、ハイライトされたエラーテキストと共にプレースホルダセルを表示します。
- *子 ノード無しのマージされたルールノード*：マージされたルールノードが子 ノードを持たない場合、レイアウトは、エラーテキストと共にプレースホルダセルを表示します。
- *無効なDTS*: タクソノミが無効な場合、 XBRL タクソノミエディターにタクソノミスキーマをロード、または検証の後は、XPath モデルを使用することができません。テーブルレイアウトプレビューは、テーブルがDTS と同期されていませんのツールバーアイコンを表示し、エラー状態になります。. これに反して、レイアウトはまだ作成されることができる可能性があります。XPath 式は、しかしながら評価されることはできません。ツールバーアイコンのツールヒントはこの問題の解決方法をアドバイスします（検証 エラーおよび再検証 タクソノミをフィックスすること）。

テーブル レイアウトプレビュー内でのフォーミュラの作成

テーブル レイアウトプレビューにもフォーミュラ タブは、表示されます（*下のスクリーンショット参照*）。変数セットの下（すなわち、フォーミュラ または値 存在 アサーション下）のファクト変数の作成をサポートします。この場合は、フォーミュラ リンクベース内で対応するテーブル定義ノードが非表示のため、軸内のセルを選択することはできません。この一方、フォーミュラツリー内の変数セットが選択されるとすぐに有効化される**[ファクト変数の追加]** アイコンを表示します（*下のスクリーンショット参照*）.

選択された フォーミュラ または アサーションへのファクト変数の追加

変数セットにファクト変数を追加するには、フォーミュラ タブ内の変数セットを選択します。（変数セットは、フォーミュラ または値 存在 アサーションです。下のスクリーンショット内では、選択された変数セットは、値アサーションです。）テーブルレイアウトプレビューのセル内で、**[ファクト変数の追加]** アイコンが有効化されます。変数セットに変数を追加するためにアイコンをクリックします。実行中、データセルにより定義される、各アスペクト制約のための適切な フィルター を含む新しいファクト変数は、選択された変数セットの下に作成されます（すなわち、フォーミュラ または アサーション）。



選択された フォーミュラ または アサーションが、プレビュー内のデータセルのアスペクト制約に一致するフィルターを含む変数を

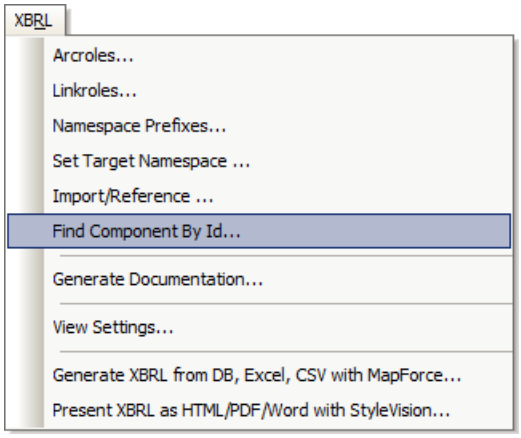
持つ場合、変数名 がデータセル内に表示されます。これは、 **[ファクト変数の追加]** アイコンを介して、新しいファクト変数を作成した後の特徴です (*上のスクリーンショット参照*).

13.3.8 テーブルコンポーネントの検索

テーブルコンポーネントは、以下の方法で検索することができます (i) ID を使用、および (ii) ドキュメント内のコンポーネントの発生のナビゲート。

ID によりテーブルコンポーネントを検索する

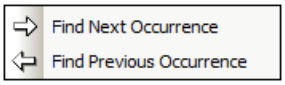
同じ型の複数のコンポーネントを含む、フォーミュラまたはテーブル リンクベースを持つ、タクソノミ内では、 (例 アサーション、フィルター、テーブル) コンポーネントをID により検索することが役にたちます。メニューコマンド **[XBRL | ID によりテーブルコンポーネントを検索する]** によりID による検索が有効化されます。



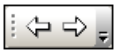
コマンドをクリックすると、検索するID を問うダイアログがポップアップします。

コンポーネントの発生を検索する

フォーミュラ コンポーネントのほとんどは、フォーミュラ リンクベースダイアグラム内に複数回表示されています : (i) 適切な セクション ノードの下に直接存在する定義、(ii) コンポーネントへのすべての参照 (*リレーションシップを介して*)。コンポーネントの コンテキスト メニュー内のコマンド **[次の発生を検索]** および **[前の発生を検索]** は、フォーミュラコンポーネントが参照されているすべての箇所にナビゲートします (*下のスクリーンショット*)。



これらのコマンドには、ツールバーアイコンを介してもアクセスすることができます (*下のスクリーンショット*)。



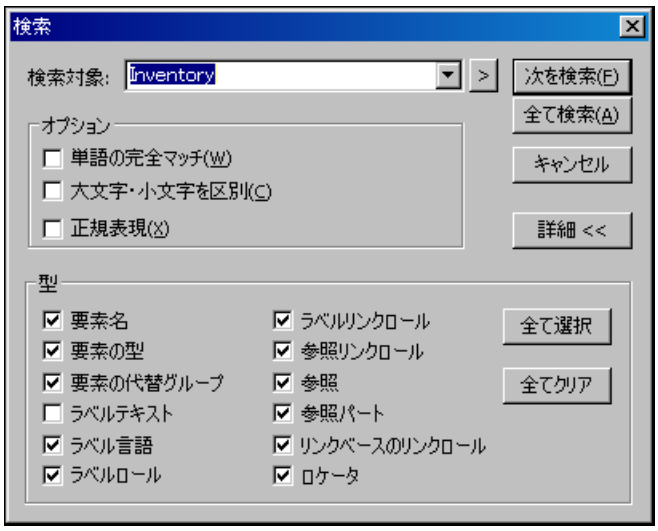
コンポーネントの定義に到達すると、効果に関するメッセージが表示されます。

13.4 XBRL 内を検索

XBRL ビューにおいてタクソノミがアクティブになっている場合、XMLSpy の XBRL 内を検索する機能を使って、タクソノミの検索を行うことができます。XBRL 内を検索する機能へアクセスする方法は、以下のようにいくつかあります：

- XBRL ビューにて XBRL タクソノミがアクティブになっているときに、メニューコマンドから **編集 | 検索** を選択。
- XBRL ウィンドウ内の **検索** ボタンを押下。
- **Ctrl** + **F** を押下

上のどの方法を使用しても、以下のスクリーンショットにある検索ダイアログが表示されます。

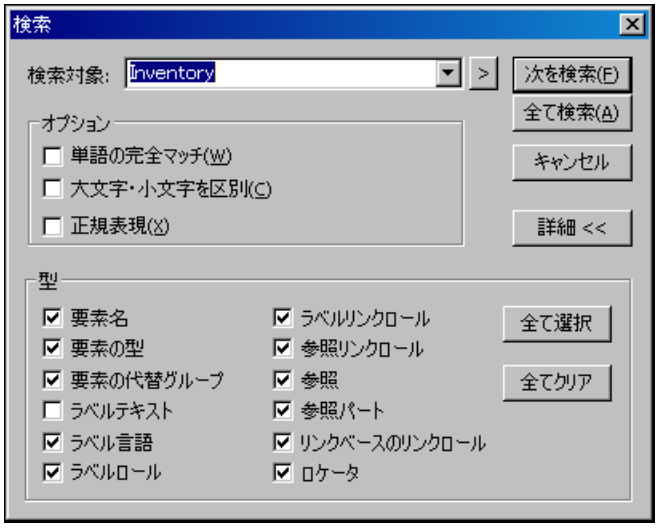


使用方法は以下のとおりです：

- 検索対象のテキストフィールドに検索語の入力を行ない、必要なオプションにチェックを入れます。
- 型ペインにて検索を行うXBRL コンポーネントを指定します。
- **次を検索** や **全て検索** ボタンを使ってコマンドを実行します。
- スキーマの中を検索ウィンドウを使用して検索の結果を確認したり、コンポーネントの素早いナビゲーションを行ないます。

13.4.1 検索語

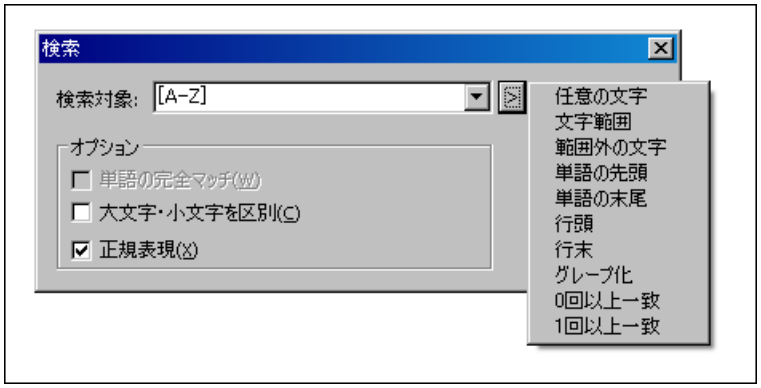
オプションペインにある対応するオプションにチェックを入れることで、大文字・小文字を区別した検索や、単語に完全にマッチする検索を行うことができます。正規表現を使った検索を行うには、**次を検索** や **全て検索** ボタンをクリックする前に、オプションペインにある*正規表現* オプションにチェックを入れてください。正規表現の使用に関する詳細は、以下を参照ください。



メモ アルファベットではない (もしくは未知の) 文字は、全て区切り文字として検索は行われます。例えば、`asset` という単語で検索が行われたとき、`xbrl:asset` という結果が返されますが、これはコロン文字 (:) が区切り文字として見なされるためです。

型ペインにて、検索対象となるコンポーネントを指定することもできます。

正規表現
正規表現を使用することで、検索の条件を更に細かく設定することができます。ポップアップリストを使用することで、正規表現の構築を楽に行うことができます。リストにアクセスするには、検索語の入力フィールドの右にある> ボタンをクリックしてください。



リストから必要なアイテムをクリックすることで、対応する正規表現の文字が入力フィールドへ挿入されます。以下は正規表現の構文文字 (メタ文字) です。

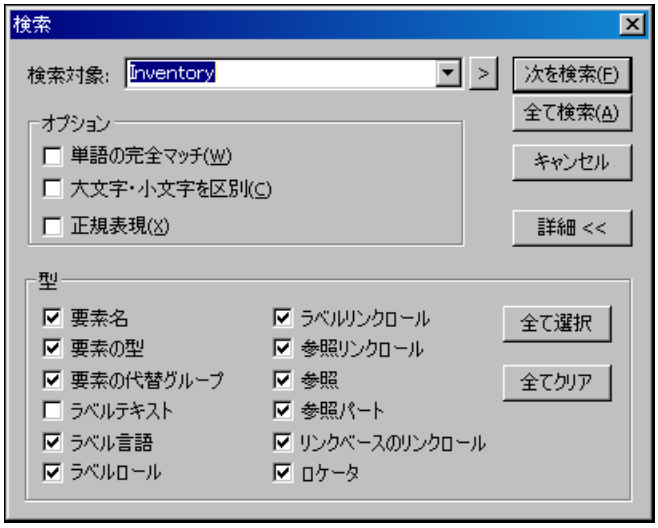
.	任意の1文字にマッチします。
\(	タグ付けされた領域の開始を表します。
\)	タグ付けされた領域の終わりを表します。
\<	単語の開始にマッチします。
\>	単語の終わりにマッチします。
\x	通常特別な意味を持つ文字 x の使用を許可します。例えば、\[は [として

解釈され、文字セットの開始とは解釈されません。

`[...]`	文字の集合を表します。例えば、`[abc]` により、a、b、c 全てにマッチすることになります。例えば、`[a-z]` などとすることで、アルファベットの小文字全てを表す範囲を表すこともできます。
`[^...]`	文字の集合の否定を表します。例えば、`[^A-Za-z]` は、アルファベット以外の文字を表します。
`^`	行の開始を表します。
`$`	行の終わりを表します。例えば`A+$` により、行の最後にある1つ以上の`A` がマッチします。
`*`	0個以上の文字の繰り返しにマッチします。例えば、`Sa*m` により`Sm`、`Sam`、`Saam`、`Saaam..` といった文字にマッチします。
`+`	1個以上の文字の繰り返しにマッチします。例えば、`Sa+m` により`Sam`、`Saam`、`Saaam...` といった文字にマッチします。

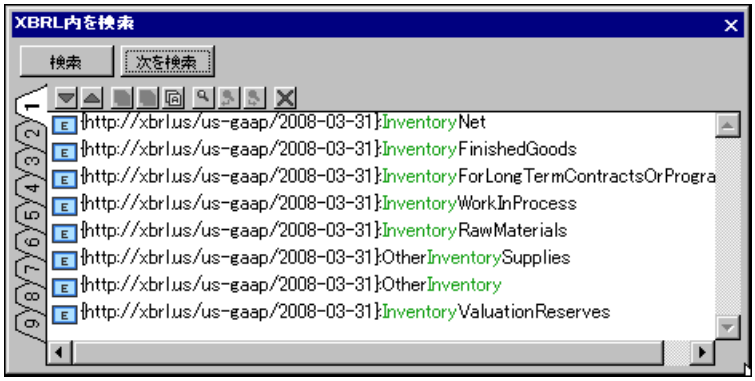
13.4.2 コマンドの実行

検索語が入力され、検索を行うコンポーネントの型に関する検索オプションがセットされた状態で、使用することのできるコマンドには[**次を検索**] と[**すべてを検索**] の 2種類あります。これらコマンドは検索ダイアログのボタンにより使用することができます (以下のスクリーンショットを参照)。



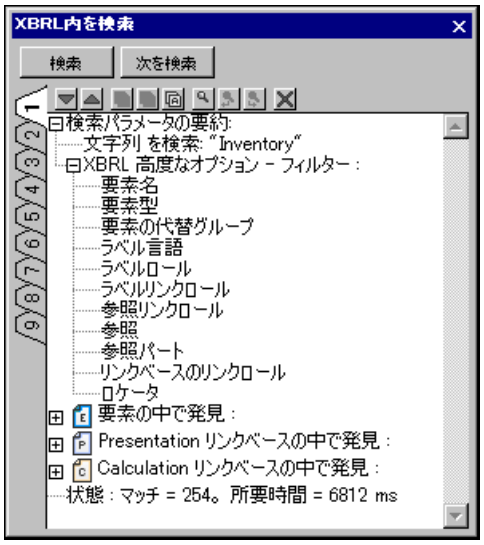
次を検索

次を検索コマンドにより、スクリーンショットにあるように検索語にマッチした次の検索結果がXBRL 内を検索ウィンドウに表示されます。次のインスタンスへの検索は、現在アクティブなドキュメントにてカーソルがある位置の次のセルから開始されます。 **「次を検索」**処理は、全てのドキュメントにあるインスタンスが表示されるまで続けられます。



全て検索

全て検索コマンドにより、全ての検索結果が、その概要と共にXBRL 内を検索ウィンドウ内に表示されます（以下のスクリーンショットを参照）。

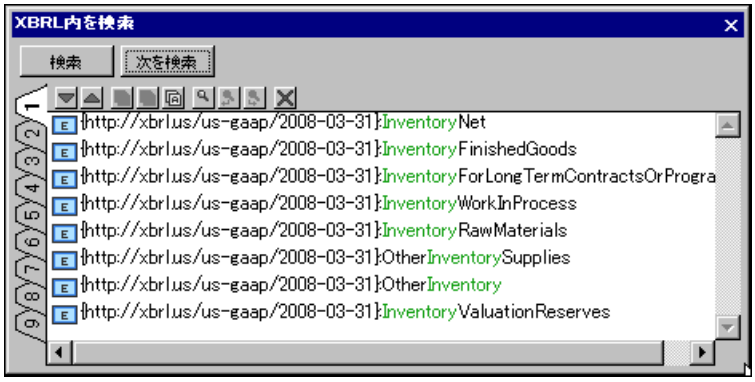


検索結果には、(i) 検索されたXBRL コンポーネント型の概要、(ii) リンクベースごとに表示される検索語に対してマッチしたインスタンスのリスト、そして (iii) マッチしたインスタンスの数と検索に要した時間が表示されます。各リンクベースグループは展開や縮退を行うことができ、そのグループ内のマッチを確認することができます。マッチをクリックすることで、XBRL ドキュメントの対応する要素がメインウィンドウにてハイライトされます。

XBRL 内を検索ウィンドウに関するより詳しい情報は、結果と情報のセクションを参照ください。

13.4.3 結果と情報

「検索」や **「次を検索」**コマンドが実行されるたびに、その実行結果が（以下のスクリーンショットにある）XBRL 内を検索ウィンドウに表示されます。検索された単語は緑色で表示されます。以下のスクリーンショットでは、`Inventory` という単語が検索された結果が示されます。



XBRL 内を検索ウィンドウの機能

検索の結果は（1から9まで番号が与えられている）9つあるタブに個別に表示させることができます。これらのタブを使うことで、各検索結果を個別のタブに保持しておくことができます。XBRL 内を検索ウィンドウの検索結果をクリックすることで、対応するコンポーネントがXBRL ビューのメインウィンドウに表示、ハイライトされます。これらXBRL 内を検索ウィンドウの機能により、目的のコンポーネントへの検索とナビゲーションを素早く行うことができます。

以下のツールバーコマンドを利用することができます：

- **「前」**と**「次」**アイコンにより、選択されている結果から見て、それぞれ前、そして次の検索結果を選択することができます。
- **[メッセージをコピー]**コマンドにより、選択されたメッセージ、選択されたメッセージとその子、そして全てのメッセージをクリップボードにコピーすることができます。
- **「検索」**コマンドにより、XBRL 内を検索ウィンドウ内のテキストを検索します。
- **「クリア」**コマンドにより、現在アクティブになっているタブに表示されている全てのメッセージを削除します。

14 Office Open XML、ZIP、EPUB

Office Open XML (OOXML) や ZIP ファイル、そして EPUB ファイルは、他のファイルを内部に含むパッケージという意味では、すべて似たようなものです。XMLSpy の Archive View では、これらパッケージの内部構造を閲覧するためのビューが提供され、XMLSpy 内部から構造の修正や、パッケージ内のファイルにアクセス、編集を行うことができます。EPUB ファイルの場合、XMLSpy の Archive View からブラウザービュー内にて EPUB ブックを直接閲覧することができます。

Office Open XML (OOXML)

OOXML はドキュメント、スプレッドシート、そしてプレゼンテーションの記述を行うためのファイル形式です。元々は Microsoft の Office パッケージのために開発されましたが、現在はオープンな ECMA 仕様になっています。

OOXML ファイルの構造

各 OOXML ドキュメントは、Open Package Convention に従った複数ファイルのパッケージになっています。パッケージは XML やその他のデータファイル (例えばイメージファイル) と、それらファイルの関係が記述されたリレーションシップファイルから構成されています。

OOXML ファイルの内部構造や、内部フォルダー、ファイル名の構造はドキュメントの種類により変化します。しかし、`[Content_Types].xml` という共通の XML 構造ファイルがルート部にあり、ドキュメントの種類により変化するディレクトリと、`_rels`、そして `docProps` の3つのディレクトリは常に存在します。例えば、`.docx` ドキュメントの場合、ディレクトリの名前は `word` になり、`.xlsx` ドキュメントの場合は、`xl`、そして `.pptx` の場合は `ppt` となります。

```
OOXML ファイル
|-- ファイル:          [Content_Types].xml
|-- フォルダー:   _rels
|-- フォルダー:   docProps
|-- フォルダー:   word/xl/ppt
```

- `_rels` フォルダーには `rels.xml` ファイルが含まれており、パッケージ内の様々なファイルが記述されています。
- `docProps` フォルダーにはドキュメントプロパティの記述を行う `app.xml` と `core.xml` が含まれています。
- `word`、`xl`、そして `ppt` フォルダーには、ドキュメントのコンテンツとなる XML ファイルが含まれています。例えば、`word` フォルダーには、ドキュメントのコアコンテンツとなる `document.xml` というファイルが保存されています。

XMLSpy のアーカイブビューで見る OOXML

以下のスクリーンショットにある XMLSpy のアーカイブビューを使うことで、OOXML ファイルのコンテンツの閲覧と編集を行うことができます。

画面の左側にあるフォルダービューには、パッケージ内のフォルダーが表示され、メインウィンドウにはフォルダービューで選択されているフォルダー内にあるファイルが表示されます。アーカイブビューを使えば、アーカイブからファイルやフォルダーの追加や削除を行うことができます。更に、アーカイブビューのファイルをダブルクリックして XMLSpy にて素早く編集を行うこともできます。

OOXML の内部ファイルに対するインテリジェントな編集

OOXML パッケージ内部にあるXML ドキュメントは、標準的なスキーマをベースにしています。XMLSpy はOOXML ドキュメントに対してインテリジェントな編集機能をサポートしており、入力ヘルパーや自動補完、検証などの機能を使用することができます。

ZIP ファイル

ZIP ファイルは複数のファイルを可逆圧縮の形で含むアーカイブファイルです。含まれるファイルの形式には様々なものがあります。XMLSpy のアーカイブビューを使えば、ZIP ファイルの作成、内部構造の修正や、アーカイブされたファイルの編集を行うことができます。これらの機能については、このセクション以下のZIP ファイルサブセクションにて記述されます。

EPUB ファイル

EPUB ファイルとはデジタル形式の出版物（EPUB books）を配布するためのファイルの集合体のことです。Archive View ではEPUB ファイルを開いたり、EPUB ファイルの作成、デジタルEPUB ブックのプレビュー、EPUB アーカイブのコンポーネントファイルをXMLSpy にて直接編集したり、EPUB ファイルを検証、コンポーネントをEPUB アーカイブへ保存しなおすことができます。詳細についてはEPUB ファイルのセクションを参照ください。

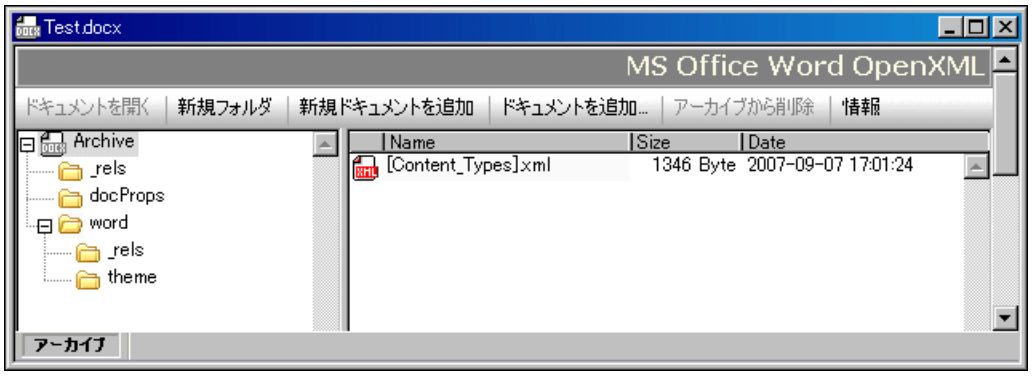
14.1 OOXML ファイルの編集

このセクションでは、Archive View を使ったOOXML ドキュメントの編集方法について説明します。以下の機能について解説します：

- OOXML ファイルを作成、開き、保存する
- OOXML ファイルの構造を編集する
- OOXML 内部のドキュメントを開き、編集、保存する
- OOXML 内部ドキュメントのインテリジェントな編集
- OOXML ファイル内部のアドレス
- OOXML アーカイブの比較

OOXML ファイルを作成、開き、保存する

OOXML ファイルは（**「ファイル | 新規作成」**コマンドから開始できる）新規ドキュメントを作成ダイアログから、目的のファイルタイプ（.docx、.pptx、.xlsx）を選択して、作成することができます。ファイル名とファイルを保存する場所を指定するよう促されます。指定された場所に新しいファイルが作成され、（以下のスクリーンショットにある）Archive View にて開かれます。OOXML ドキュメントの基本的な内部構造が既に作成されていることに注目してください。



XMLSpy のファイルを開くダイアログ（**「ファイル | 開く」**）により既存のOOXML ファイルを開くこともできます。OOXML の保存を行うには **「ファイル | 上書き保存」**コマンドを使用します。このコマンドによりOOXML ファイルの構造とリレーションシップが保存されます。

OOXML ファイルの構造を編集する

Archive View の機能を使用することでOOXML ファイルのコンテンツにフォルダーやドキュメントの追加、削除を行うことができます。これら構造上の修正を終えると、それらの変更を反映させるために **「ファイル | 上書き保存」**コマンドを使ってOOXML ファイルの保存を行う必要があります。以下の点に注意してください：

- 新規フォルダーまたは新規ドキュメントがArchive View のコマンドボタンから追加された場合、作成時（追加時）にその名前を決定しなければなりません。Archive View でフォルダーまたはドキュメントの名前を変更することはできません。
- 新規ドキュメントがアーカイブフォルダーに追加されると、そのドキュメントを保存するか、OOXML ファイルそのものを保存することで、そのドキュメントが実際にアーカイブに保存されます。

OOXML 内部のドキュメントを開き、編集、保存する

Archive View で表示されているOOXML 内部のドキュメントをダブルクリックするか、メインウィンドウで選択し、ドキュメントを開くコマンドボタンをクリックすることでドキュメントを開くことができます。ドキュメントはXMLSpy のウィンドウで別に開かれます。ドキュメントの編集後は、ドキュメントを保存することで、ドキュメントがOOXML アーカイブファイル内に保存されます。

OOXML 内部ドキュメントのインテリジェントな編集

XMLSpy のインテリジェントな編集機能をOOXML ファイルパッケージ内のOOXML ドキュメントに対して使用することができます。インテリジェントな編集機能により提供される入力ヘルパー、自動補完、そして検証機能を使用することができます。

OOXML ファイル内部のアドレス

OOXML ファイル内のドキュメントのアドレスは、通常のファイルパスとパイプ文字 (|) を組み合わせることで表現することができます。例えば、以下のファイルパスは：

```
C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017
\Examples\Office20XX\ExcelDemo.xlsx|zip\xl\テーブル\テーブル1.xml
```

XMLSpy サンプルフォルダーの`Examples/Office20XX` フォルダーにある`ExcelDemo.xlsx` OOXML ファイル内に格納されている`xl\テーブル` フォルダーの`テーブル.xml` というファイルを表します。

OOXML アーカイブの比較

OOXML ファイルがArchive View で開かれている場合、**[ツール| ディレクトリを比較]** コマンドを使用することで、他のアーカイブとの比較を行うことができます。

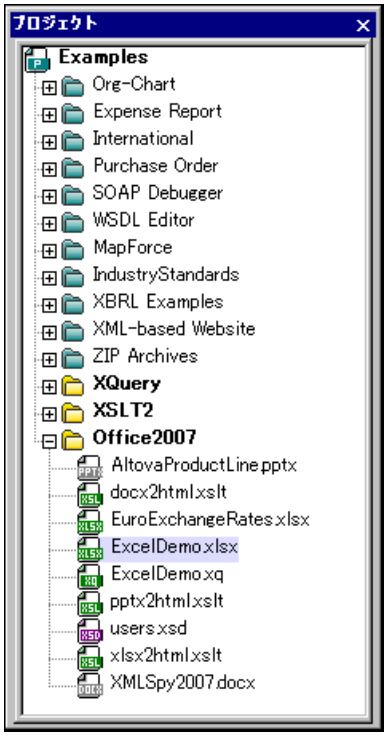
14.2 OOXML サンプルファイル

お使いのXMLSpy アプリケーションフォルダー以下にあるExamples\Office2007 フォルダーには、以下のサンプルファイルが配置されています：

- OOXML ファイル：(i) Word Open XML ファイル (.docx)、(ii) Excel Open XML ファイル (.xlsx)、(iii) PowerPoint Open XML ファイル (.pptx)
- XSLT ファイル：(i) docx2html.xslt (サンプル.docx ファイルをHTML へ変換)、(ii) xslx2html.xslt (サンプル.xslx ファイルをHTML へ変換)、(ii) pptx2html.xslt (サンプル.pptx ファイルをHTML へ変換)
- XQuery ファイル：ExcelDemo.xq (.xslx ファイルからデータを取得する)

XSLT ならびにXQuery ファイルはOOXML ファイル内部のデータにアクセス、変換する際にXSLT やXQuery がどのように使用できるかのデモを行う目的で作成されています。XSLT やXQuery ドキュメントを実行するには、以下のどれかの方法を使用します：

- Archive View にてOOXML ファイルを開きます。フォルダービューにて、Archive を選択し、メニューコマンドから**[XSL/XQuery | XSL 変換]** を選択するか、**[XSL/XQuery | XQuery 実行]** を選択します。XSLT またはXQuery ファイルを参照し、「OK」をクリックします。
- XMLSpy のプロジェクトウィンドウにて、以下のスクリーンショットにあるように Examples プロジェクトの Office2007 フォルダー以下にある.xlsx、.pptx または.docx ファイルを右クリックし、変換コマンドを選択します。変換を行うファイルを選択し、「OK」をクリックします。



- XMLSpy にてXSLT またはXQuery ファイルを開き、メニューコマンドから**[XSL/XQuery | XSL 変換] または [XSL/XQuery | XQuery 実行]** をそれぞれ選択します。変換を行うXML を選択するよう促されるので、.docx、.xlsx、または.pptx ファイルを選択します。
- これらのオペレーションはAltova XSLT ならびにXQuery エンジンを搭載したスタンドアロンパッケージのAltovaXML を使っても行うことができます。

14.3 ZIP ファイル

Archive View ではZIP ファイルを作成したり、ファイル内部の構造を修正したり、XMLSpy にてZIP パッケージに含まれるファイルを直接編集し、そのファイルをZIP アーカイブへ保存しなおすことができます。

ZIP ファイルの作成と保存

ZIP ファイルは新規ドキュメント作成ダイアログ（**「ファイル| 新規作成」**コマンド）にて .zip を選択することで作成することができます。（スクリーンショットにあるような）空のZIP アーカイブがXMLSpy で作成されます。その後 **「ファイル| 上書き保存」**コマンドを使ってZIP ファイルを適切な場所に保存する必要があります。以下に記されるようにファイルやフォルダーを追加して、これらの変更を保存するためにZIP ファイルを保存します。

既存のZIP ファイルはXMLSpy のファイルを開くダイアログ（**「ファイル| 開く」**）から開くことができます。

注意：OOXML ファイルの作成時には、アーカイブがArchive View で開かれる前に保存場所を尋ねられるのに対して、ZIP ファイルの場合、作成時に保存場所を選択するよう促されることはありません。Archive View で開かれている空のアーカイブのZIP ファイルを保存するには、明示的に **「ファイル| 上書き保存」**コマンドを選択する必要があります。

フォルダーやファイルの追加とアーカイブ構造の修正

選択されたアーカイブフォルダーに対してフォルダー（**新規フォルダー**）や、既存のドキュメント（**ドキュメントを追加**）、そして新規ファイル（**新規ドキュメントを追加**）を追加することができます。新規フォルダーや新規ドキュメントを追加したときには、直ちにフォルダーまたはファイルの名前を入力する必要があります。Archive View にてフォルダーやファイルの名前の変更を行うことはできません。

ZIP ファイル内部のアドレス

ZIP ファイル内部のドキュメントは、通常のパスとパイプ文字（|）を組み合わせることで表現することができます。例えば、以下のファイルパスは：

```
C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017
\Examples\Test.zip|zip\TestFolder\MyFile.xml
```

XMLSpy サンプルフォルダーのExamples フォルダーにあるTest.zip ZIP ファイル内に格納されているTestFolder フォルダーのMyFile.xml というファイルを表します。

ZIP アーカイブの比較

ZIP ファイルがArchive View で開かれている場合、**[ツール| ディレクトリの比較]** コマンドを使用することで、別のアーカイブとの比較を行うことができます。

14.4 EPUB ファイル

EPUB ファイルとはデジタル形式の出版物 (EPUB books) を配布するためのファイルの集合体のことです。Archive View ではEPUB ファイルを開き、EPUB ファイルの作成、デジタルEPUB ブックのプレビュー、EPUB アーカイブのコンポーネントファイルをXMLSpy にて直接編集したり、EPUB ファイルを検証、コンポーネントをEPUB アーカイブへ保存し直すことができます。詳細については EPUB ファイルのセクションを参照ください。

メモ： **(i)** XMLSpy ではEPUB 2.0.1 がサポートされ、**(ii)** サンプルEPUB ファイルは `(My) Documents/Altova/XMLSpy2017/Examples` フォルダー内にある`Examples` プロジェクトにてご利用になれます。

用語解説

このセクションでは、以下のような用語が使用されます：

- **EPUB ファイル**とは、`.epub` というファイル拡張子をともなうEPUB ファイルのことです。ファイルそのものはZIP ファイルで、Archive View にて開くことができる全てのファイルがアーカイブ内に含まれます。
- **アーカイブファイル**とは、EPUB アーカイブを含んでいるファイルのことです。
- **EPUB ブック**という用語は、アーカイブされたEPUB ファイルにより生成されたデジタル形式の出版物のことを指します。

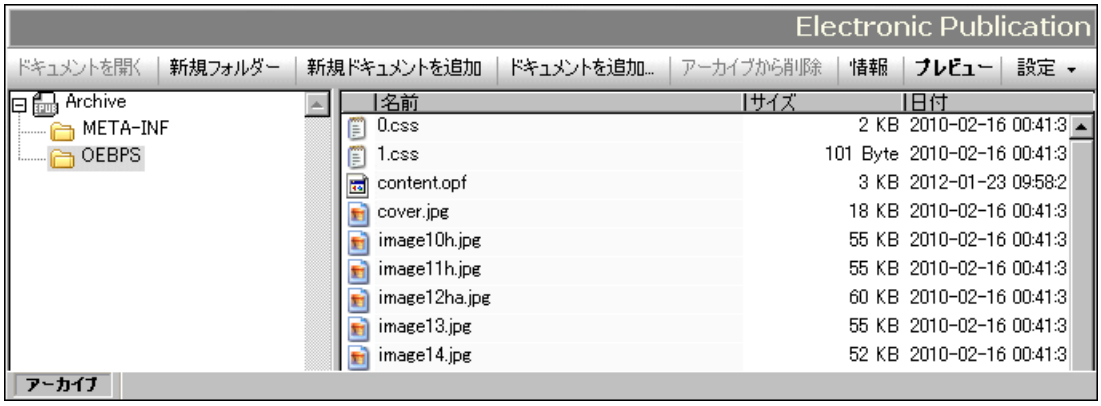
このセクションの構成

このセクションでは、XMLSpy におけるEPUB の機能が以下のように構成されます：

- Archive View にてEPUB ファイルを開く
- 新規 EPUB ファイルを作成する
- EPUB ブックのプレビュー
- EPUB アーカイブのコンテンツや構造を修正する
- 情報と設定
- XMLSpy にてアーカイブファイルを編集する
- アーカイブファイルの入力ヘルパー
- EPUB ファイルの検証

Archive View にて EPUB ファイルを開く

メニューコマンドから**ファイル | 開く**を選択し、EPUB ファイルを選択します。EPUB ファイルがArchive View にて表示されます (以下のスクリーンショットを参照)。Windows エクスプローラーにてEPUB ファイルを右クリックし、コンテキストメニューからファイルをXMLSpy にて開くこともできます。XMLSpy をEPUB ファイルのデフォルトエディターとすることで、EPUB ファイルをダブルクリックすることで、ファイルがArchive View にて開かれるようになります。



左側にはアーカイブ内にあるフォルダーが表示され、メインウィンドウには、フォルダービューにて選択されたフォルダー内にあるファイルが表示されます。EPUB アーカイブには以下のような構造とコンポーネントが表示されます。

```
Archive
|-- Mimetype ファイル
|
|-- META-INF フォルダー
|     |-- container.xml
|
|-- ドキュメントフォルダー (上のスクリーンショットでは OEBPS がドキュメントフォルダーとなります)
      |-- HTML、CSS、イメージファイル、そして OPF ならびに NCX ファイルが含まれます
```

新規 EPUB ファイルを作成する

新規にEPUB ファイルを作成するには、メニューコマンドから**[ファイル| 新規作成]** を選択します。新規ドキュメントの作成ダイアログにて、.epub というファイルタイプを選択します。名前を付けて保存ダイアログにて、EPUB ドキュメントの名前を入力し、**「保存」** ボタンをクリックします。妥当なEPUB アーカイブを構成するために必要な全てのファイルが含まれるスケルトンのEPUB アーカイブが、Archive View 内の新規ウィンドウにて作成されます。以下に記される方法により、必要に応じてファイルやフォルダーをアーカイブへ追加して、EPUB ファイルを保存します。XMLSpy にて直接アーカイブファイルを編集するには、Archive View にあるファイルをダブルクリックします。新たな XMLSpy ウィンドウにてファイルが開かれます。ファイルの編集を行い **[ファイル| 保存]** コマンドにより保存してください。

EPUB ブックのプレビュー

EPUB ブックをプレビューするには、Archive View にてEPUB ファイルがアクティブになっている状態で、Archive View のツールバーにて**プレビュー**ボタンをクリックします。EPUB ブックがXMLSpy 内にある (nternet Explorer) ブラウザーウィンドウにて開かれます。これらファイルのどれか -コンテンツファイルであれ構造に関連するファイルであれ -が変更され、まだ保存されていない場合、ファイルを保存するように促されます。変更点を保存しない場合、以前保存されたデータがプレビューには使用されます。Archive View のツールバーにある**[設定]** ボタンから、プレビューを行う前に、変更されたファイルを保存するように指定することができます。

以下の点に留意してください：

- EPUB データがプレビューウィンドウで開かれている時にアーカイブビューの**[プレビュー]** ボタンをクリックすると、EPUB データが既に開かれているプレビューウィンドウで再ロードされます。
- プレビューウィンドウ (nternet Explorer の再描画 (F5) コマンドにより) を再描画しても、プレビューウィンドウそのものは更新されません。アーカイブビューの**[プレビュー]** ボタンを使用して、プレビュー ウィンドウ内のEPUB データを更新してください。
- プレビュー ウィンドウを閉じることで、プレビューを閉じることができます。

メモ： 全てのEPUB マークアップがInternet Explorer によりサポートされているわけではなく、プレビューのレイアウトが

崩れることもあります。更に、デジタル出版物のドキュメントがHTML ではなくXML をベースにしたものである場合、プレビューが正常に行われない可能性もあります。最新バージョンのInternet Explorer ではEPUB マークアップの扱いが改善されており、問題が発生した場合は、最新バージョンのInternet Explorer へ更新することをおすすめします。

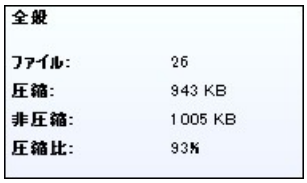
EPUB アーカイブのコンテンツや構造を修正する

選択されたアーカイブフォルダーに対して、フォルダーを追加 (**[新規フォルダー]** ボタンをクリック)、新規ファイルを追加 (**[新規ドキュメントの追加]** ボタン)、既存のファイルを追加 (**[ドキュメントの追加]** ボタン) することができます。新規フォルダーや新規ドキュメントを追加する場合、そのファイルやフォルダーの名前を直ちに入力する必要があり、Archive View にてフォルダー名を変更することはできません。ファイルやフォルダーを選択して、**[アーカイブから削除]** ボタンをクリックすることで、ファイルやフォルダーを削除することができます。

アーカイブを変更したら、それら変更を反映させるために (**[ファイル | 保存]** コマンドにて) EPUB ファイルを保存する必要があります。

情報と設定

[情報] ボタンをクリックすることで、Archive View の下部にアーカイブ情報の概要が表示されます (以下のスクリーンショットを参照)。再度 **[情報]** ボタンをクリックすることで、概要が非表示となります。概要には、アーカイブ内にある (Mimetype や `container.xml` を含む) ファイルの数や圧縮されたEPUB ファイルの大きさ、圧縮されていない状態のファイルの大きさが示されます。



[設定] ボタンをクリックすることで、ファイルの自動保存オプションに対応したドロップダウンが表示されます。このコマンドにより、() 検証前、そしてEPUB ファイルを (XMLSpy の **[プレビュー]** ボタンにより) プレビューする前にファイルを自動保存するように指定することができます。

XMLSpy にてアーカイブファイルを編集する

アーカイブファイルをXMLSpy にて直接編集するには、Archive View に表示されているファイルをダブルクリックします。または、Archive View にて目的のファイルを選択し、Archive View のツールバーからドキュメントを開くボタンをクリックします。XMLSpy にてファイルが開かれます。ファイルを編集した後に、**[ファイル | 保存]** コマンドにてファイルを保存するようにしてください。

アーカイブファイルの入力ヘルパー

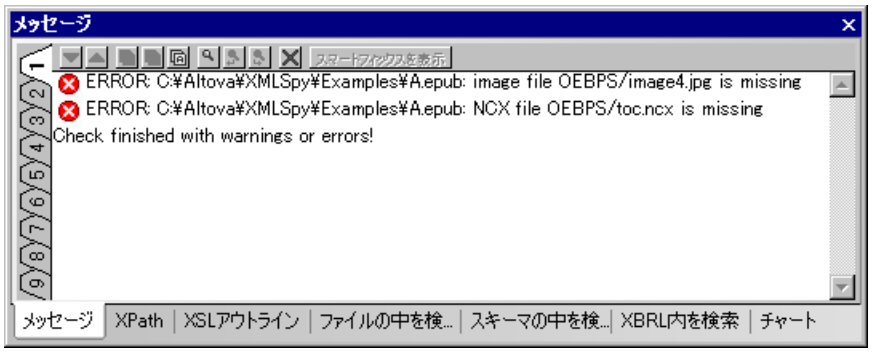
アーカイブファイルがXMLSpy にて開かれると、標準的なアーカイブファイルに対応した入力ヘルパーが表示されます。アーカイブファイルは以下のようなものになります：

- 通常 `content.opf` という名前のOPF ファイルにはEPUB ブックのメタデータが含まれます。Open Packaging Format (OPF) の仕様をベースにしています。
- 通常 `toc.ncx` という名前のNCX ファイル (XML に対するナビゲーションコントロールファイル) には、出版物の目次が含まれます。OPF 仕様のNCX パートをベースにしています。
- `META-INF` という名前のフォルダーには `container.xml` という名前のファイルが含まれており、ブック (OPF ファイル) のコンテンツを定義するファイルを指し示します。`container.xml` ファイルによりOpen Container

Format (OCF) 仕様のルールに従ったかたちでアーカイブファイルが構成されます。

EPUB ファイルの検証

EPUB ファイルを検証するには、**[XML | XML の検証]** コマンドを選択します。検証結果がメッセージウィンドウに表示されます (以下のスクリーンショットを参照)。アクティブなファイルのどれか (コンテンツファイルであれ構造に関連するファイルであれ) が変更され、まだ保存されていない場合、ファイルを保存するように促されます。EPUB ファイルの検証を行うには、変更されたファイルを保存する必要があります。Archive View のツールバーにある **[設定]** ボタンから、プレビューを行う前に、変更されたファイルを保存するように指定することができます。



エラーメッセージには、(i) エラーが発見されたファイルの場所と、可能な場合はエラーが発生した行番号、(ii) エラーの詳細が表示されます。上のスクリーンショットでは、`content.opf` ファイルの21行目にてエラーが発生したことが示されています。メッセージウィンドウ内のエラー行をクリックすることで関連するファイルが開かれ、エラーの箇所がハイライトされます。

メモ: EPUB 検証エンジンは Java のユーティリティーで、検証エンジンを使用するにはお使いのコンピューターに Java がインストールされていなければなりません。

15 データベース

XMLSpy を使えば様々なデータベース (DB) へ接続し、DB に対して問い合わせを行ったり、DB の構造をXML スキーマとしてインポート、DB からXML データファイルを生成したり、DB へデータをエクスポートすることができます。DB に関連した各機能はXMLSpy のメニューコマンドから扱うことができ、詳細はユーザーマニュアルに記述されています。これらのコマンドの完全なリストを、対応する記述へのリンクと共に以下に記します。

このセクションは、以下のように構成されます：

- データベースへの接続方法：XMLSpy のDB に関するコマンドを実行する際に必要なオペレーションを説明します。
- テストによりXMLSpy で既に使用できることが確認されているDB のリストを示します。

注意: 64ビット版のXMLSpy をお使いの場合は、DB への接続に使用されるデータベースドライバーも64ビットである必要があります。

XMLSpy のDB に関する機能

XMLSpy のDB に関する機能は[DB] ならびに[変換] メニューコマンドから利用できます。

- *データベースクエリの実行* (**[DB]** メニュー)：DB の構造を別のデータベースクエリウィンドウにコードし、DB への問い合わせを行ないます。結果はデータベースクエリウィンドウに表示されます。
- *IBM DB2* (**[DB]** メニュー)：IBM DB2 はXML DB で、XMLSpy によりXML DB のXML スキーマ管理や、XML DB の編集ならびに検証を行うことができます。
- *Oracle XML DB* (**[DB]** メニュー)：XML スキーマ管理、データベースの問い合わせ、そしてDB スキーマをベースにしたXML ファイルの生成など、Oracle XML DB に対する数多くの機能を提供します。
- *データベースのデータをインポート* (**[変換]** メニュー)：DB データをXML ファイルへインポートします。
- *データベース構造からXML スキーマを作成* (**[変換]** メニュー)：DB の構造をベースにしたXML スキーマを生成します。
- *XML スキーマを元にデータベースをインポート* (**[変換]** メニュー)：XMLSpy にてXML スキーマドキュメントがアクティブになっている状態で、DB の接続を確立し、選択されたDB テーブルのデータをインポートします。その結果作成されるXML ドキュメントはDB 接続を確立したときにアクティブになっているXML スキーマをベースとします。
- *XML スキーマを元にデータベース構造を作成* (**[変換]** メニュー)：既存のXML スキーマの構造をベースにした、データを持たないDB テーブルが作成されます。
- *データベースへエクスポート* (**[変換]** メニュー)：XML ドキュメントのデータをDB へエクスポートします。XML データを使って既存のDB テーブルを更新するか、XML データを含んだ新しいテーブルを作成することができます。

データ型の変換

XML ドキュメントとDB 間のデータ変換を行うにあたり、データの型も対応する形式に変換する必要があります。XMLSpy によるデータ型の変換方法は付録のDBにより生成されたXML のデータ型 とXML スキーマから生成されたDB のデータ型 に収められています。

Altova DatabaseSpy

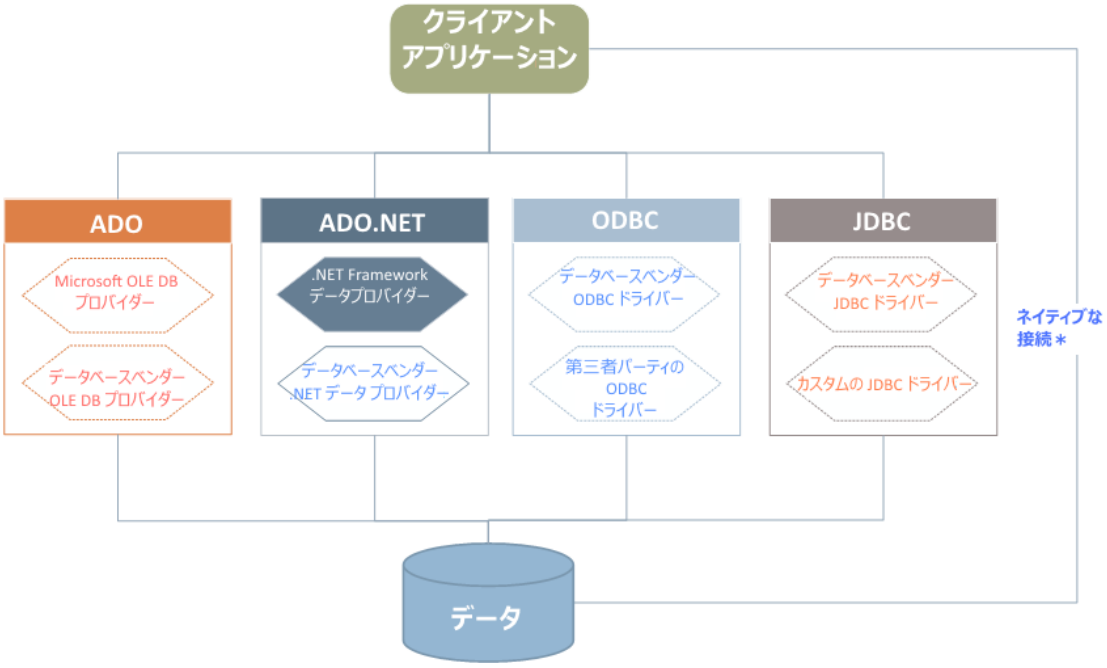
Altova のDatabaseSpy は複数のデータベースに対応したDB デザインツールで、XMLSpy にはない機能が提供されます。Altova DatabaseSpy に関する詳しい情報については、Altova ウェブサイトをご覧ください。

15.1 データベースへの接続

最も簡単なケースとして、データベースは、Microsoft Access、または、SQLite データベースファイルなどのローカルファイルが挙げられます。高度なケースの例としては、データベースは、アプリケーションが接続し、データを消費するオペレーティングシステムと同じオペレーティングシステムを使用するとは限らない、リモート、または、ネットワークデータベースサーバーに存在します。例えば、XMLSpy が、Windows で作動し、データ (例えばMySQL) へアクセスするデータベースがLinux マシーンで作動することは可能です。

多種のデータベース型とリモートおよびローカルで作業するには、XMLSpy は、データ接続インターフェイスとオペレーティングシステム内で使用することのできる、または、定期的にメジャーなデータベースベンダーによりリリースされるデータベースドライバーに従います。常に進化するデータベース技術の景観の中、このアプローチは、クロスプラットフォームに対する柔軟性および相互運用性に対応します。すなわち、XMLSpy により、以下のデータアクセス技術により主要なデータベースにアクセスすることができます

次の図は、最も簡単な方法で、(ジェネリックなクライアントアプリケーションとして表示されている)XMLSpy と (データベースサーバー、または、データベースファイルであることができる)データ保管の間で使用することのできるデータ接続のオプションを示しています。



** 直接なネイティブな接続 は、SQLite とPostgreSQL データベースのためにサポートされています。使用中のシステムにインストールするために追加ドライバーは必要ではありません。*

上の図で示されるように、XMLSpy は、主なデータベース型に次のアクセス技術を使用してアクセスすることができます :

- 基となるOLE DB (Object Linking and Embedding, Database) プロバイダーを使用するADO (Microsoft® ActiveX® Data Objects)
- ADO.NET (データのインタアクションを有効化するMicrosoft .NET Framework 内で使用することのできるライブラリのセット)
- JDBC (Java データベース接続)
- ODBC (Open データベース接続)

選択するデータベースへの接続のインターフェイスは、既存のソフトウェアのインフラストラクチャによります。データのアクセス技

術と 接続するデータベースと密に統合することのできる、データベースドライバーを通常選択します。例えば、Microsoft Access 2013 データベースに接続するには、**Microsoft Office Access データベース**Engine OLE DB **プロバイダー** などのネイティブなプロバイダを使用するADO 接続文字列を作成します。 に接続するには、他方、最新の JDBC、ODBC、 またはADO.NET インターフェイスをOracle Web サイトからダウンロードします。

Windows 製品（Microsoft Access またはSQL Serverなど）のためのドライバーは、Windows オペレーティングシステムで使用可能ですが、他のデータベースの種類では使用することができない可能性があります。メジャーなデータベースベンダーは、対応するデータベースにADO、ADO.NET、ODBC、 またはJDBCの組み合わせからクロスプラットフォームアクセスをすることができる、一般使用可能なデータベースクライアントソフトウェアとドライバーを定期的にリリースします。更に、上記の技術のためのサードパーティーのドライバーを使用することができます。多くの場合は、 1つ以上の方法で必要なデータベースへ使用中のオペレーティングシステムから接続することができます、この結果、XMLSpy からも接続することができます。使用可能な機能、パフォーマンスパラメーター、と既知の問題はデータアクセス技術および使用されるドライバーにより異なります。

15.1.1 データベース接続ウィザードの開始

データベース接続を必要とするアクションには、接続をセットアップするために必要なステップを案内するウィザードが表示されます。

ウィザードのステップを開始する前に、データベースの一部の種類は、データベースドライバーまたはデータベースクライアントソフトウェアなどの複数のデータベースの必須条件を個別にインストールまたは構成する必要があることに注意してください。これらは、通常各データベースのベンダーにより提供され、特定のWindows バージョン用に調整されたドキュメントを含みます。データベースの種類別にグループ化されたデータベースドライバーのリストに関しては、をデータベースドライバーの概要参照してください。

データベース接続ウィザードを開始するには次を行ってください：

- **DB** メニューから、**『データベースのクエリ』**をクリックします。

データベースの種類を選択した後、**『次へ』**をクリックします。スクリーンの指示はデータベースの種類（ADO、ODBC、JDBC）およびドライバーにより異なります。

例えば、各データベースの種類に適用できるサンプルはデータベース接続のサンプルのセクションを参照してください。各データベースへのアクセス技術に関しては以下のトピックを参照してください：

- ADO 接続の設定
- ADO.NET 接続の設定
- ODBC 接続の設定
- JDBC 接続の設定

15.1.2 データベースドライバー概要

以下のテーブル、特定のデータベースに特定のデータアクセス技術を使用して接続するためのはよく使用されるデータベースのドライバーのリストです。このリストは、完全または規範的なものではありません。以下に表示されているドライバに加え他のネイティブまたはサードパーティー代替を使用することができます。

(下のテーブル内でハイライトされている)Windows オペレーティングシステムのデフォルトの複数のデータベースドライバーを使用することができますが、代替ドライバーをダウンロードする必要がある可能性があります。データベースの一部に対して、オペレーティングシステムに付随するドライバーよりもデータベースベンダーより提供される最新のドライバーのパフォーマンスの方が良い場合があります。

一部の例外を除いて、データベースのベンダーは、通常必要なデータベースドライバーを含むデータベースクライアントソフトウェアを提供します。または、インストールするドライバーとコンポーネントのインストール中にオプションを与えます。データベースクライアントソフトウェアは、通常、データベースの管理と接続、およびデータベースクライアントとそのコンポーネントのインストールと構成を簡素化するために管理と構成ユーティリティにより構成されています。

データベースクライアントを正確に構成することは、データベースへの接続を成功させるために必須です。データベースをインストールして、使用する前に、各データベースバージョンおよび各 Windows バージョンにより通常異なるデータベースのインストールおよび構成の手順を確認することが奨励されます。

データベース	インターフェイス	ドライバー
Firebird	ADO.NET	Firebird ADO.NET データプロバイダー (http://www.firebirdsql.org/en/additional-downloads/)
	JDBC	Firebird JDBC ドライバー (http://www.firebirdsql.org/en/jdbc-driver/)
	ODBC	Firebird ODBC ドライバー (http://www.firebirdsql.org/en/odbc-driver/)
IBM DB2	ADO	DB2 のための IBM OLE DB プロバイダー
	ADO.NET	.NET のための IBM Data Server プロバイダー
	JDBC	JDBC とSQLJ のための IBM Data Server ドライバー
	ODBC	IBM DB2 ODBC ドライバー
IBM DB2 for i	ADO	• i5/OS IBMDA400 OLE DB プロバイダーのための IBM DB2 • i5/OS IBMDARLA OLE DB プロバイダーのための IBM DB2 • i5/OS IBMDASQL OLE DB プロバイダーのための IBM DB2
	ADO.NET	Windows .NET プロバイダーのための IBM i Access
	JDBC	Java JDBC ドライバーのための IBM Toolbox

データベース	インターフェイス	ドライバー
	ODBC	iSeries Access ODBC ドライバー
IBM Informix	ADO	IBM Informix OLE DB プロバイダー
	JDBC	IBM Informix JDBC ドライバー
	ODBC	IBM Informix ODBC ドライバー
Microsoft Access	ADO	• **Microsoft Jet OLE DB プロバイダー** • **Microsoft Access Database Engine OLE DB プロバイダー**
	ADO.NET	**OLE DB のための NET Framework データプロバイダー**
	ODBC	• **Microsoft Access ドライバー**
Microsoft SQL Server	ADO	• **SQL Server のための Microsoft OLE DB プロバイダー** • **SQL Server Native Client**
	ADO.NET	• **SQL Server のための NET Framework データプロバイダー** • **OLE DB のための NET Framework データプロバイダー**
	JDBC	• SQL Server のための Microsoft JDBC ドライバー (http://msdn.microsoft.com/en-us/data/aa937724.aspx)
	ODBC	• **SQL Server Native Client**
MySQL	ADO.NET	Connector/NET (http://dev.mysql.com/downloads/connector/net/)
	JDBC	Connector/J (http://dev.mysql.com/downloads/connector/j/)
	ODBC	Connector/ODBC (http://dev.mysql.com/downloads/connector/odbc/)
Oracle	ADO	• OLE DB のための Oracle プロバイダー • **Oracle のための Microsoft OLE DB プロバイダー**
	ADO.NET	.NET のための Oracle データプロバイダー (http://www.oracle.com/technetwork/topics/dotnet/index-085163.html)
	JDBC	• JDBC Thin ドライバー • JDBC Oracle Call Interface (OCI) ドライバー これらのドライバーは、通常、データベースクライアントのインストール中にインストールされます。(Thin ドライバーではない) OCI ドライバーを使用する接続は、Oracle XML DB のコンポーネントを使用しています。
	ODBC	• **Oracle のための Microsoft ODBC** • Oracle ODBC ドライバー (通常、Oracle データベースクライアントのインストール中にインストールされます)。
PostgreSQL	JDBC	Postgre JDBC ドライバー (https://jdbc.postgresql.org/download.html)
	ODBC	psqlODBC (https://odbc.postgresql.org/)
	ネイティブの接続	はい
Progress OpenEdge	JDBC	JDBC コネクタ (https://www.progress.com/jdbc/openedge)
	ODBC	ODBC コネクタ (https://www.progress.com/odbc/openedge)

データベース	インターフェイス	ドライバー
SQLite	ネイティブの接続	はい
Sybase	ADO	Sybase ASE OLE DB プロバイダー
	ADO.NET	ASE ADO.NET データプロバイダー
	JDBC	JDBC のためのjConnect™
	ODBC	Sybase ASE ODBC ドライバー

** 太字でハイライトされているドライバーはMicrosoft により提供されます。Windows システムにより利用することができない場合は、Microsoft Web サイトからダウンロードすることができます。*

ADO.NET プロバイダの一部はサポートされておらず、機能が制限されている場合があります。ADO.NET サポートに関するメモを参照してください。

データベースの種類に関して各データアクセステクノロジーの機能と制限を理解するには、特定のデータベース製品のドキュメントを参照し、特定の環境に対しての接続をテストします。よく発生する接続問題を回避するために、以下の全般的な注意と奨励を考慮してください:

- 32-ビット版と64ビット版のドライバーには互換性がない可能性がありますので、Altova アプリケーションに適用することのできる、ドライバーバージョンを構成してインストールしてください。例えば、64ビット版 オペレーティングシステムで 32ビット版 Altova アプリケーションを使用している場合、データベース接続は 32ビット版バージョンのドライバーを使用してください。
- 最新のドライバーのバージョンは、古いバージョンでは使用することのできない機能を含んでいる可能性があります。
- ODBC データソースをセットアップする際、データソース名 (DSN) をユーザー *DSN* の代わりに *システム DSN* として作成することが奨励されます。
- JDBC データソースをセットアップする際、JRE (Java Runtime Environment) がインストールされている、またオペレーティングシステムのCLASSPATH 環境変数が構成されていることを確認してください。
- Microsoft-により提供されるデータベースドライバーに適用することのできるサポートの詳細と既知の問題に関してはMSDN のドキュメントを参照してください。
- データベースベンダーからインストールするドライバーおよびデータベースクライアントソフトウェアのインストールの手順とサポートの詳細は、インストールパッケージのドキュメントを参照してください。公式なまたはサードパーティデータベースドライバーの場合、特定のオペレーティングシステムでの特定のドライバーに適用することのできる最も包括的な情報と構成手順は、通常ドライバーインストールパッケージの一部です。

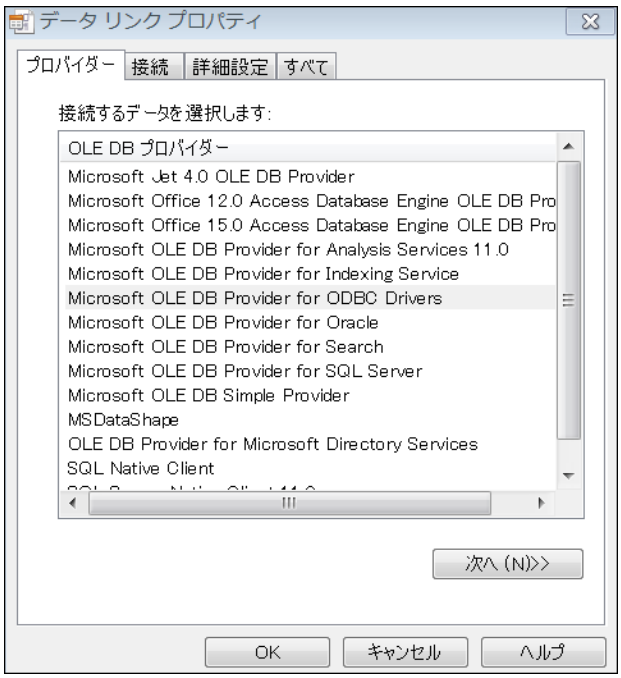
15.1.3 ADO 接続の設定

Microsoft ActiveX Data Objects (ADO) は、OLE DB を介して多種のデータソースに接続することのできるデータアクセス技術です。OLE DB は、ODBC またはJDBC の代替インターフェイスので、COM (Component Object Model コンポーネント・オブジェクト・モデル)環境内で、データへの一定のアクセスを提供します。ADO は、Microsoft Access およびSQL Server 等のMicrosoft ネイティブなデータベースへ接続する典型的な選択ですが、他のデータソースにも使用することができます。

ADO 接続のセットアップ:

1. データベース接続ウィザードの開始。
2. **[ADO 接続]** をクリックします。

3. **[構築]** をクリックします。



4. 接続するデータプロバイダーを選択します。下のテーブルは、よくある例をリストしています。

接続するデータベース...	使用するプロバイダー...
Microsoft Access	• **Microsoft Office Access Database Engine OLE DB Provider** Microsoft Access 2003 に接続する際には、**Microsoft Jet OLE DB Provider** も使用することができます。

接続するデータベース...	使用するプロバイダー...
SQL Server	• **SQL Server Native Client** • **Microsoft OLE DB Provider for SQL Server**
他のデータベース	データベースに適用することのできるプロバイダーを選択します。 データベースへのOLE DB プロバイダーが使用できない場合は、データベースベンダー (データベースドライバー概要を参照) から必要なドライバーをインストールします。別の方法として、ODBC またはJDBC 接続をセットアップします。 オペレーティングシステムに必要なデータベースに対するODBC ドライバーがある場合、**Microsoft OLE DB Provider for ODBC Drivers** を使用することができます。

5. **「次へ」** をクリックして、ウィザードを終了します。

次のウィザードのステップは選択されたプロバイダーに固有の手順です。SQL Server には、データベースユーザー名とパスワードに加え、データベースサーバーのホスト名を提供または選択する必要があります。Microsoft Access には、データベースファイルへのパスを参照または提供するように求められます。

初期化プロパティ (接続パラメーター) の完全なリストは、接続ダイアログボックスの**すべて** タブで使用することができます？ これらのプロパティは、選択されたプロバイダーにより異なります。次のセクションはMicrosoft Access and SQL Server データベースのための基本的な初期化プロパティの手引きを説明しています：

- SQL サーバー データリンク プロパティのセットアップ
- Microsoft Access データリンク プロパティのセットアップ

既存の Microsoft Access データベースへの接続

このアプローチは、パスワードで保護されていないMicrosoft Access データベースに接続する場合に適しています。データベースのパスワードの設定は、Microsoft Access (ADO) への接続で説明されています。

既存のMicrosoft Access データベースへの接続方法：

1. データベース接続ウィザードを実行します (データベース接続ウィザードの開始を参照)。
2. **Microsoft Access (ADO)** を選択して、**「次へ」** をクリックします。

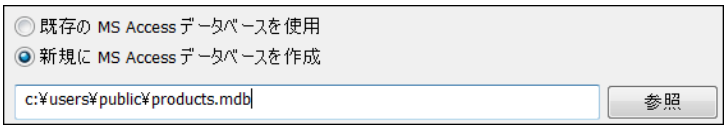
3. **[既存のMS Access データベースを使用]** を選択します。
4. データベースファイルを参照、または、(相対または絶対) パスを入力します。
5. **「接続」** をクリックします。

新規の Microsoft Access データベースの作成

既存のデータベースファイルに接続する代替として、Microsoft Access が使用中のマシンにインストールされていない場合でも、新規のMicrosoft Access データベースファイル (accdb, .mdb) を作成して、接続することができます。XMLSpy により作成されるデータベースは、空です。Microsoft Access または、DatabaseSpy (http://www.altova.com/ja/databasespy.html) などのツールを使用して、必要とされるデータベース構造を作成し、データを追加します。

新規の Microsoft Access データベースの作成方法：

1. データベース接続ウィザードを実行します（データベース接続ウィザードの開始を参照）。
2. **Microsoft Access (ADO)** を選択して、**「次へ」**をクリックします。



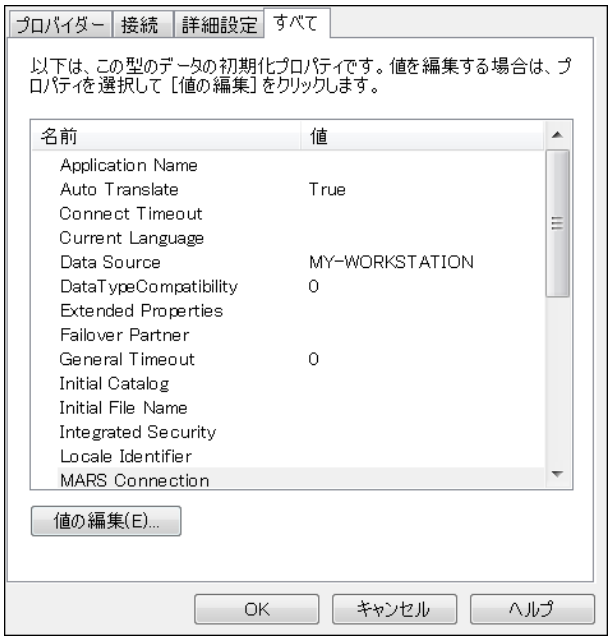
3. **[新規 MS Access データベースの作成]** を選択して、作成されるデータベースファイルの（相対または絶対）パスを入力します（例えば、**c:\users\public\products.mdb**）。または、**[参照]** をクリックして、フォルダーを選択して、ファイル名テキストボックスにデータベースファイルの名前（例えば、**products.mdb**）を入力し、**[保存]** をクリックします。

> **メモ**
> - データベースファイルを作成する場所に対して、書き込みアクセスの許可があることを確認してください。
> - **.accdb** フォーマットでファイルを作成する場合、Microsoft Access 2007 または以降が使用するコンピューターにインストールされている必要があります。**.mdb** フォーマットでファイルを作成する場合、上記の条件は必要ありません。

4. **「接続」**をクリックします。

SQL Server データ リンク プロパティの設定

Microsoft SQL Server データベースにADO（ADO 接続のセットアップを参照）を介して接続する場合、次のデータリンク プロパティが**全ての**タブのデータリンク プロパティダイアログボックスで正しく構成されていることを確認してください。

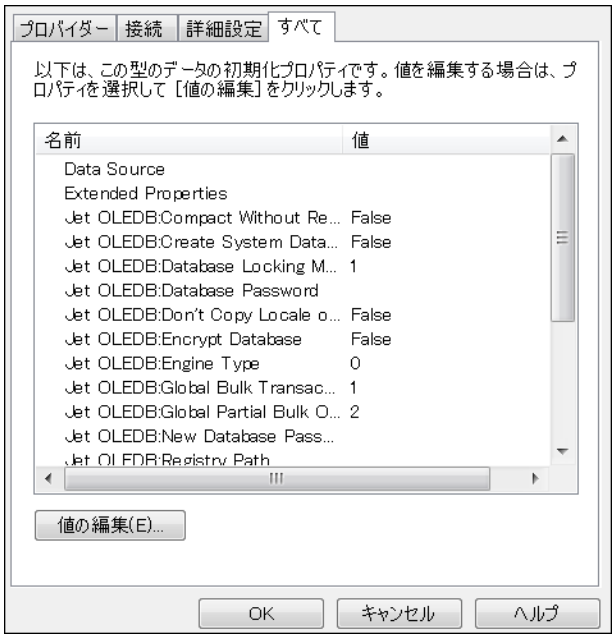


データ リンク プロパティダイアログ ボックス

プロパティ	メモ
統合セキュリティ	**プロバイダー**タブで**SQL Server Native Client** がデータプロバイダーに選択された場合は、このプロパティを、空白文字に設定します。
セキュリティ情報を保持	このプロパティを**True** に設定する。

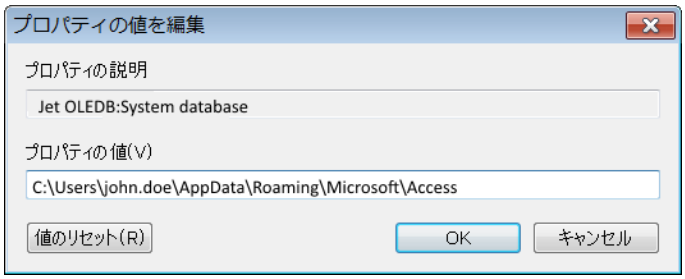
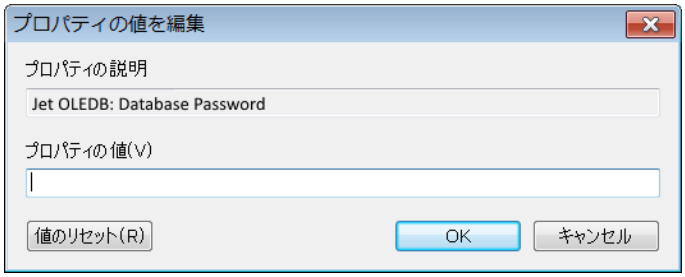
Microsoft Access データ リンク プロパティの設定

Microsoft Access データベースにADO (ADO 接続のセットアップを参照)を介して接続する場合、次のデータリンクプロパティが**全ての**タブのデータリンク プロパティダイアログボックスで正しく構成されていることを確認してください。



データ リンク プロパティダイアログ ボックス

プロパティ	メモ
データソース	このプロパティは、Microsoft Access データベースファイルへのパスを保管します。データベースへの接続問題を回避するために、UNC (Universal Naming Convention 汎用名前付け規則)パスフォーマットの使用が奨励されます。例 : `\\anyserver\share$\filepath`
Jet OLEDB: システムデータベース	このプロパティはワークグループ情報ファイルへのパスを保管します。データベースに接続する前に、このプロパティを明示的に設定する必要がある可能性があります。 "ワークグループ情報ファイル"エラーが理由で接続できない場合、ワークグループ情報ファイル (**System.MDW**) をユーザープロファイルに適用できるように設定します (手順はhttp://support.microsoft.com/kb/305542を参照してください)。プロパティの値を**System.MDW** ファイルのパスに設定します。

プロパティ	メモ
	
Jet OLEDB: データベースパスワード	データベースがパスワードで保護されている場合は、このプロパティの値をデータベースのパスワードに設定します。 

15.1.4 ADO.NET 接続のセットアップ

ADO.NET は、データベースからのデータを含むデータと作業するためにデザインされたMicrosoft .NET フレームワークライブラリのセットです。XMLSpy から、データベースに接続するには、ADO.NET、Microsoft .NET フレームワーク 4、または、以降を使用する必要があります。下に表示されているとおり、 NET プロバイダーを選択して、接続文字列を提供し、データベースにADO.NET を介して接続します。

.NET データプロバイダーは、コマンドを実行する特定の型のデータソースへの接続を可能にするクラスのコネクションです（例えば、SQL Server または Oracle データベースなど）。すなわち、ADO.NET とは、XMLSpy などのアプリケーションがデータベースプロバイダーを介してデータベースと作動することを意味します。各 データプロバイダーは特定の型のデータソースと作動するように最適化されています。NET プロバイダーには以下の2つの種類があります ：

1. デフォルトとしてMicrosoft .NET フレームワークを使用して提供されます。
2. 主要なデータベースベンダーを使用して NET フレームワークの拡張として提供されます。このような ADO.NET プロバイダーは、個別人ストールされている必要があり、通常それぞれのデータベースベンダーのウェブサイトからダウンロードすることができます。

メモ： 特定のADO.NET プロバイダーは、サポートされていない、または、サポートが制約されています。ADO.NET サポートに関するメモを参照してください。

ADO.NET 接続の作成 :

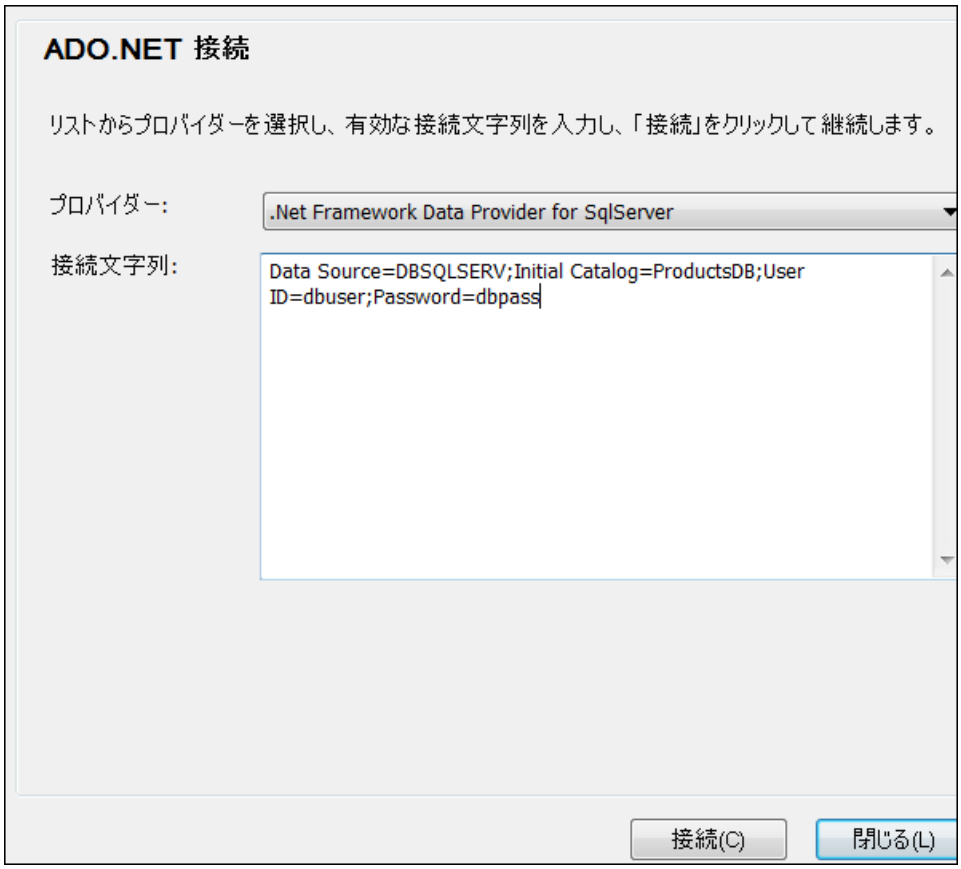
1. データベース接続ウィザードを開始します。
2. 「**ADO.NET 接続**」をクリックします。
3. .NET データプロバイダーをリストから選択します。

> .NET フレームワークと共に使用することのできるプロバイダーのリストはデフォルトでプロバイダーのリストに表示されます。ベンダーに特有の NET データプロバイダーは、使用中のシステムに既にインストールされている場合の

み使用することができます。 NET データプロバイダーは、データベースベンダーにより提供される、msi、または、exe ファイルを実行して、GAC (Global Assembly Cache) 内にインストールされる必要があります。

4. データベース接続文字列を入力します。接続文字列は、データベース接続情報をセミコロンで区切られた接続パラメーターのキー値ペアとして定義します。例えば、`Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;password=dbpass` などの文字列は、サーバー `DBSQLSERV` 上で、ユーザー名 `dbuser` とパスワード `dbpass` を使用して、SQL Server データベース `ProductsDB` に接続します。接続文字列を、キー値ペアを直接「接続文字列」ダイアログボックスに入力することにより作成することができます。更に便利な接続オプションは Visual Studio で作成することができます（次を参照：Visual Studio 内で接続文字列を作成する）。

接続文字列の構文は、プロバイダーリストより選択されたプロバイダーにより異なります。例えば、次を参照してください：サンプル ADO.NET 接続文字列。



5. **「接続」**をクリックします。

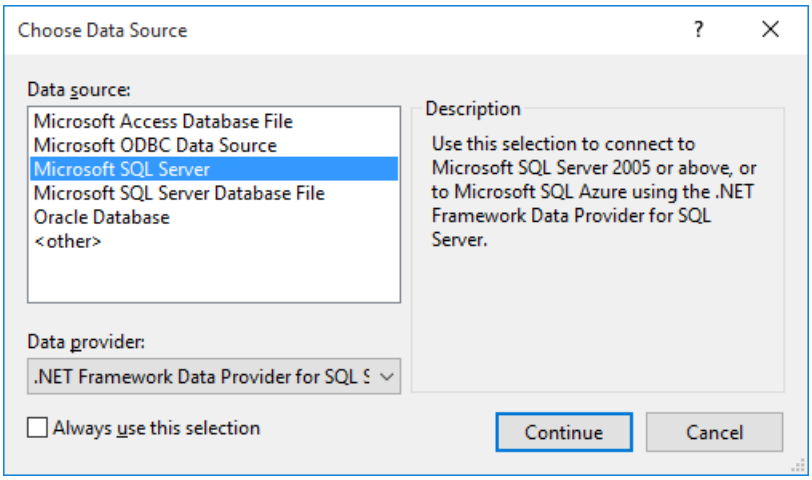
Visual Studio 内で接続文字列を作成する

ADO.NET を使用してデータソースに接続するには、有効な接続文字列が必要です。次の手順は、Visual Studio から接続文字列を作成する方法を説明しています。

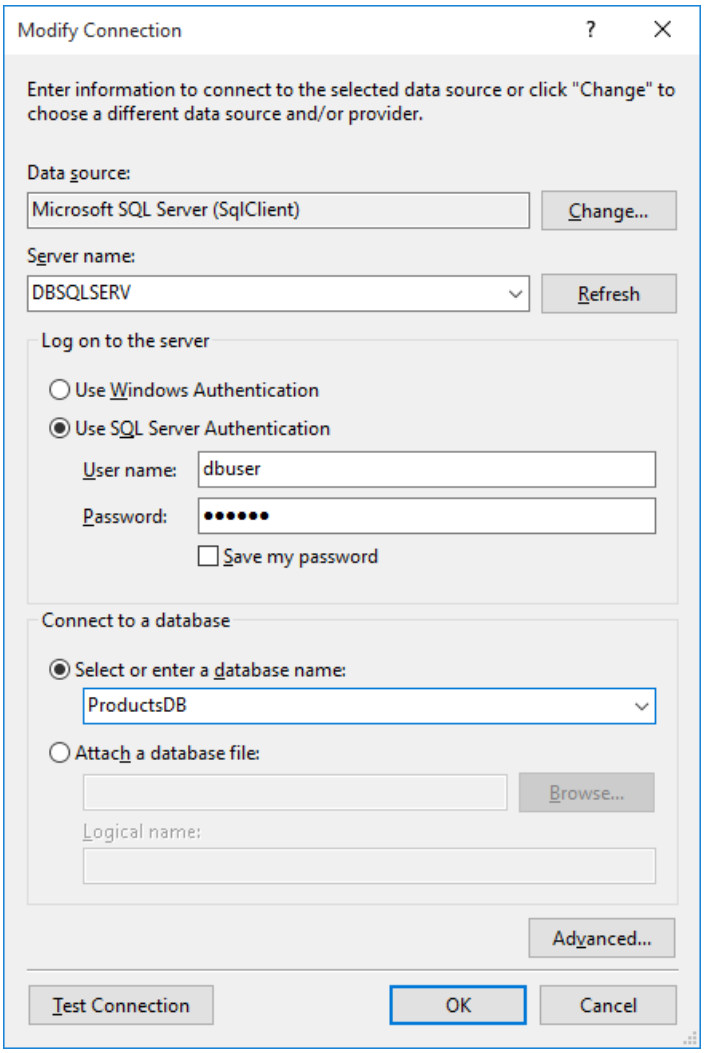
Visual Studio 内で接続文字列を作成する：

1. **「ツール」**メニューから、**「データベースへの接続」**をクリックします。
2. リストからデータソースを選択します（この例の場合、Microsoft SQL Server）。データプロバイダーは、選択に従

い自動的に記入されます。



3. **接続**をクリックします。



4. データベースへのサーバーホスト名、ユーザー名、およびパスワードを入力します。この例の場合、サーバー DBSQLSERV 上のデータベース ProductsDB に SQL Server 認証を使用して接続します。
5. 「OK」をクリックします。

データベースへの接続に成功した場合、サーバーエクスプローラー ウィンドウに表示されます。サーバーエクスプローラー ウィンドウを、メニューコマンド **表示 | サーバーエクスプローラー** を使用して表示することができます。データベース接続文字列を取得するには、サーバーエクスプローラー ウィンドウ内の接続を右クリックし、**「プロパティ」** を選択します。接続文字列が、Visual Studio プロパティウィンドウに表示されます。XMLSpy の「接続文字列」ボックスに文字列を貼り付ける前に、アスタリスク (*) 文字を実際のパスワードと置き換える必要があることに注意してください。

サンプル ADO.NET 接続文字列

ADO.NET 接続をセットアップするには、データベース接続ダイアログボックスから ADO.NET プロバイダー を選択し、接続文字列を入力します。 (次も参照してください ADO.NET 接続のセットアップ)。異なるデータベースのサンプル ADO.NET 接続文字列は .NET プロバイダーの下にリストされています。

IBM i のための .NET Framework データプロバイダー

このプロバイダーは、*IBM i Access Client Solutions - Windows Application Package* の一部としてインストールされます。サンプル接続文字列は以下のとおりです :

```
DataSource=ServerAddress;UserID=user;Password=password;DataCompression=True;
```

詳細に関しては、インストールパッケージ内に含まれる「.NET プロバイダー 技術レファレンス」ヘルプファイルを参照してください。

MySQL のための .NET Framework データプロバイダー

このプロバイダーを MySQL ウェブサイトからダウンロードすることができます (https://dev.mysql.com/downloads/connector/net/)。サンプル接続文字列は以下のとおりです :

```
Server=127.0.0.1;Uid=root;Pwd=12345;Database=test;
```

次も参照してください :https://dev.mysql.com/doc/connector-net/en/connector-net-programming-connecting-connection-string.html

SQL Server のための .NET Framework データプロバイダー

サンプル接続文字列は以下のとおりです :

```
Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User
ID=dbuser;Password=dbpass
```

次も参照してください :https://msdn.microsoft.com/en-us/library/ms254500(v=vs.110).aspx

.NET Framework 4.0 のための IBM DB2 データプロバイダー 10.1.2

```
Database=PRODUCTS;UID=user;Password=password;Server=localhost:50000;
```

メモ: このプロバイダーは、通常 IBM DB2 Data Server Client パッケージと共にインストールされます。IBM DB2 Data Server Client パッケージをインストール後 ADO.NET プロバイダーのリスト内でプロバイダーが不足している場合、次の技術メモを参照してください:http://www-01.ibm.com/support/docview.wss?uid=swg21429586。

次も参照してください:http://www.ibm.com/support/knowledgecenter/en/SSEPGG_10.1.0/com.ibm.swg.im.dbclient.adonet.ref.doc/doc/DB2ConnectionClassConnectionStringProperty.html

.NET (ODP.NET) のための Oracle データプロバイダー

ODP.NET プロバイダーを含むインストールパッケージをOracle ウェブサイトからダウンロードすることができます (次を参照してください:http://www.oracle.com/technetwork/topics/dotnet/downloads/index.html). サンプル接続文字列は以下のとおりです:

```
Data Source=DSORCL;User Id=user;Password=password;
```

Oracle への接続 (ODBC)内で説明されているように**tnsnames.ora** ファイル内で定義されたOracle サービス名をさす DSORCL がデータソース名の場合。

tnsnames.ora ファイル内のサービス名を構成せず接続する場合は、以下のような文字列を使用してください:

```
Data Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=host)
(PORT=port)))(CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=MyOracleSID)));User Id=user;Password=password;
```

次も参照してください:https://docs.oracle.com/cd/B28359_01/win.111/b28375/featConnecting.htm

ADO.NET サポートに関するメモ

下のテーブルは、XMLSpy 内でサポートされていない、または、制限の有るサポートを持つ、既知のADO.NET データベースドライバをリストしたテーブルです。

データベース	ドライバ	サポートに関するメモ
すべてのデータベース	**.Net Framework Data Provider for ODBC**	制限つきサポート。Microsoft Access 接続の際に既知の問題があります。ODBC への直接の接続を代わりに使用することが奨励されます。次を参照してください:ODBC 接続のセットアップ。
	.Net Framework Data Provider for OleDb	制限つきサポート。Microsoft Access 接続の際に既知の問題があります。ADO への直接の接続を代わりに使用することが奨励されます。次を参照してください:ADO 接続のセットアップ。
Firebird	**Firebird ADO.NET Data Provider**	制限つきサポート。ODBC または JDBC を代わりに使用することが奨励されます。次を参照してください:Firebird (ODBC) への接続 と Firebird (JDBC) への接続.
Informix	**IBM Informix Data Provider for .NET Framework 4.0**	サポートされていません。**DB2 Data Server Provider** を代わりに使用してください。
IBM DB2 for i (iSeries)	**.Net Framework Data Provider for i5/OS**	サポートされていません。*IBM i Access Client*

データベース	ドライバ	サポートに関するメモ
		Solutions - Windows Application Package の一部として提供されている**.Net Framework Data Provider for IBM i** を代わりに使用してください。
Oracle	**.Net Framework Data Provider for Oracle**	制限つきサポート。.NET Framework と共にこのドライバは提供されていますが、Microsoft により使用を奨励されていません。
PostgreSQL	-	このベンダーのための ADO.NET ドライバはサポートされていません。
Sybase	-	このベンダーのための ADO.NET ドライバはサポートされていません。

15.1.5 ODBC 接続の設定

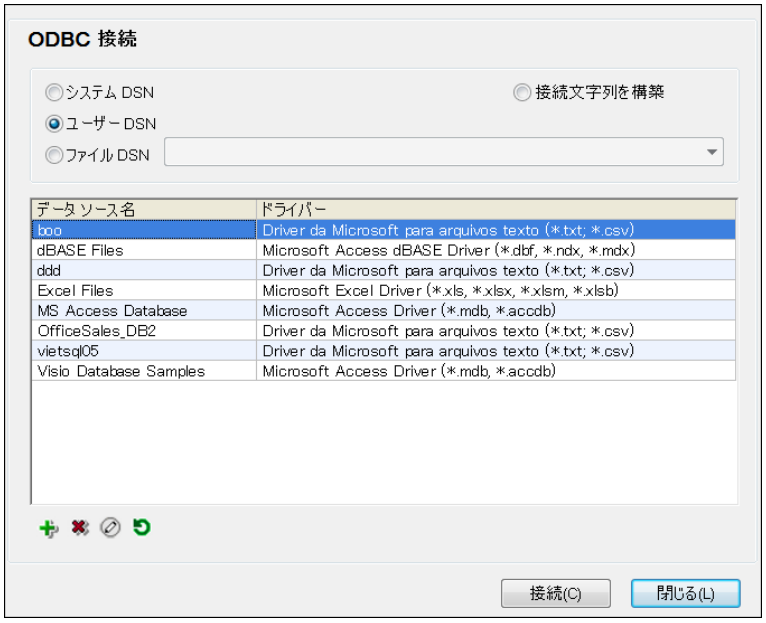
ODBC (Open Database Connectivity) は、一般に使用されているデータアクセス技術で、XMLSpy からデータベースに接続することができます。データベースにアクセスするための主要な手段として、またはOLE DB- や JDBC-による接続の代替として使用することができます。

ODBC を使用してデータベースに接続するには、最初にODBC データソース名 (DSN) を作成する必要があります。このステップは、他のユーザーまたはオペレーティングシステムによりDSN が既に作成されている場合必要ありません。DSN は、統一された、XMLSpy を含むオペレーティングシステム上のODBC に対応したクライアントアプリのデータベース接続方法を表します。DSN には以下の種類があります :

- システムDSN
- ユーザーDSN
- ファイルDSN

オペレーティングシステムで特権を持つすべてのユーザーはシステム データソースにアクセスすることができます。作成したユーザーのみがユーザー データソースにアクセスすることができます。*ファイルDSN* を作成した場合、データソースは dsn 拡張子を持つファイルとして作成され、データソースにより使用されているドライバーがインストールされていることが前提で、他のユーザーと共有することができます。

ODBC 接続ダイアログボックス内の**[ODBC 接続]** をクリックすると、使用中のコンピューターで既に使用することのできるDSN はデータベース接続ダイアログボックスに表示されています。



ODBC 接続ダイアログボックス

データベースにより必要とされるDSN が無い場合、XMLSpy データベース接続ウィザードが作成の手助けをします。しかし、Windows オペレーティングシステムに直接作成することもできます。いずれにせよ、続行する前に、使用中のデータベースにODBC ドライバーがオペレーティングシステムのODBC ドライバーリストに適用できることを確認してください。(使用可能なODBC ドライバーの表示を参照)。

新規 DSN を使用しての接続 :

1. データベース接続ウィザードを開始します。
2. データベース接続ダイアログボックスの **[ODBC 接続]** をクリックします。
3. ソースの種類を選択します (ユーザーDSN、システムDSN、ファイルDSN)。

   > システムDSN を作成するには、オペレーティングシステムの管理者権限が必要です。

4. **「追加」** をクリックします。
5. ドライバーを選択して、(作成するDSN の種類に応じて) **[ユーザーDSN]** または **[システムDSN]** をクリックします。使用中のデータベースに適用することのできるドライバーがリストされていない場合、データベースベンダーからダウンロードして、インストールしてください。(次を参照 : データベースドライバー概要)。
6. ポップアップするダイアログボックスに特定の接続情報を入力してセットアップを完了します。

接続を成功させるには、データベースサーバーのホスト名(またはIP アドレス)、とデータベースのユーザー名およびパスワードを提供する必要があります。他の接続パラメーターがある可能性がありますが、これらのパラメーターはデータベースプロバイダーにより異なります。各接続方法に固有のパラメーターについての詳細は、ドライバープロバイダーのドキュメントを参照してください。一度作成されると、DSN はデータソース名のリストで使用することができます。ユーザーDSN はユーザーDSN のリストに追加されますが、システムDSN はシステムDSN のリストに追加されます。

既存のDSNを使用しての接続 :

1. データベース接続ウィザードを開始します.
2. **[ODBC 接続]** をクリックします。
3. 既存のデータソース (ユーザーDSN、システムDSN、ファイルDSN) から種類を選択します
4. 既存のDSN をクリックして、 **「接続」** をクリックします。

既存の dsn ファイルをベースにして接続文字列を構築する:

1. データベース接続ウィザードを開始します.
2. **[ODBC 接続]** をクリックします。
3. **[接続文字列の構築]** を選択して、**[構築]** をクリックします。
4. ファイルDSN を使用して、接続文字列を構築する場合、**[ファイルデータソース]** タブをクリックします。それ以外の場合は、**[マシーンデータソース]** タブをクリックします。(システムDSN とユーザーDSN は "マシン"データソースとして知られています。)
5. 必要とされる dsn ファイルを選択して、 **「OK」** をクリックします。

用意された接続文字列を使用して接続する:

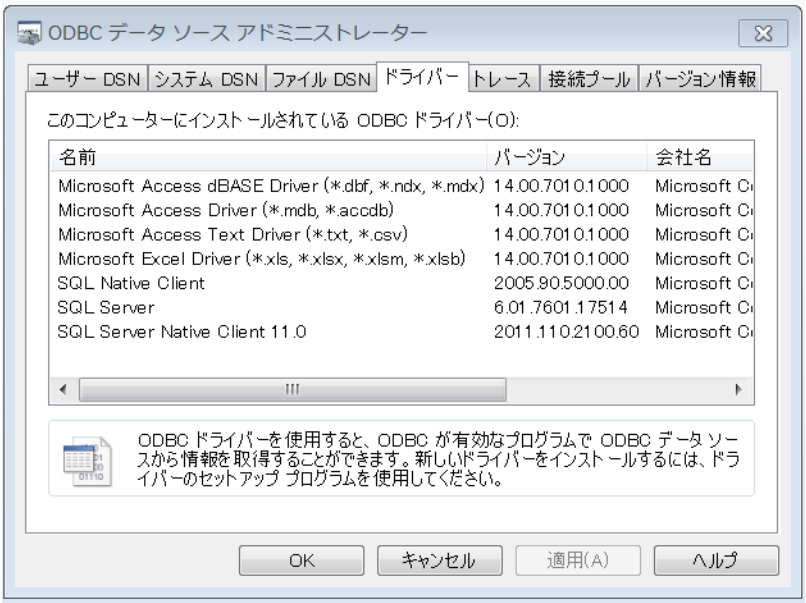
1. データベース接続ウィザードを開始します.
2. **[ODBC 接続]** をクリックします。
3. **[接続文字列の構築]** を選択しています。
4. 接続文字列を与えられたボックスに貼り付け、 **「接続」** をクリックします。

使用可能な ODBC ドライバーの表示

ODBC データソース管理で使用中のオペレーティングシステムで使用可能なODBC ドライバーを確認することができます。Windows コントロールパネルの**管理ツール**からODBC データソース管理 (**Odbcad32.exe**) にアクセスすることができます。 64 ビット版オペレーティングシステムには、2種類のバージョンの実行可能ファイルがあります:

- **C:\Windows\SysWoW64** ディレクトリにある、**Odbcad32.exe** ファイルの 32 ビット版バージョン (**C:** が使用中のシステムドライブとして前提されています)。
- **C:\Windows\System32** ディレクトリにある、**Odbcad32.exe** ファイルの 64 ビット版バージョン

インストールされている 32 ビット版データベースドライバーは、ODBC データソース管理の 32 ビット版バージョンで、64-ビット版バージョン内の 64 ビット版ドライバーで確認することができます。ですから、対応するバージョンのODBC データソース管理でデータベースドライバーを確認してください。



ODBC データソース管理者

ターゲットデータベースのドライバーがリストに存在しない場合、または、代替ドライバーを追加する場合、データベースのベンダーからダウンロードしてください（データベースドライバー概要 を参照）。ODBC ドライバーがシステムで使用可能になると、ODBC 接続を作成することができます（ODBC 接続のセットアップを参照）。

15.1.6 JDBC 接続の設定

JDBC (Java Database Connectivity) は、Oracle の Java ソフトウェアプラットフォームの一部であるデータベースアクセスインターフェイスです。JDBC 接続は ODBC 接続に比べ、通常よりもリソースを集中的に使用する接続ですが、ODBC で使用することのできない機能を使用することができます。ODBC コネクタにより使用できないデータベース機能を使用する場合 JDBC 接続を使用することが奨励されています。

前提条件

- Java Runtime Environment (JRE) または Java Development Kit (JDK) がオペレーティングシステムにインストールされている必要があります。インストールされていない場合は Java の公式 Web サイトからパッケージとインストールの手順をダウンロードしてください。
- データベースベンダーからの JDBC ドライバーがインストールされている必要があります。Oracle データベースに接続する場合は、一部の Oracle ドライバーは特定の JRE バージョンに固有のため、追加コンポーネントと構成が必要な点に注意してください。Oracle 製品のドキュメント（例えば、『JDBC 開発者ガイドおよびリファレンス』）には、各 JDBC ドライバーの構成の手続きの詳細が説明されています。
- オペレーティングシステムの `PATH` 環境変数は、JRE または JDK インストールディレクトリの `bin` ディレクトリへのパスを含む必要があります。例えば、`C:\Program Files (x86)\Java\jre1.8.0_51\bin` など。
- `CLASSPATH` 環境変数は、(jar ファイルの 1 つまたは複数の) 使用中の Windows オペレーティングシステムの JDBC ドライバーのパスを含む必要があります。データベースクライアントをインストールする際に、インストーラーはこの変数を自動的に構成する可能性があります。JDBC ドライバーのドキュメントは通常ステップバイステップで CLASSPATH 変数の設定方法を説明しています（CLASSPATH の構成を参照）。

JDBC 接続の設定方法

1. データベース接続ウィザードを開始します。
2. 『**JDBC 接続**』をクリックします。
3. 任意で、セミコロンで区切られた jar ファイルパスのリストを『クラスパス』テキストボックスに入力することもできます。ここで入力される .jar ライブラリは、`CLASSPATH` 環境変数内で既に定義されているものに加え、環境にロードされます。『クラスパス』テキストボックスの編集が完了すると、ソース jar ライブラリ内で検索される JDBC ドライバーは、『ドライバー』リストに自動的に追加されます（次のステップを参照してください）。
4. 『ドライバー』の横で JDBC ドライバーをリストから選択、または、Java クラス名を入力します。`CLASSPATH` 環境変数により構成された JDBC ドライバーおよび、『クラスパス』テキストボックス内で検出されたドライバーが含まれます（次を参照：CLASSPATH の構成）。

> CLASSPATH 変数内で定義されている JDBC ドライバーのパスを使用することができます。また、データベース接続ダイアログボックスに直接入力された jar パスは Java Virtual Machine (JVM) によりサポートされています。JVM は接続を構築するためのドライバーを決定します。JVM にロードされる Java クラスを確認し、JDBC ドライバーの競合、および、データベースに接続する際に発生する予期されないエラーを回避するようにしてください。

5. 対応するボックスに、ユーザー名とパスワードを入力します。
6. データベース URL テキストボックス内に、JDBC 接続 URL (文字列) をデータベースの型のとおり入力します。次のテーブルは一般のデータベースの型のための JDBC 接続 URL (文字列) の構文について説明しています。

データベース	JDBC 接続 URL
Firebird	`jdbc:firebirdsql://<host>[:<port>]/<database path or alias>`

データベース	JDBC 接続 URL
IBM DB2	jdbc:db2://**hostName**:**port**/**databaseName**
IBM Informix	jdbc:informix-sqli://**hostName**:**port**/**databaseName**:INFORMIXSERVER=**myserver**
Microsoft SQL Server	jdbc:sqlserver://**hostName**:**port**;**databaseName**=name
MySQL	jdbc:mysql://**hostName**:**port**/**databaseName**
Oracle	jdbc:oracle:thin:@//**hostName**:**port**:**service**
Oracle XML DB	jdbc:oracle:oci:@//**hostName**:**port**:**service**
PostgreSQL	jdbc:postgresql://**hostName**:**port**/**databaseName**
Progress OpenEdge	jdbc:datadirect:openedge://**host**:**port**;databaseName=**db_name**
Sybase	jdbc:sybase:Tds:**hostName**:**port**/**databaseName**

メモ: 上記のフォーマットに対する構文のバリエーションも使用することができます (例えば、データベースURL はポートを除外、またはデータベースのユーザー名とパスワードを含む可能性があります)。詳細はデータベースのベンダーのドキュメントを参照してください。

7. **接続** をクリックします。

CLASSPATH の構成

CLASSPATH 環境変数はクラスと、オペレーティングシステム上の他のリソースファイルを検索するため Java Runtime Environment (JRE) により使用されます。JDBC を介してデータベースに接続すると、この変数はオペレーティングシステム上の JDBC ドライバーへのパス、または一部の場合、使用中のデータベースの種類に固有の追加ライブラリファイルへのパスを含むように構成されなければなりません。

下のテーブルは、CLASSPATH 変数に通常含まれる必要のあるサンプルファイルパスをリストしています。重要なことは、システム上の JDBC ドライバのロケーション、JDBC ドライバー名、オペレーティングシステム上のドライバーの場所をベースにこの情報を調整する必要があります。接続問題を避けるために、使用中のオペレーティングシステムにインストールされている JDBC ドライバに適用されるインストールの指示、インストール前とインストール後の構成ステップを確認してください。

データベース	サンプル CLASSPATH エントリ
IBM DB2	C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc.jar;C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc_license_cu.jar;
IBM Informix	C:\Informix_JDBC_Driver\lib\ifxjdbc.jar;
Microsoft SQL Server	C:\Program Files\Microsoft JDBC Driver 4.0 for SQL Server\sqljdbc_4.0\enu\sqljdbc.jar
MySQL	mysql-connector-java-*version*-bin.jar;
Oracle	**ORACLE_HOME**\jdbc\lib\ojdbc6.jar;
Oracle (with XML DB)	**ORACLE_HOME**\jdbc\lib\ojdbc6.jar;**ORACLE_HOME**\LIB\xmlparserv2.jar;**ORACLE_HOME**\RDBMS\jlib\xdb.jar;
PostgreSQL	**<installation directory>**\postgresql.jar

データベース	サンプルCLASSPATH エントリ
Sybase	C:\sybase\jConnect-5_5\classes\jconn2.jar

- CLASSPATH 変数の変更は、使用中のコンピュータのJava アプリケーションの振る舞いに影響する可能性があります。続行する前に生じえる影響を把握するために、Java ドキュメントを参照してください。
- 環境変数は、ユーザーまたはシステムであることができます。システム環境変数を変更するには、オペレーティングシステムに対する管理者権限が必要です。
- 環境変数を変更後、設定の変更を有効にするには、プログラムを再起動してください。または、オペレーティングシステムをログオフまたは再起動してください。

Windows 7 でCLASSPATH を構成する:

1. **[スタート]** メニューを開始して、**[コンピューター]** を右クリックします。
2. **[プロパティ]** をクリックします。
3. **[システムの詳細設定]** をクリックします。
4. **[詳細設定]** タブから**[環境変数]** をクリックします。
5. CLASSPATH 変数を、ユーザーまたはシステム環境変数からロケートして、編集をクリックします。CLASSPATH 変数が存在しない場合は、**[新規]** をクリックして作成します。
6. オペレーティングシステム内でJDBC ドライバーがある場所にパスを含むように変数の値を編集します。JDBC ドライバーパスを他の既にCLASSPATH 変数内にあるパスから区別するためにセミコロン(;)で区切ります。

Windows 8、Windows 10 でCLASSPATH を構成する:

1. Windows のスタートボタンを右クリックして、**[システム]** をクリックします。
2. **[高度なシステム設定]** をクリックします。
3. **[環境変数]** をクリックします。
4. CLASSPATH 変数を、ユーザーまたはシステム環境変数からロケートして、編集をクリックします。CLASSPATH 変数が存在しない場合は、**[新規]** をクリックして作成します。
5. オペレーティングシステム内でJDBC ドライバーがある場所にパスを含むように変数の値を編集します。JDBC ドライバーパスを他の既にCLASSPATH 変数内にあるパスから区別するためにセミコロン(;)で区切ります。

15.1.7 PostgreSQL 接続のセットアップ

PostgreSQL データベースへの接続は、ネイティブな接続として、または、ODBC、JDBC、および他のドライバーを使用して確立することができます。ネイティブな接続の利点は、使用中のシステムにインストールする場合、ドライバーを必要としないことです。

ドライバーを使用して接続を構築する場合は、次を参照してください:データベースドライバーの概要. JDBC を使用しての接続に関しては、次を参照してください:JDBC 接続のセットアップ。ODBC を使用したPostgreSQL への接続の例は、次を参照してください:PostgreSQL (ODBC) への接続 。

それ以外の場合、PostgreSQL へのネイティブの接続をセットアップする場合、以下の手順に従ってください。継続するには、以下の前提条件が必要です :ホスト名、ポート、データベース名、ユーザー名、およびパスワード。

ネイティブPostgreSQL 接続のセットアップ:

1. データベース接続ウィザードを開始します.
2. **「PostgreSQL 接続」**をクリックします。
3. 対応するボックスに、ホスト(PostgreSQL が同じマシンで作動している場合、*ローカルホスト*)、ポート(通常、5432 ですが、これは任意です)、データベース名、ユーザー名とパスワードを入力します。

4. **接続** をクリックします。

PostgreSQL データベースサーバーが異なるマシンに存在する場合、次の点に注意してください:

- PostgreSQL データベースサーバーは、クライアントからの接続を受け入れるように構成されなければなりません。具体的には、**pg_hba.conf** ファイルがローカルではない接続を許可するように構成される必要があります。第 2 に、**postgresql.conf** ファイルは、特定された IP アドレスとポートをリスンするように構成される必要があります。詳細に関しては、PostgreSQL ドキュメントを参照してください (https://www.postgresql.org/docs/9.5/static/client-authentication-problems.html)。
- サーバーマシンがファイアウォールを越えてデザインされたポート (通常、5432) 上で接続を受け入れるように構成されている必要があります。例えば、**「コントロールパネル > Windows ファイアウォール > 詳細設定 > 受信の規則」** で設定すると、Windows 上で作動するデータベースサーバーでは、ファイアーウォールを越えてポート 5432 に接続を許可するルールを作成する必要がない場合があります。

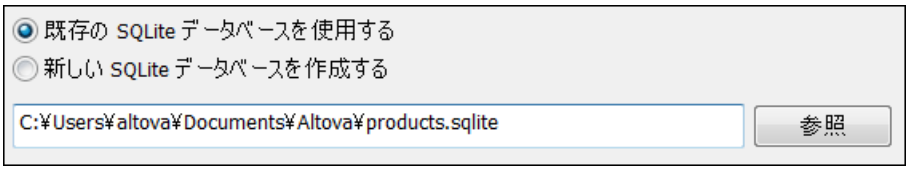
15.1.8 SQLite 接続の設定

SQLite (http://www.sqlite.org) は、ポータブル性とインストールの簡便性が重要な場合理想的な構成のファイルベースの、内蔵のデータ型です。Since SQLite データベースは、XMLSpy によりネイティブにサポートされるため、接続のためにドライバーをインストールする必要はありません。

既存の SQLite データベースへの接続

既存の SQLite データベースへの接続方法:

1. データベース接続ウィザードを実行します (データベース接続ウィザードの開始を参照)。
2. **SQLite** を選択して、**「次へ」** をクリックします。



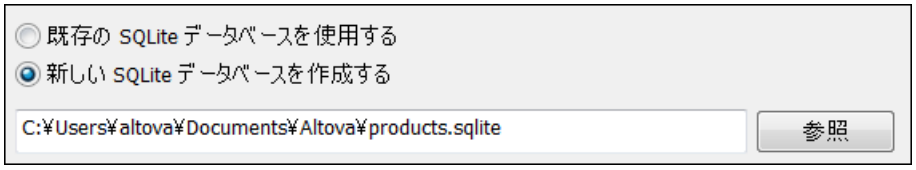
3. **[既存の SQLite データベースを使用]** を選択して、SQLite データベースファイルを参照、またはデータベースへの (相対または絶対) パスを入力します。 **「接続」** ボタンは、データベースファイルへのパスを入力すると有効化されます。
4. **「接続」** をクリックします。

新規の SQLite データベースの作成

既存のデータベースファイルに接続する代替として、新規の SQLite データベースファイルを作成して、接続することができます。XMLSpy により作成されるデータベースは、空です。クエリまたはスクリプトを使用して、必要とされるデータベース構造を作成し、データを追加します。

新規の SQLite データベースの作成方法：

1. データベース接続ウィザードを実行します (データベース接続ウィザードの開始を参照)。
2. **SQLite** を選択して、**「次へ」** をクリックします。



3. **[新規 SQLite データベースの作成]** を選択して、作成されるデータベースファイルの (相対または絶対) パスを入力します (例えば、**c:\users\public\products.sqlite**)。または、**[参照]** をクリックして、フォルダーを選択して、ファイル名テキストボックスにデータベースファイルの名前 (例えば、**products.sqlite**) を入力し、**[保存]** をクリックします。

> データベースファイルを作成する場所に対して、書き込みアクセスの許可があることを確認してください。

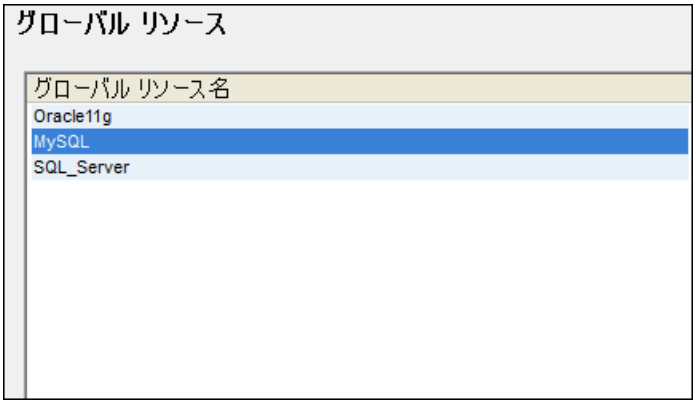
1. **「接続」** をクリックします。

15.1.9 グローバルリソースからの接続の使用

以前にデータベースをグローバルリソースとして使用可能に構成している場合、接続をいつでも再使用することができます (異なる Altova アプリケーション間でも使用可能)。

グローバルリソースからのデータベース接続の使用：

1. データベース接続ウィザードを開始します。
2. **[グローバルリソース]** をクリックします。グローバルリソースとして使用可能なデータベース接続がリストされます。



3. データベース接続履歴から **[接続]** をクリックします。

ヒント: 各グローバルリソースの追加情報を取得するために、マウスカーソルをグローバルリソースにポイントします。

15.1.10 データベース接続のサンプル

このセクションは、XMLSpy からデータベースへの接続のサンプル手順を説明します。Windows マシン、ネットワーク環境、とデータベースクライアントまたはサーバーソフトウェアは、次に挙げられるサンプルの構成とは完全に一致しない可能性があります。

メモ: ほぼすべてのデータベースの型では、1つ以上のデータアクセステクノロジー (ADO, ODBC, JDBC) またはドライバーを使用して接続することができます。データベース接続パフォーマンス、およびドライバーの機能と制限は、XMLSpy 外で構成された選択されたドライバー、(該当する場合) データベースクライアントソフトウェア、および追加接続パラメータなどによって異なります。

Firebird (ODBC) への接続

このトピックは、Linux サーバー上で実行される、Firebird 2.5.4 データベースへの接続のサンプルの命令を説明しています。

必要条件:

- Firebird データベースサーバーは、クライアントからの TCP/IP 接続を受け入れるように構成されています。
- Firebird ODBC ドライバーを使用中のオペレーティングシステムにインストールする必要があります。このサンプルは、を使用します。 Firebird ODBC ドライバー バージョン 2.0.3.154 は、Firebird ウェブサイト (http://www.firebirdsql.org/) からダウンロードすることができます。
- Firebird クライアントを、使用中のオペレーティングシステムにインストールする必要があります。 Firebird 2.5.4 クライアントのためのスタンドアロンのインストーラーが存在しない点に注意してください。また、クライアントは、サーバーのインストールパックの一部です。Firebird サーバー インストールパックを Firebird ウェブサイト (http://www.firebirdsql.org/) からダウンロードすることができます。ダウンロードの際、[スーパークラシック/クラシック または スーパーサーバーのための Windows 実行ファイル] インストーラーを選択してください。クライアントファイルのみをインストールする場合、**[最小クライアントインストール-サーバーツール無し]** を選択します。

  重要点:

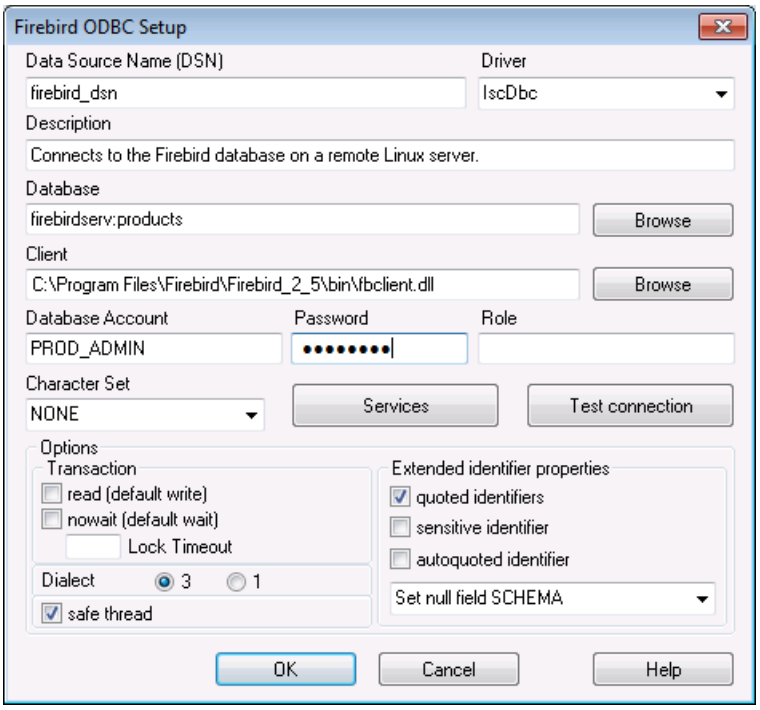
  - Firebird ODBC ドライバー および クライアント (32 ビット または 64 ビット) のプラットフォームは、XMLSpy のプラットフォームに適応する必要があります。
  - Firebird クライアントのバージョンは、接続されている Firebird サーバーのバージョンに適応する必要があります。

- データベース接続の詳細 サーバー上の サーバー ホスト名 または IP アドレス、データベース パス (または エイリアス)、ユーザー名、および パスワード。

ODBC を使用して Firebird に接続する:

1. データベース接続ウィザードを開始します。
2. **[ODBC 接続]** をクリックする。
3. **[ユーザー DSN]** を選択して、(または、管理者の特権がある場合、**[システム DSN]** を選択して、**[追加]** をクリックします。

4. Firebird ドライバーを選択して、をクリックします。そして**[ユーザー DSN]** (前のステップでの選択により、または**[システム DSN]**)。Firebird ドライバーがリスト内に存在しない場合、使用中のオペレーティングシステムにインストールされていることを確認してください (使用可能な ODBC ドライバーの表示 を参照)。



5. データベース接続の詳細を入力します：

データソース名 (DSN)	作成する詳細名とデータソースを入力します。
データベース	サーバー ホスト名 または IP アドレスをコロンとデータベース エイリアス (または パス) の後に入力します。このサンプルでは、ホスト名は、`firebirdserv`、データベースエイリアスは、`products` です: `firebirdserv:products` データベース エイリアスを使用することにより、サーバー側上では、データベース管理者は、エイリアス製品を構成し、サーバー上の実際の Firebird (.fdb) データベースファイルをポイントします (詳細は、Firebird のドキュメントを参照してください)。

	ホスト名の代わりに サーバーIP アドレス、エイリアスの代わりにパスを使用することもできます。ですから、以下のサンプル接続の文字列は有効です：
	
	`firebirdserver:/var/Firebird/databases/butterflies.fdb`
	`127.0.0.1:D:\Misc\Lenders.fdb`
	データベースが Windows マシン上に存在する場合、**[参照]** をクリックして、Firebird (.fdb) データベースファイルを直接選択します。

クライアント	**fbclient.dll** ファイルへのパスを入力します。デフォルトでは、これは Firebird インストールディレクトリのサブディレクトリです。
データベース アカウント	データベース管理者 (このサンプルの場合 `PROD_ADMIN`) により与えられたデータベースユーザー名を入力します。
パスワード	データベース管理者により与えられたデータベースパスワードを入力します。

6. **「OK」** をクリックします。

Firebird (JDBC) への接続

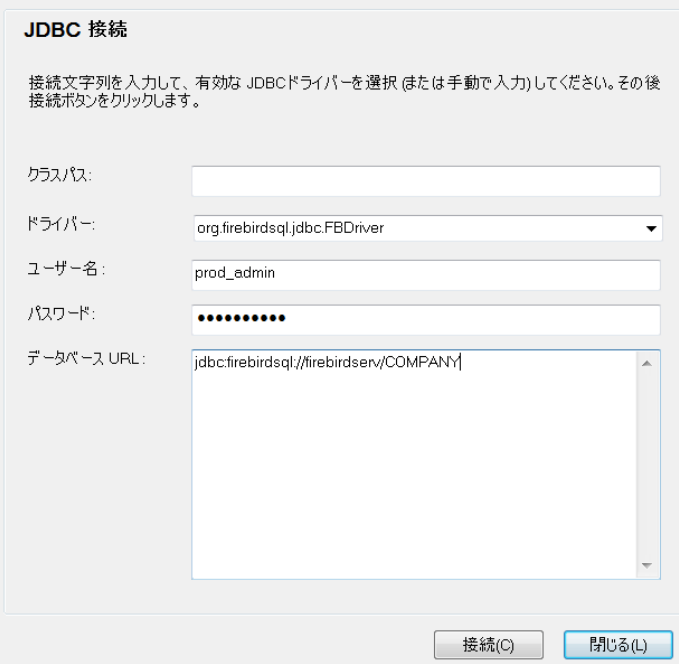
このトピックは、を使用してのFirebird データベースサーバーへの接続のサンプルの命令を説明しています。

必要条件：

- Java Runtime Environment (JRE) またはJava Development Kit (JDK) が使用中のオペレーティングシステムにインストールされている必要があります。
- オペレーティングシステムの `PATH` 環境変数は、JRE またはJDK のインストールディレクトリの `bin` ディレクトリへのパスを含む必要があります。例えば、`C:\Program Files (x86)\Java\jre1.8.0_51\bin`.
- Firebird JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります (データベースへの接続を可能にする jar ファイルのフォームをとります)。ドライバーはFirebird Web サイトからダウンロードすることができます (http://www.firebirdsql.org/)。このサンプルは、*Jaybird 2.2.8* を使用します。
- データベース接続の詳細があります：ホスト、データベースパス、またはエイリアス、ユーザー名、およびパスワード。

JDBC を使用してFirebird に接続する：

1. データベース接続ウィザードを開始します.
2. 「**JDBC 接続**」をクリックします。.
3. 「クラスパス」の横にデータベースへの接続を与える jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる jar ファイルは次のパスを使用して見つけることができます：**C:\jdbc\firebird\jaybird-full-2.2.8.jar**。jar ファイルパスをオペレーティングシステムのCLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照：CLASSPATH の構成)。
4. 「ドライバー」ボックスから、**org.firebirdsql.jdbc.FBDriver** を選択します。このエントリは、有効な jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムのCLASSPATH 環境変数内で検出された場合のみ使用することができます (前のステップを参照してください)。



5. 対応するテキストボックスに、データベースへのユーザー名とパスワードを入力します。
6. URL テキストボックスに、データベースサーバーに適用することのできるハイライトされた値を置き換える、データベースサーバーの接続文字列を入力します。

```
jdbc:firebirdsql://<host>[:<port>]/<databese path or alias>
```

7. **接続**」をクリックします。

IBM DB2 (ODBC) への接続

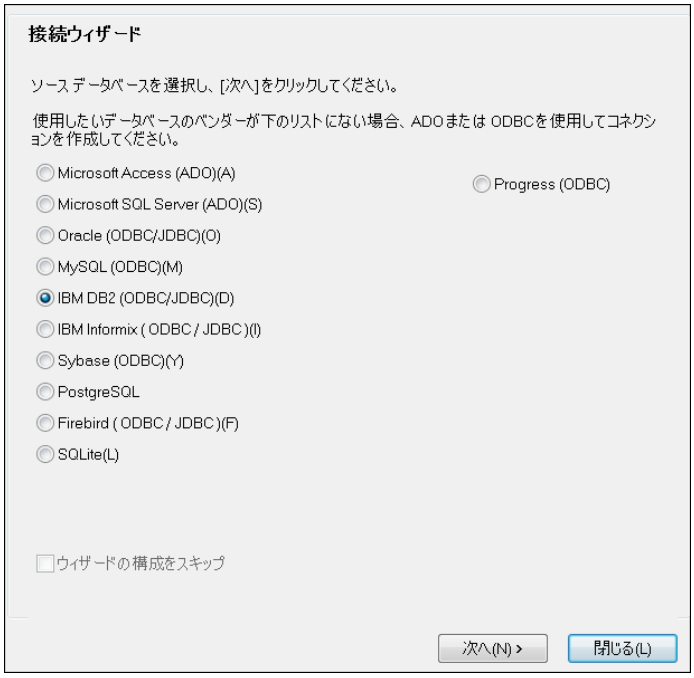
このトピックではODBC を使用した、IBM DB2 への接続のサンプル手順を紹介します。

前提条件：

- IBM Data Server Client は、使用中のオペレーティングシステムにインストールおよび構成される必要があります (このサンプルの場合はIBM Data Server Client 9.7 を使用します)。インストールの手順に関しては、IBM DB2 ソフトウェアにより提供されたドキュメントを確認してください。 IBM Data Server Client インストール後、使用中のコンピューターでODBC ドライバーが使用可能か確認してください (使用可能なODBC ドライバーの表示を参照)。
- データベースエイリアスを作成します。複数の作成方法があります：
  - IBM DB2 構成アシスタントから
  - IBM DB2 コマンドラインプロセッサーから
  - ODBC データソースウィザードから(この場合は、下に表示される手順を参照してください)
- データベース接続の詳細：ホスト、データベース、ユーザー名およびパスワード

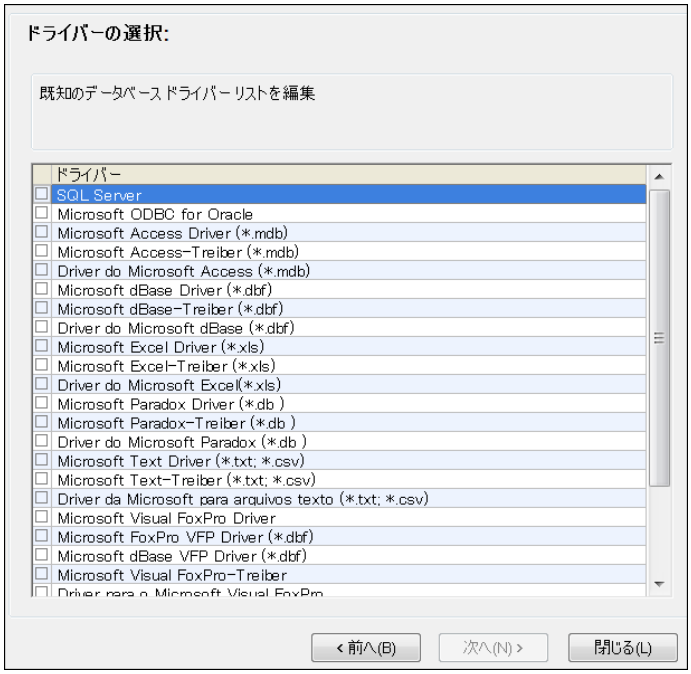
IBM DB2 への接続：

1. データベース接続ウィザードを開始します そして**IBM DB2 (ODBC/JDBC)** を選択します。

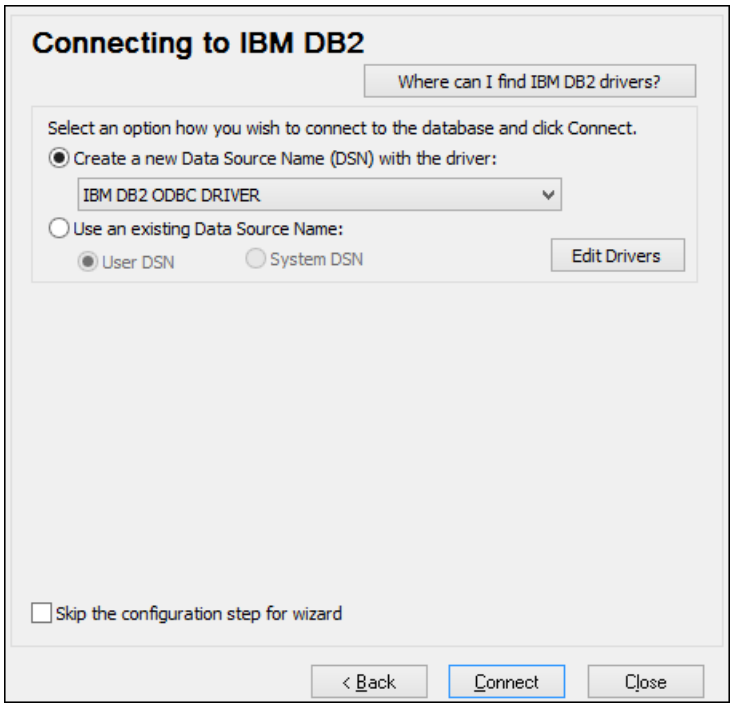


2. **次へ** をクリックします。

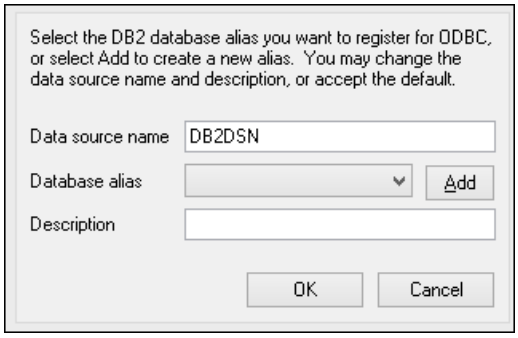
3. **ODBC** を選択して **次へ** をクリックします。データベースのための既知のリストを編集するようにプロンプトされると、IBM DB2 に適用することのできるドライバーを選択して(前提条件を参照)、**次へ** をクリックします。



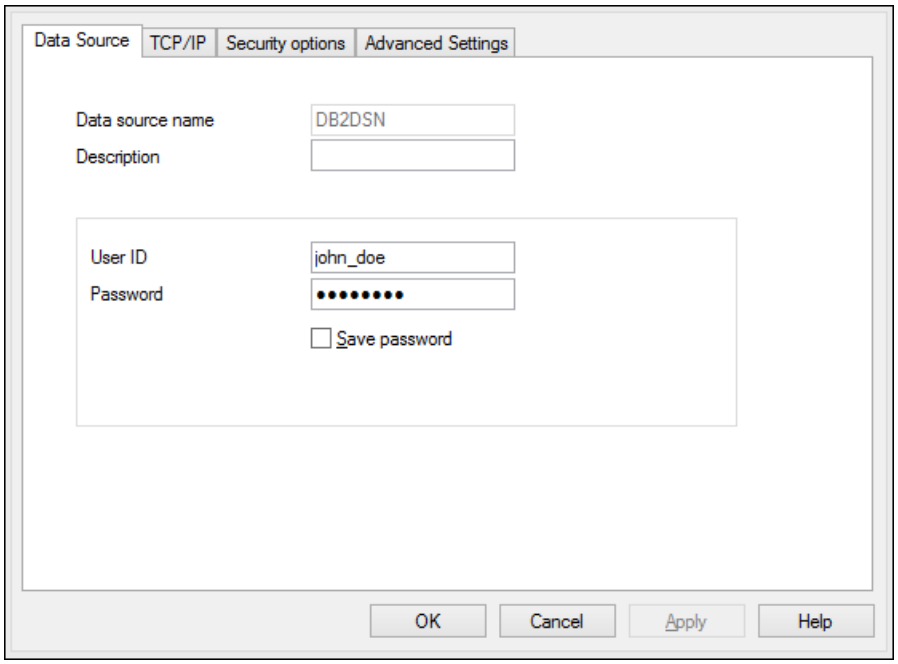
4. リストからIBM DB2 を選択して、 **接続」**をクリックします。 (ドライバーで使用可能なリストを編集するには、**[ドライバーの編集]** をクリックして、使用可能なドライバーのリストを編集します。クリックして追加または削除するIBM DB2 ドライバーをチェックまたはチェックの解除をします。ドライバーを選択または選択の解除をします。)



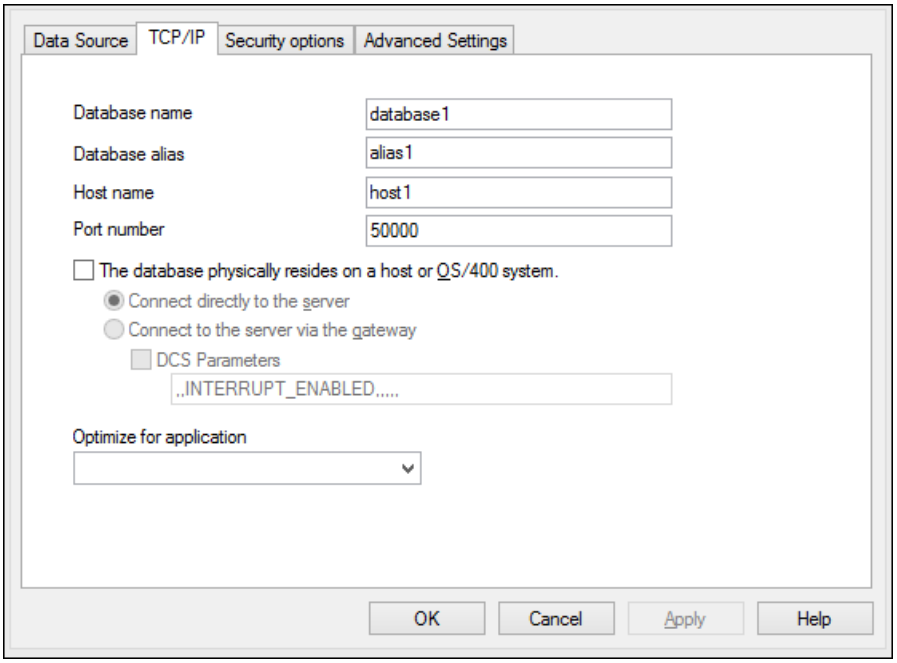
5. ソース名を入力します (このサンプルでは**DB2DSN**)　**追加」**をクリックします。



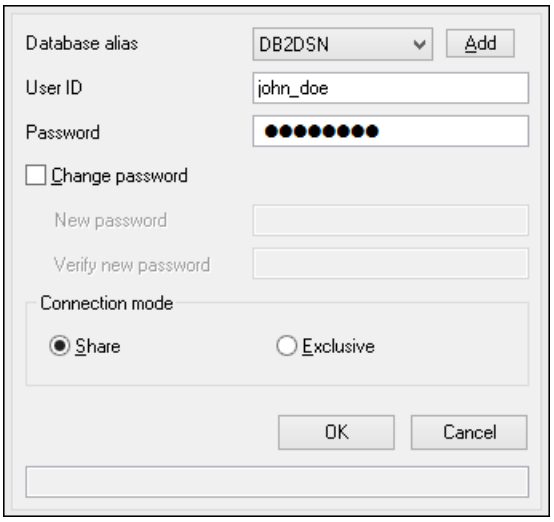
6. **[データソース]** タブで、データベースのユーザー名とパスワードを入力します。



7. **[TCP/IP]** タブで、データベース名、エイリアスの名前、ホスト名とポート名を入力して **「OK」** をクリックします。



8. ユーザ名とパスワードを再入力して「OK」をクリックします。

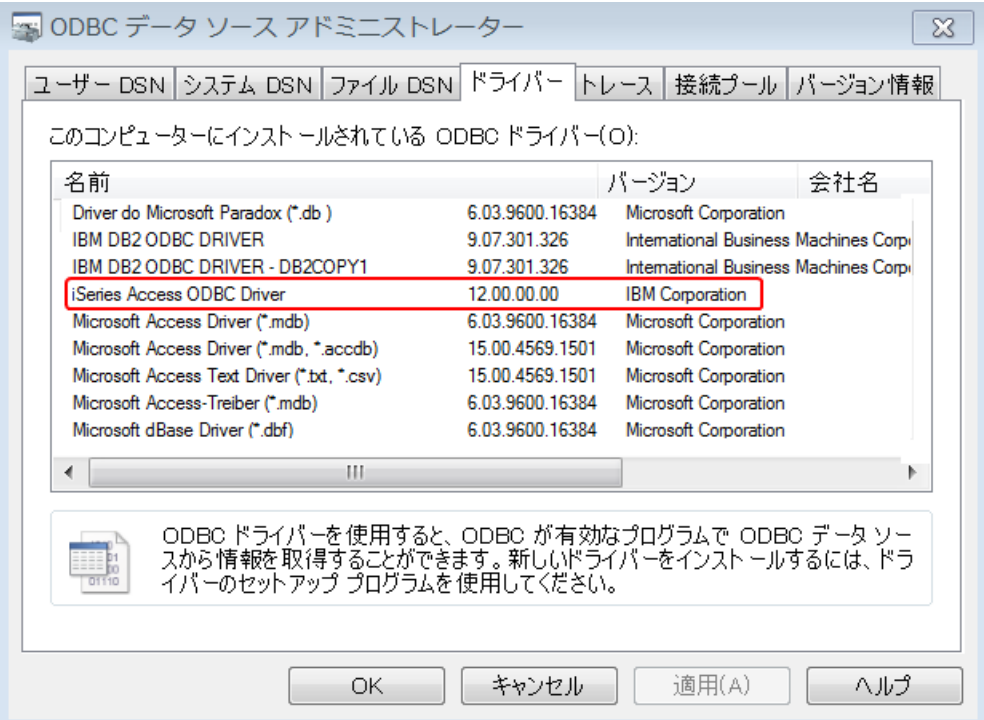


IBM DB2 for i (ODBC) への接続

このトピックは、ODBC を使用して *IBM DB2 for i* データベースへの接続のサンプル手順を説明します。

前提条件：

- *IBM System i Access for Windows* は使用中のオペレーティングシステム（このサンプルでは *IBM System i Access for Windows V6R1M0* を使用しています）にインストールされる必要があります。インストールの手順は、*IBM DB2 for i* ソフトウェアにより提供されているドキュメントを参照してください。インストール後、ODBC ドライバーが使用中のコンピューターで使用可能か確認してください（使用可能なODBC ドライバーの表示を参照）。

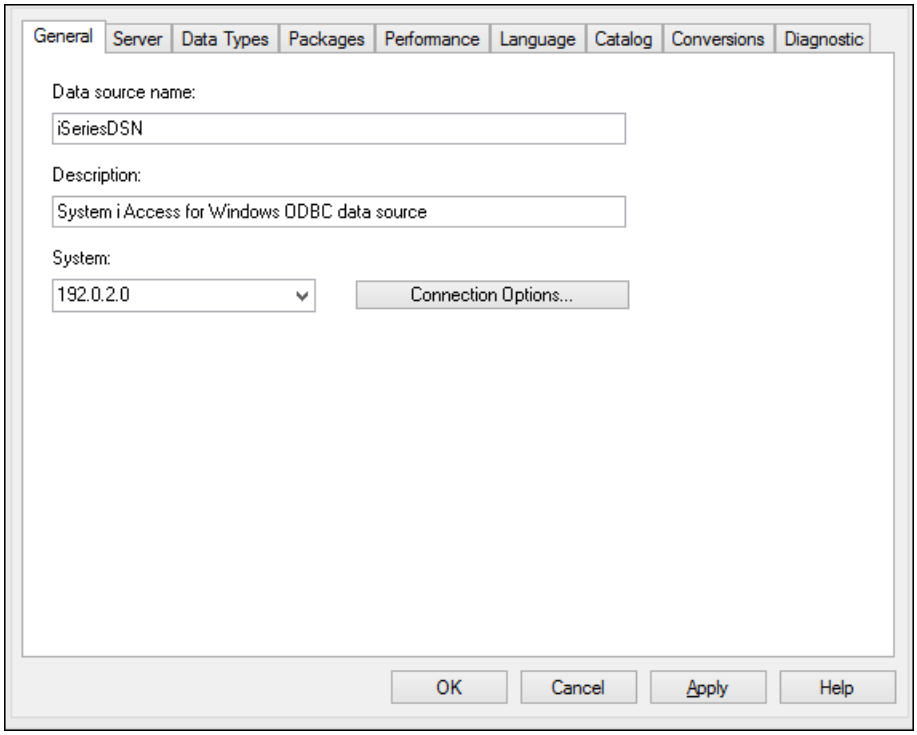


- データベース接続の詳細 データベースサーバーの I.P. アドレス データベースユーザー名、およびパスワード
- *i Navigator* を実行して、新規接続を作成するウィザードの手順を踏んで新規接続を作成してください。システムの指定をプロンプトされると、データベースサーバーのI.P. アドレスを入力してください。接続を作成した後、検証する(接続をクリックして、**ファイル** > **診断** > **接続の検証** を選択) ことが奨励されます。接続エラーが発生した場合、データベースサーバー管理者に連絡してください。

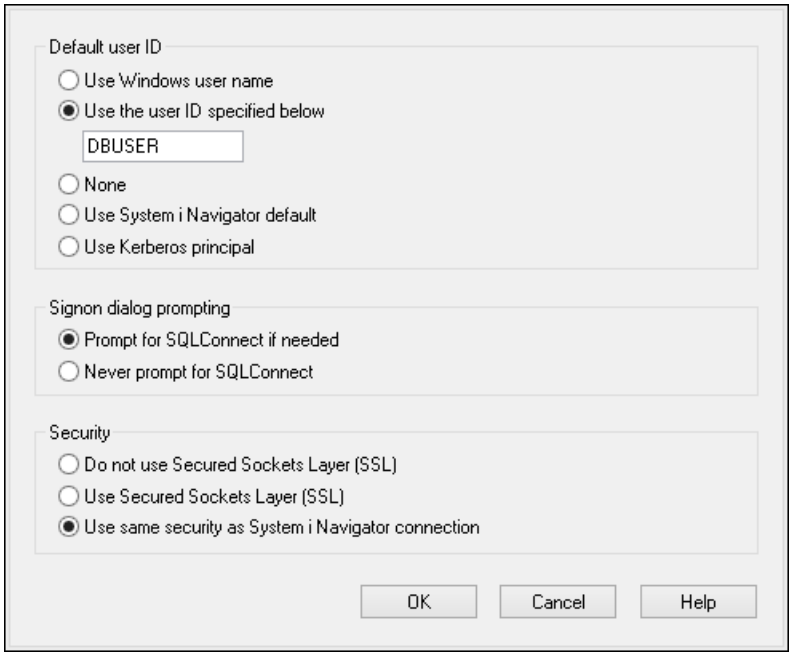
IBM DB2 for i への接続 :

1. データベース接続ウィザードを開始します.
2. **[ODBC 接続]** をクリックします。
3. **[ユーザー DSN]** をクリックします。(または **[システム DSN]** または、後に続く手順が同じ **[ファイル DSN]** をクリックします)。
4. **「追加」** をクリックします。
5. **[iSeries Access ODBC ドライバー]** をリストから選択します。**[ユーザー DSN]** (または、該当する場合 **[システム DSN]**) をクリックします。

6. データソース名を入力して、システムコンボボックスから接続を選択します。このサンプルでは、データソース名は **iSeriesDSN** でシステムは **192.0.2.0** です。



7. **[接続オプション]** をクリックします。**[指定されたユーザー ID を使用]** を選択して、データベースユーザー (このサンプル内では **DBUSER**) の名前を入力します。



8. **「OK」** をクリックします。DSN 内のリストで新しいデータソースを使用することができます。
9. **「接続」** をクリックします。
10. データベースのユーザー名とパスワードをプロンプトされると入力して **「OK」** をクリックします。

IBM Informix (JDBC) への接続

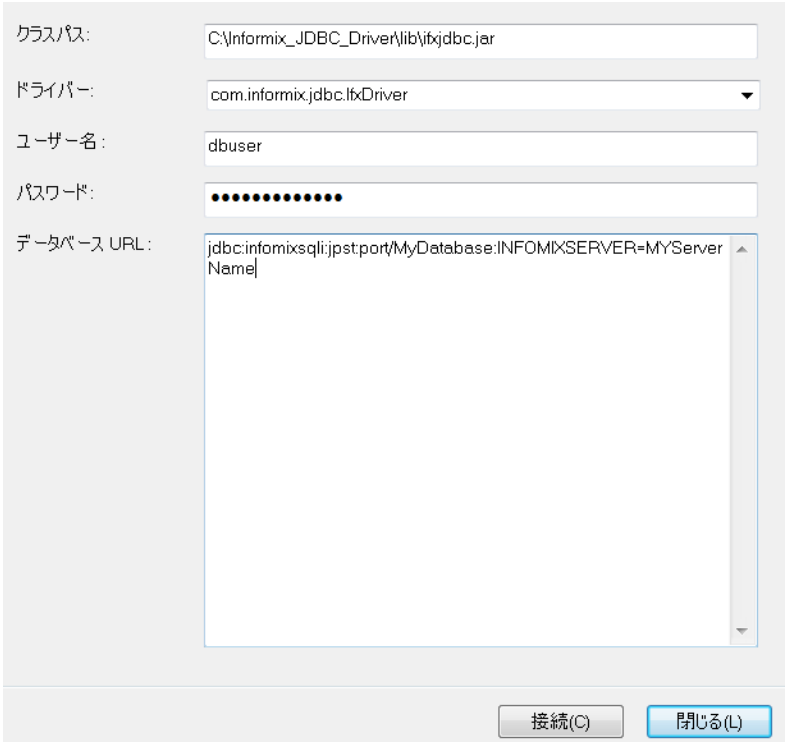
このトピックでは、JDBC を使用して IBM Informix データベースサーバーに接続する手順のサンプルを説明します。

前提条件：

- Java Runtime Environment (JRE) がオペレーティングシステムにインストールされている必要があります。
- (データベースへの接続を与える1つ、または、複数の jar ファイル) JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります。この例では、IBM Informix JDBC ドライバー バージョン 3.70 が使用されています。ドライバーのインストールの説明に関しては、ドライバーに付随するドキュメント、または、「IBM Informix JDBC ドライバプログラマーズガイド」を参照してください)。
- データベース接続の詳細：ホスト、サーバーの名前、データベース、ポート ユーザー名、および パスワード

JDBC を使用して IBM Informix に接続する：

1. データベース接続ウィザードを開始します.
2. 「**JDBC 接続**」をクリックします。.
3. 「クラスパス」の横にデータベースへの接続を与える jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる jar ファイルは次のパスを使用して見つけることができます：**C:\Informix_JDBC_Driver\lib\ifxjdbc.jar**。jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照：CLASSPATH の構成)。
4. 「ドライバー」ボックスから、**com.informix.jdbc.IfxDriver** を選択します。このエントリは、有効な jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます（前のステップを参照してください)。



5. データベースへのユーザー名とパスワードを対応するテキストボックスに入力します。
6. ハイライトされた値を、使用するデータベースサーバーに適用する値と置き換えて、データベースURL テキストボックス内にデータベースサーバーへの接続文字列を入力します。

```
jdbc:informix-sqli://hostName:port/
databaseName:INFORMIXSERVER=myserver;
```

7. **接続** をクリックします。

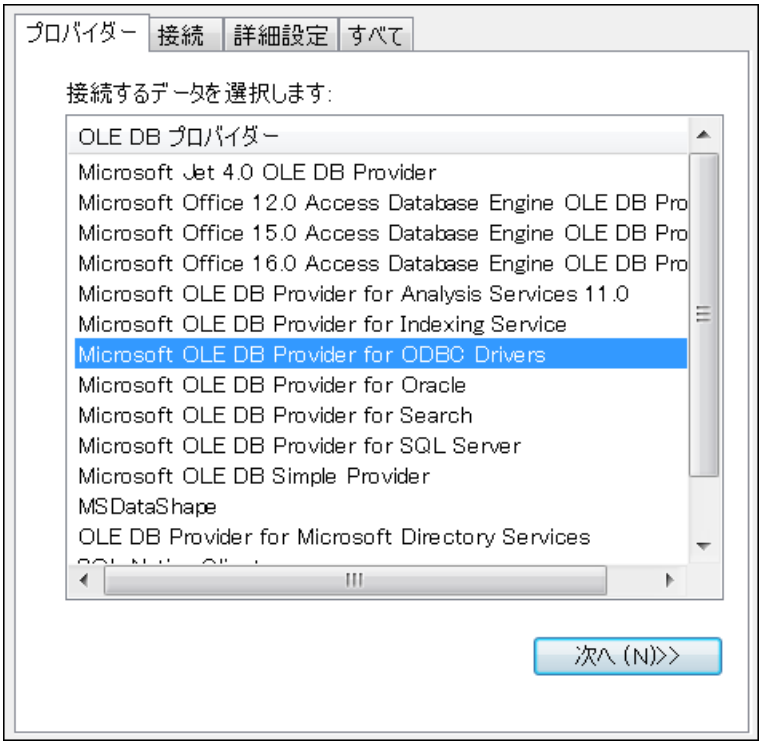
Microsoft Access (ADO) への接続

Microsoft Access データベースに接続するには、既存の Microsoft Access データベースに接続するで説明されているとおり、データベースファイルのためのウィザードとブラウザーの手順を踏みます。別の方法としては、このトピックで説明されているとおり、明確的に ADO 接続をセットアップすることです。このアプローチは、データベースがパスワードで保護されている場合、役に立ちます。

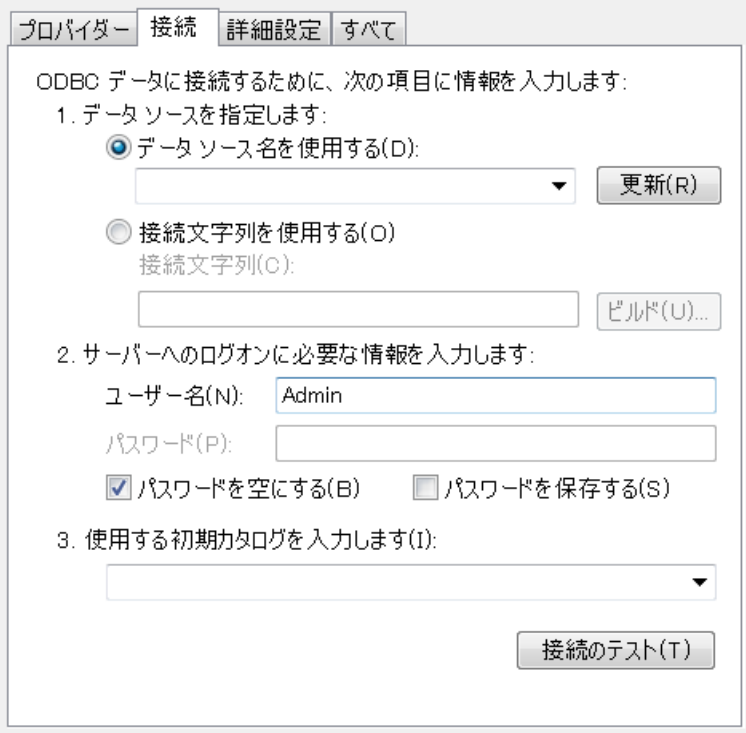
接続を使用して、に接続することもできますが、このシナリオには限度があるため、奨励されません。

パスワードで保護されている Microsoft Access データベースに接続する：

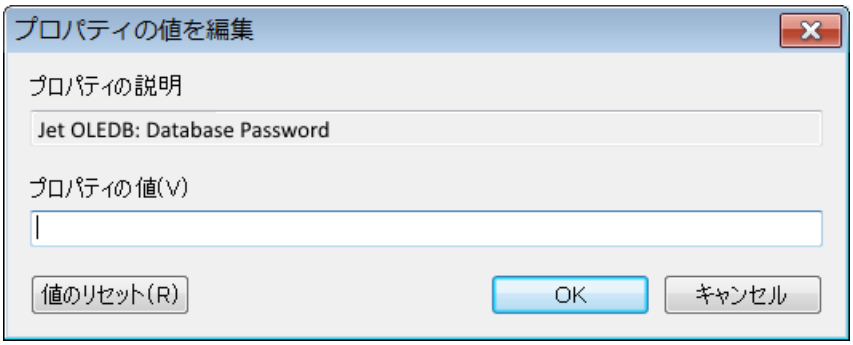
1. データベース接続ウィザードを開始します。
2. **[ADO 接続]** をクリックします。
3. **[構築]** をクリックします。



4. **[Microsoft Office 15.0 Access Database Engine OLE DB Provider]** を選択して、**次へ** をクリックします。



5. *データソースボックス*内で、Microsoft Access ファイルへのパスを入力します。ファイルがローカルネットワーク**U:\Departments\Finance\Reports\Revenue.accdb** を共有するため、ファイルをUNC フォーマットすなわち、**server1** isがサーバーの名前でdfs が共有されるネットワークの名前である、**\\server1\\dfs\Departments\Finance\Reports\Revenue.accdb** に変換します。
6. **全て** タブの**Jet OLEDB:Database Password** プロパティをダブルクリックし、データベースパスワードをプロパティの値として入力します。



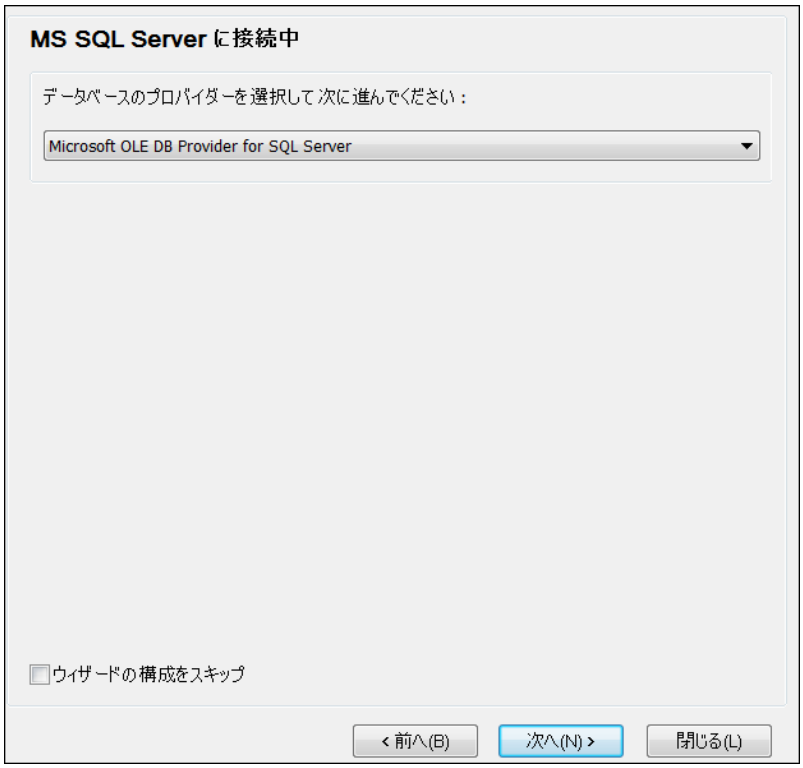
メモ: また接続できない場合、ワーキンググループ情報ファイル（**System.MDW**）を検索して、ユーザープロフィールに適用できるようにします。（手順はhttp://support.microsoft.com/kb/305542 を参照してください。）**Jet OLEDB: System database** プロパティの値を**System.MDW** ファイルのパスに設定します。

Microsoft SQL Server (ADO) への接続

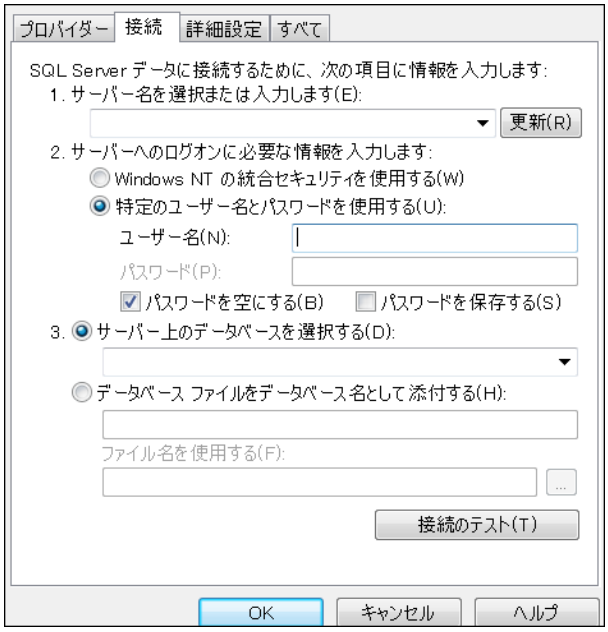
Microsoft OLE DB プロバイダーを使用してのSQL Server への接続 :

1. データベース接続ウィザードを開始します。

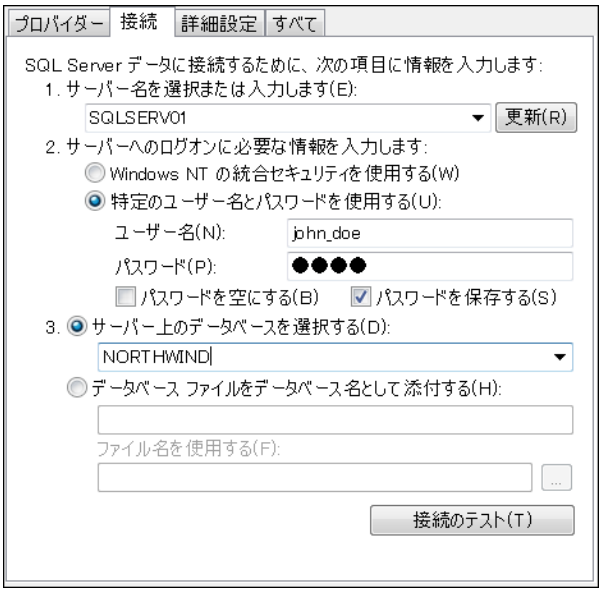
2. **Microsoft SQL Server (ADO)** を選択して、 **「次へ」** をクリックします。使用可能なADO ドライバーが表示されます。



3. **[Microsoft OLE DB Provider for SQL Server]** を選択して、 **「次へ」** をクリックします。



4. データベースサーバーの名前を選択または入力します (このサンプルでは、**SQLSERV01** です)。このネットワーク上の全てのサーバのリストを確認するには、ドロップダウンリストを展開してください。
5. データベースサーバーがWindows ドメイン上の認証されているユーザーの接続を許可している場合、**[Windows NT 統合されたセキュリティの使用]** を選択してください。それ以外の場合は、**[特定の名前とパスワードを使用する]** を選択して、関連するボックスに入力してください。
6. 接続するデータベースを選択します (このサンプルでは、**NORTHWIND** です)。
7. この時点で接続をテストするには、**[接続をテストする]** をクリックします。このステップは任意で奨励されるステップです。
8. 以下を行ってください:
   a. **[パスワードの保存を許可]** チェックボックスを選択します。
   b. **すべて** タブで、**[セキュリティ情報を保持する]** プロパティの値を**True** に変更します。



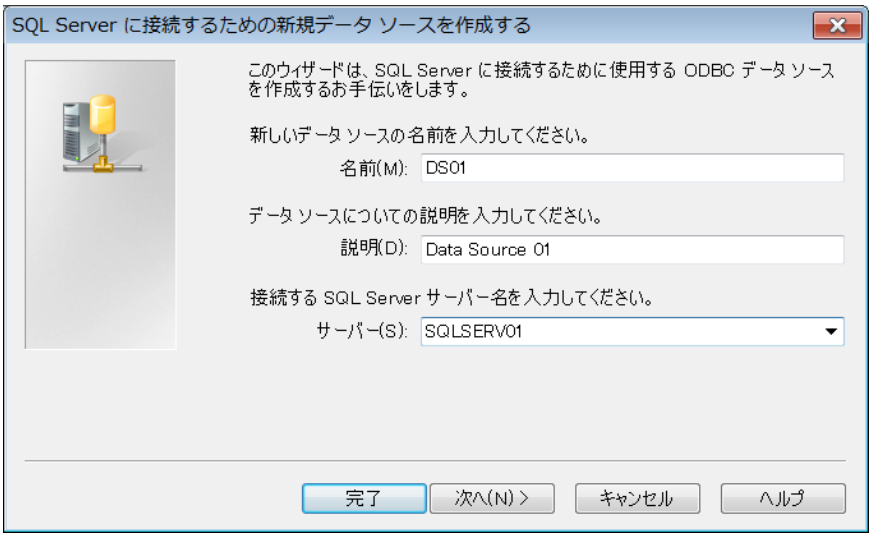
9. **「OK」** をクリックします。

Microsoft SQL Server (ODBC) への接続

ODBC を使用してのSQL Server への接続：

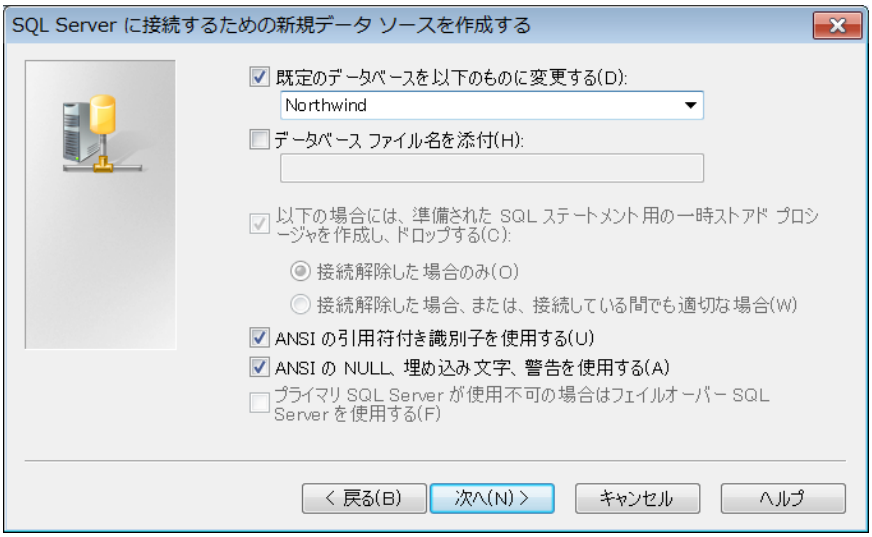
1. データベース接続ウィザードを開始します.
2. **[ODBC 接続]** をクリックします。
3. **[ユーザーDSN]** を選択して、(または管理者の特権を有する場合は、**システムDSN**) **追加」** をクリックします。

4. **[SQL Server]** (または使用可能な場合は**[SQL Server Native Client]**) を選択して、**[ユーザーDSN]** (またはシステムDSN を作成する場合は、**システム[DSN]**) をクリックします。



5. この接続を認識するための名前と詳細を入力し、接続するをリストから選択します。 (このサンプルでは、**SQLSERV01** です。)

6. データベースサーバーがWindows ドメイン上の認証されているユーザーの接続を許可している場合、**[Windows NT 承認]** を選択してください。それ以外の場合は、**[SQL Server 承認 ...]** を選択して、関連するボックスに入力してください。



1. 接続するデータベースを選択します (このサンプルでは、**NORTHWIND** です)。
7. **[完了]** をクリックします。

MySQL (ODBC) への接続

このトピックではODBC ドライバーを使用した、Windows マシンから MySQL データベースへの接続のサンプル手順を紹介します。Windows

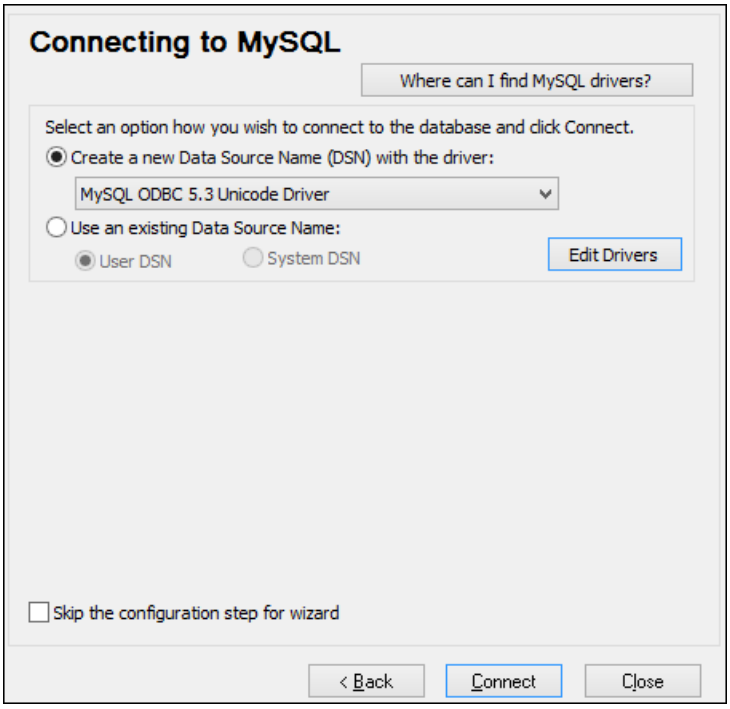
前提条件 :

- MySQL ODBC ドライバーを使用中のオペレーティングシステムにインストールする必要があります。 (インストールに関しては、ドライバーに付随するドキュメントを参照してください)。
- データベース接続の詳細 : ホスト、データベース、ユーザー名およびパスワード

ODBC を使用してのMySQL への接続 :

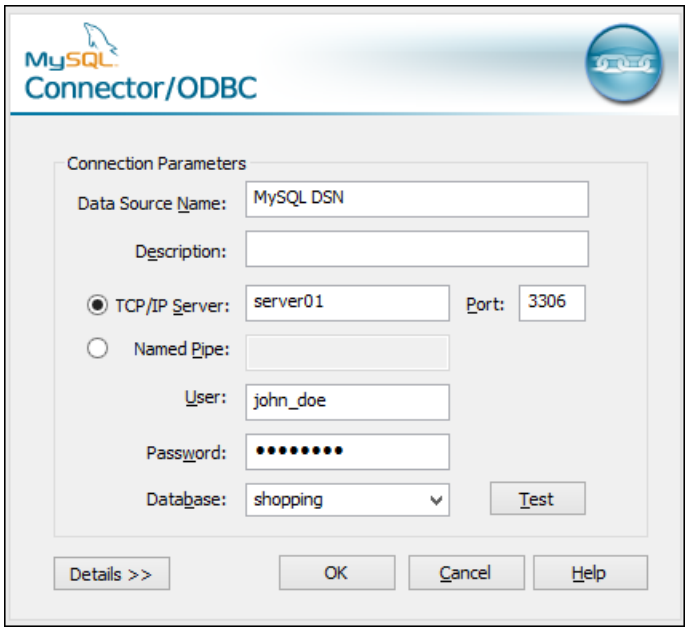
1. データベース接続ウィザードを開始します。

2. **[MySQL (ODBC)]** を選択して、**「次へ」** をクリックします。



3. **[ドライバーを使用して新しいデータソース名 (DSN) を作成]** を選択して、MySQL ドライバーを選択します。MySQL ドライバーがリストにない場合は、**[ドライバーを編集]** をクリックして、使用可能な MySQL ドライバーを選択します (使用中のオペレーティングシステムにインストールされているすべての ODBC ドライバーがリストされています)。

4. **接続**」をクリックします。



5. データソース名ボックスの中に、今後このODBC データソースを認識するために役に立つ詳細を入力します。
6. データベース接続資格（TCP/IP サーバー、ユーザー、パスワード）を入力して、データベースを選択し、「**OK**」をクリックします。

メモ: Ifデータベースサーバーがリモートの場合、サーバー管理者が使用中のコンピューターのIP アドレスからのリモート接続を承諾するように構成される必要があります。また、**[詳細 >>]** をクリックすると使用することのできる追加パラメーターが表示されます。デフォルトの値を変更する前に、ドライバーのドキュメントを確認してください。

Oracle (ODBC) への接続

このサンプルは、ローカルオペレーティングシステム上にインストールされているOracle データベースクライアントを介して、ネットワークマシン上でXMLSpy からOracle データベースサーバーへの接続のよく使用されている状況を説明します。

このサンプルでは、XMLSpy 内のデータベース接続ウィザードを使用して、ODBC データソース（DSN）のセットアップについて説明されています。既にDSN が作成されている場合は Windows 内のODBC データソース管理者から直接作成をする事も可能です。ウィザードがプロンプトした場合はそのように選択してください。データソースに関する詳細は ODBC 接続のセットアップを参照してください。

前提条件：

- （ODBC Oracle ドライバーを含む）Oracle データベースクライアントが使用中のオペレーティングシステムでインストール、および構成されていなければなりません。 Oracle データベースクライアントのインストールおよび構成方法はOracle ソフトウェアに付随するドキュメントを参照してください。
- Oracle ホームディレクトリにある**tnsnames.ora** ファイルは、データベース接続パラメーターを紹介するエントリを含みます。フォーマットは以下に類似します：

```
ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = server01)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SID = orcl)
```

```
      (SERVER = DEDICATED)
    )
  )
```

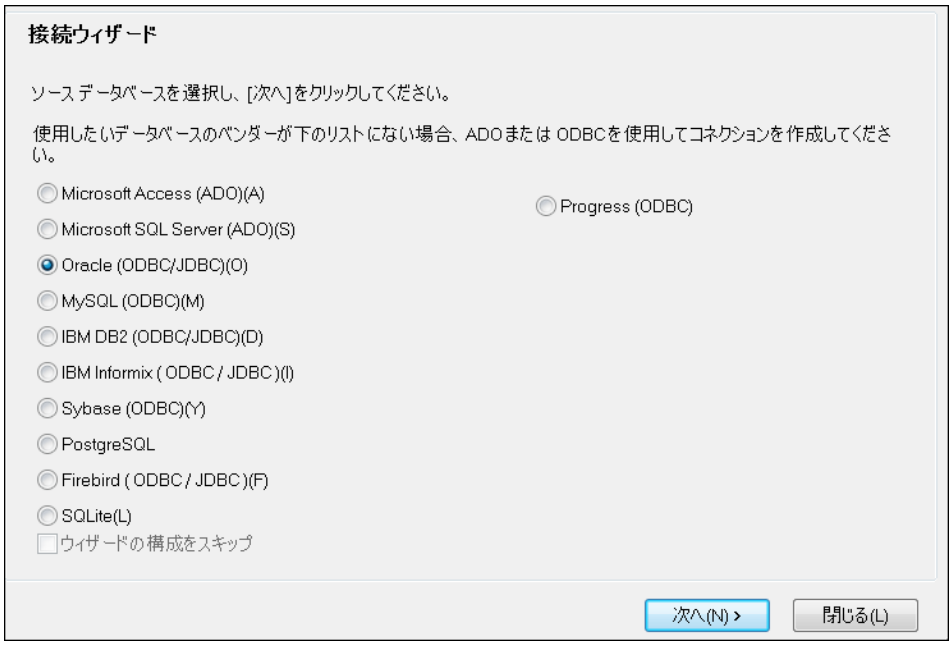
tnsnames.ora ファイルのパスはOracle ホームディレクトリがインストールされている場所によります。Oracle データベースクライアント 11.2.0 に関しては、デフォルトOracle ホームディレクトリのパスは以下の通りです：

```
C:\app\username\product\11.2.0\client_1\network\admin\tnsnames.ora
```

接続の詳細を貼り付け保存する、または (使用可能であれば) Oracle *ネット構成アシスタントウィザード* を実行することにより **tnsnames.ora** ファイルに新規エントリを追加することができます。

ODBC を使用してのOracle への接続：

1. データベース接続ウィザードを開始します。

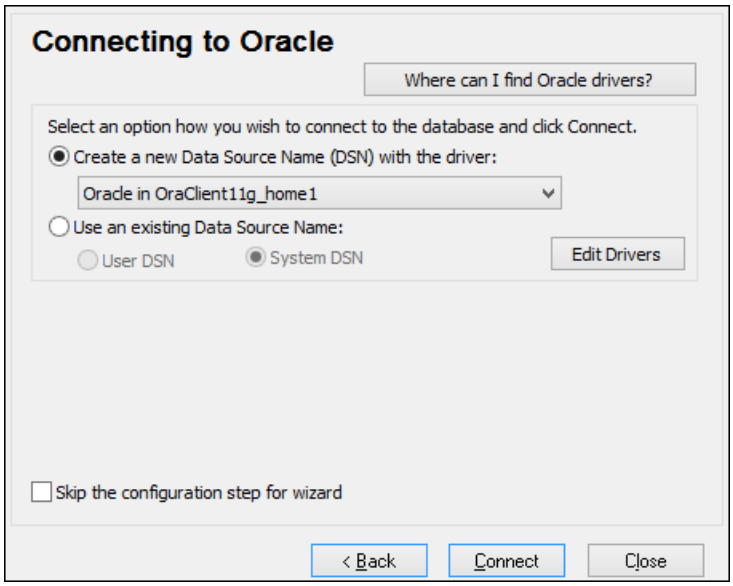


2. **[Oracle (ODBC / JDBC)]** を選択して、**「次へ」** をクリックします。

3. **[ODBC]** を選択します。

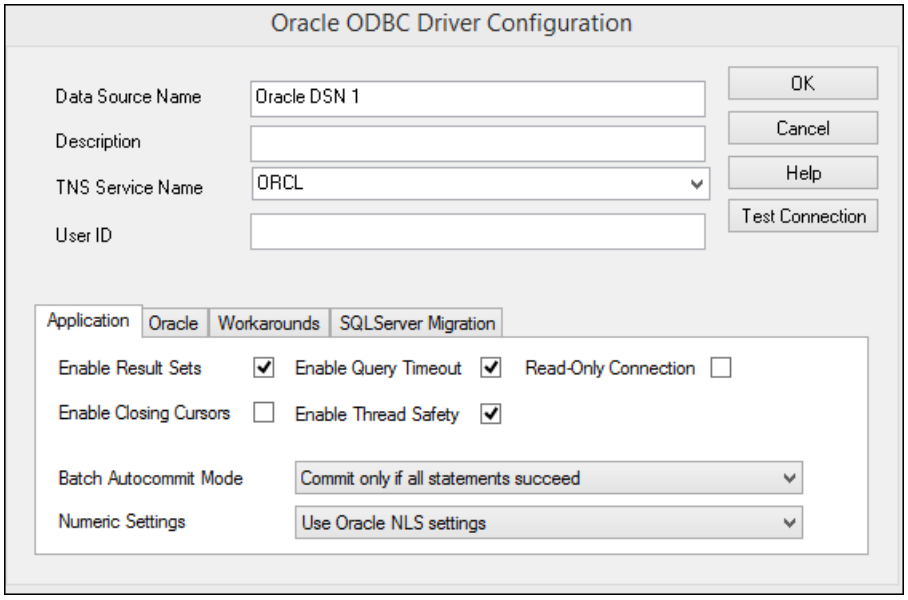
4. **[ドライバーの編集]** をクリックします。

5. 使用するOracle ドライバーをせんたくします（このサンプルでは**Oracle in OraClient11g_home1** です）。Oracle クライアントをインストール後システムで使用可能なOracle ドライバーのリストを表示します。
6. **[戻る]** をクリックします。
7. **[新しいデータソース (DSN) をドライバーを使用して作成]** を選択して、ステップ 4 で選択されたOracle ドライバーを選択します。

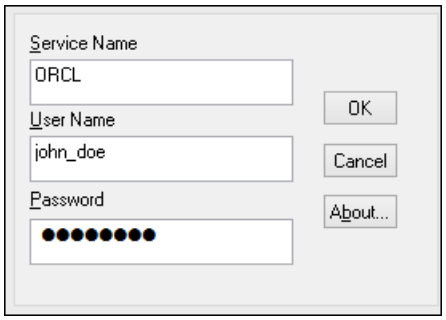


Microsoft ODBC for Oracle という名称のMicrosoft により提供されるドライバーを使用することは避けてください。Microsoft はOracle により提供されるODBC ドライバーを使用することを奨励しています。（以下参照 :http://msdn.microsoft.com/en-us/library/ms714756%28v=vs.85%29.aspx）

8. **「接続」** をクリックします。



9. データソース名テキストボックスにデータソースを識別する名前を入力します（このサンプルの場合は**Oracle DSN 1** です）。
10. TNS サービス名ボックスでは、**tnsnames.ora** ファイルで定義されている通りの接続名を入力します。[前提条件](#)を参照してください）。このサンプルでは接続名 **ORCL** はです。
11. 「**OK**」をクリックします。



12. データベースのユーザー名とパスワードを入力して、「**OK**」をクリックします。

Oracle (JDBC) への接続

JDBC インターフェイスを使用して、クライアントマシンから、Oracle データベースサーバーへの接続の方法を説明します。接続は、Oracle Web サイトから使用することのできる、**Oracle Instant Client Package (Basic)** を使用する、Java 接続として作成されます。接続の型の利点は、Oracle Instant Client Package により与えられる Java 環境と jar ライブラリのみを必要とし、更に複雑なデータベースクライアントをインストールする手間を省くことができる必点です。

前提条件：

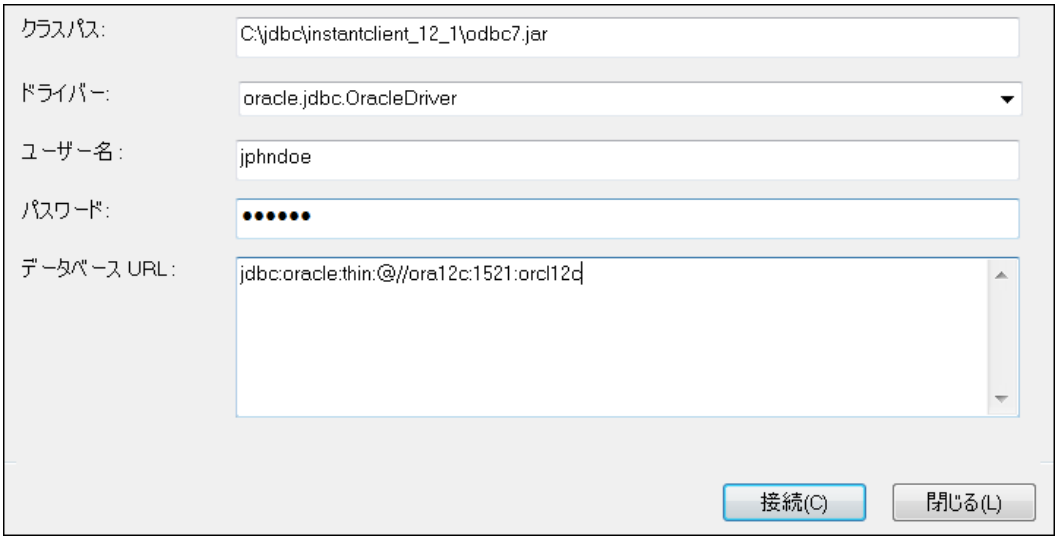
- Java Runtime Environment (JRE) または Java Development Kit (JDK) が使用中のオペレーティングシステムにインストールされている必要があります。
- オペレーティングシステムの `PATH` 環境変数は、JRE または JDK インストールの `bin` ディレクトリへのパスを含んでいる必要があります。例、`C:\Program Files (x86)\Java\jre1.8.0_51\bin`。
- **Oracle Instant Client Package (Basic)** が使用中のオペレーティングシステムで使用可能である必要があります。パッケージは、Oracle の公式 Web サイトからダウンロードすることができます。このサンプルは、

Windows 32-ビットのためのOracle Instant Client Package バージョン 12.1.0.2.0を使用します。

- データベース接続の詳細があります：ホスト、データベースパス、またはエイリアス、ユーザー名、およびパスワード。

クライアントパッケージを使用してOracle に接続する：

1. データベース接続ウィザードを開始します。
2. **JDBC Connections**をクリックします。.
3. 「クラスパス」の横にデータベースへの接続を与える jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる jar ファイルは次のパスを使用して見つけることができます：**C:\jdbc\instantclient_12_1\odbc7.jar**。 jar ファイルパスをオペレーティングシステムのCLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照：CLASSPATH の構成)。
4. 「ドライバー」ボックスから、**oracle.jdbc.OracleDriver** または**oracle.jdbc.driver.OracleDriver** を選択します。このエントリは、有効な jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムのCLASSPATH 環境変数内で検出された場合のみ使用することができます（前のステップを参照してください）。
5. データベースへのユーザー名とパスワードを対応するテキストボックスに入力します。



6. ハイライトされた値を、使用するデータベースサーバーに適用する値と置き換えて、データベースURL テキストボックス内にデータベースサーバーへの接続文字列を入力します。

```
jdbc:oracle:thin:@//host:port:service
```

7. **「接続」**をクリックします。

PostgreSQL (ODBC) への接続

このトピックではODBC ドライバーを使用した、Windows マシンから PostgreSQL データベースへの接続のサンプル手順を紹介します。Windows からは、 PostgreSQL ODBC ドライバーを入手することができませんので、個別にダウンロードおよびインストールする必要があります。このサンプルでは、公式 Web サイトからダウンロードされたpsqlODBC ドライバー (バージョン 09_03_300-1) を使用しています。 データベースドライバー概要も参照してください）。

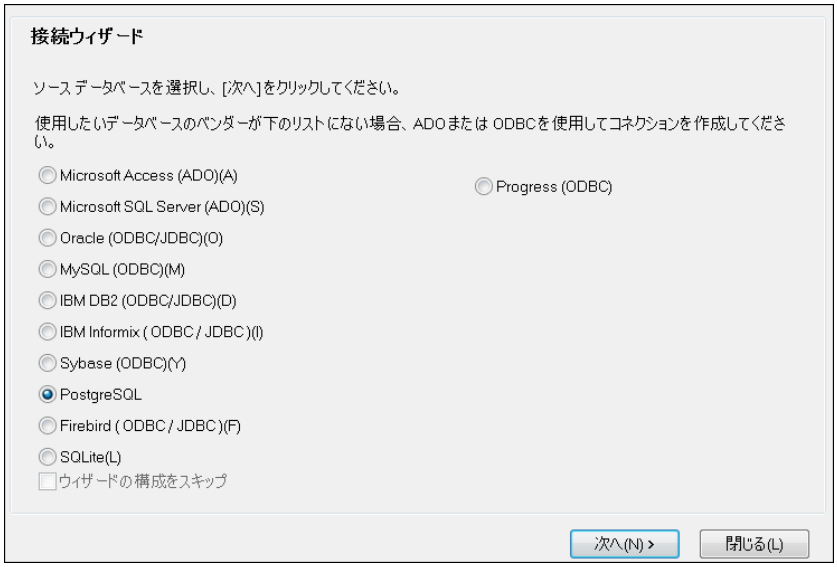
メモ： (ODBC ドライバー無しで)直接 PostgreSQL データベースサーバーに接続するには、次を参照してください：PostgreSQL 接続のセットアップ.

前提条件：

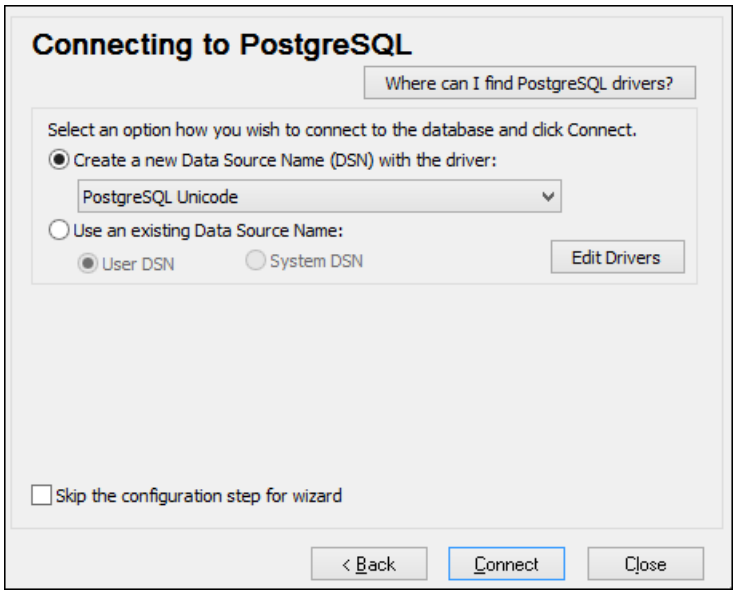
- *psqlODBC* ドライバーを使用中のオペレーティングシステムにインストールする必要があります。 (インストールに関しては、ドライバーに付随するドキュメントを参照してください)。
- データベース接続の詳細 : ホスト、データベース、ユーザー名およびパスワード

ODBC を使用しての PostgreSQL への接続 :

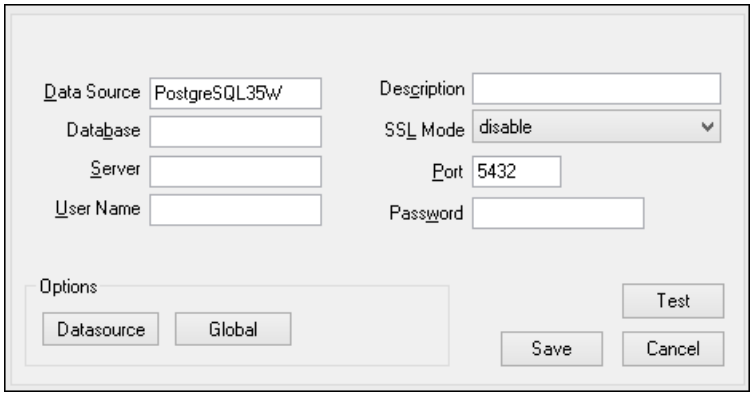
1. データベース接続ウィザードを開始します。



2. **PostgreSQL (ODBC)** を選択して、 **「次へ」** をクリックします。



3. **[ドライバーを使用して新しいデータソース名 (DSN) を作成]** を選択して、 PostgreSQL ドライバーを選択します。 MySQL ドライバーがリストにない場合は、 **[ドライバーを編集]** をクリックして、使用可能な PostgreSQL ドライバーを選択します (使用中のオペレーティングシステムにインストールされているすべての ODBC ドライバーがリストされています)。
4. **「接続」** をクリックします。



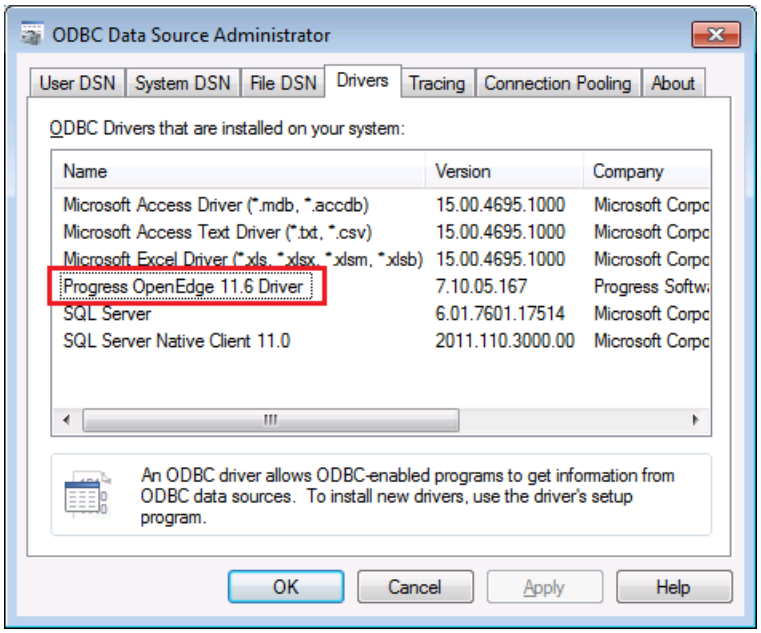
5. データベース接続資格 (データベース サーバー、ポート、ユーザー名、パスワード)を入力して、 「OK」をクリックします。

Progress OpenEdge (ODBC) への接続

このトピックは、Progress OpenEdge 11.6 ODBC ドライバーを使用して、Progress OpenEdge データベースサーバーへの接続の簡単な手順について説明しています。

前提条件

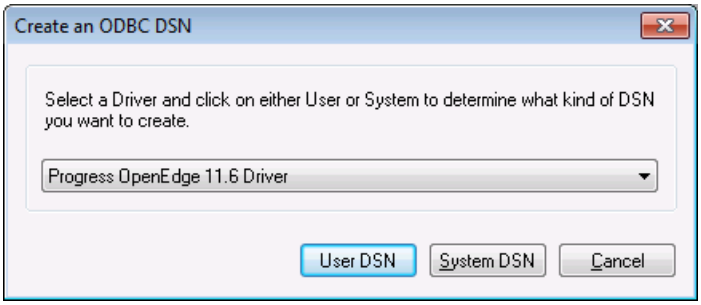
- *Progress OpenEdge* ドライバーのための *ODBC* コネクタが使用中のオペレーティングシステムにインストールされている必要があります。Progress OpenEdge ODBC ドライバーは、ベンダーのWeb サイトからダウンロードすることができます (次も参照 :データベース ドライバーの概要)。XMLSpy の 32 ビットバージョンを作動する場合は、 32 ビット ドライバーをダウンロードし、 64 ビットバージョンを作動する場合は、64 ビット ドライバーをダウンロードしてください。インストールの後、ODBC ドライバーが使用中のマシンで使用することができるかを確認してください (次も参照 :使用することのできるODBC ドライバーを確認する)。



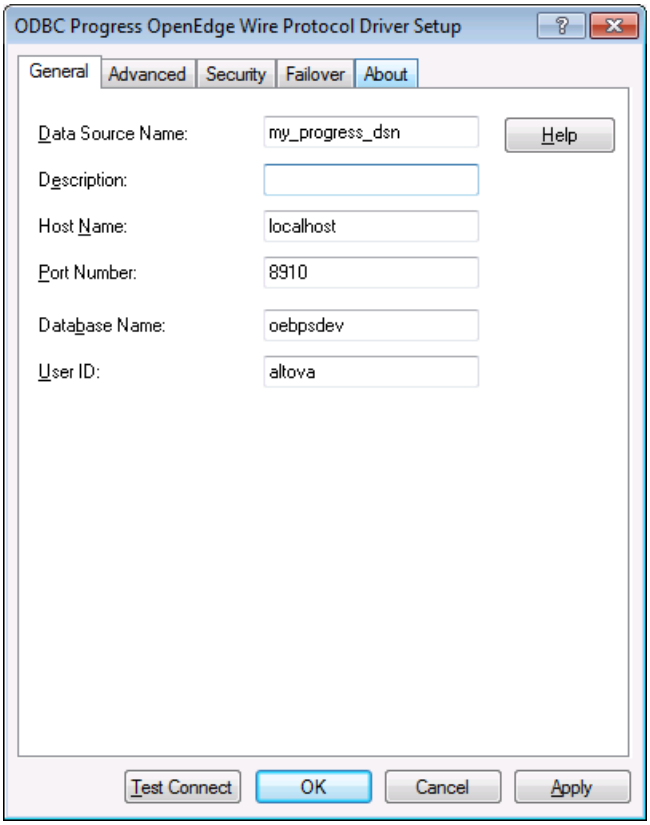
- データベース接続の詳細があります :ホスト名、ポート番号、データベース名、ユーザーID、および パスワード。

ODBC を使用して Progress OpenEdge に接続する

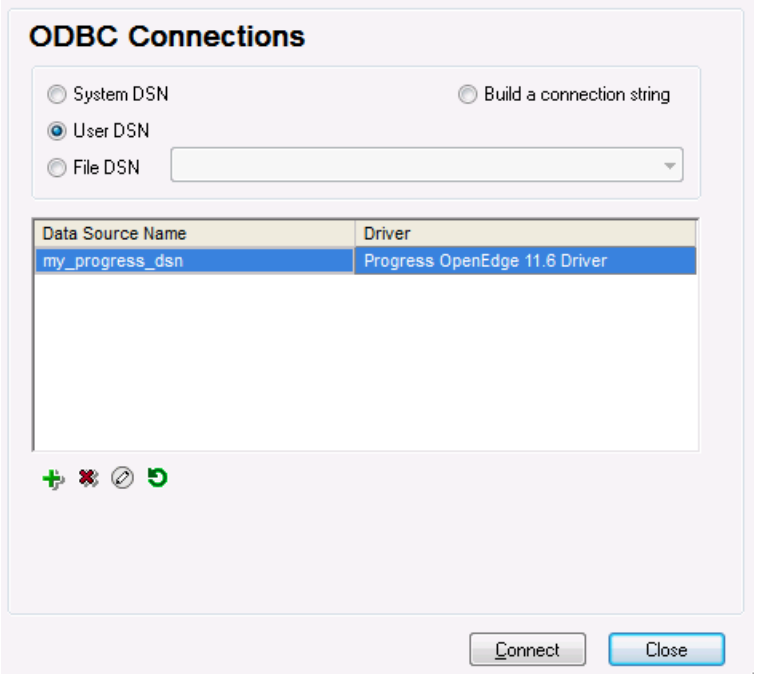
1. データベース接続ウィザードを開始します.
2. **ODBC Connections**をクリックします。
3. **「ユーザーDSN」**をクリックします。(または、次の手順が類似している**「システムDSN」**、または、**「ファイルDSN」**をクリックします)。
4. **「追加」**をクリックします。
5. **「Progress OpenEdge ドライバー」**をリストから選択し、**「ユーザーDSN」**(または、適用できる場合は、**「システムDSN」**)をクリックします。



6. データベース接続の資格情報(データベース、サーバー、ポート、ユーザー名、パスワード)を入力し、**「OK」**をクリックします。入力されたデータを保存する前に、接続を検証するには、**「接続をテストする」**をクリックします。



7. **「OK」**をクリックします。新規データソースがODBC データソースのリストに表示されます。



8. **接続**をクリックします。

Progress OpenEdge (JDBC) への接続

JDBC を使用して Progress OpenEdge 11.6 データベースサーバーへの接続するためのサンプルの手順が説明されています。

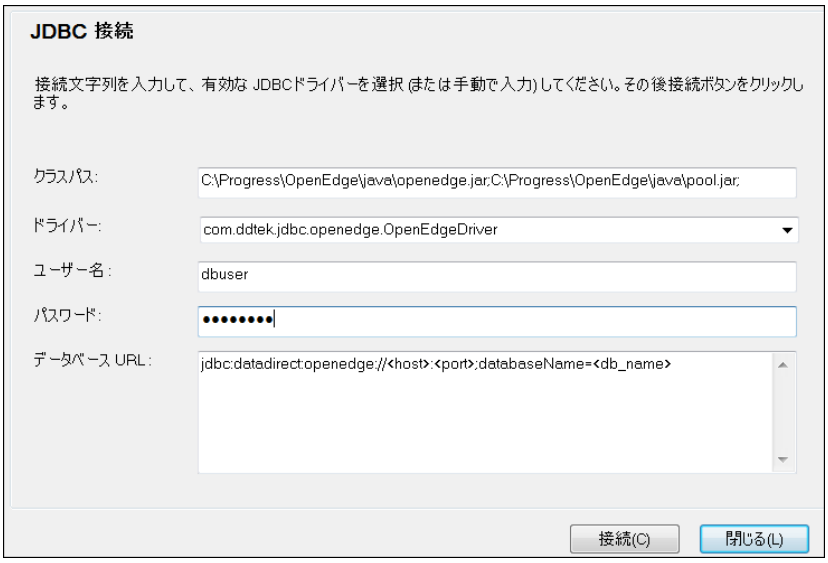
前提条件

- Java Runtime Environment (JRE) または Java Development Kit (JDK) が使用中のオペレーティングシステムにインストールされている必要があります。 XMLSpy (32-ビット、 64-ビット) のプラットフォームが、JRE/JDK に一致していることを確認してください。
- オペレーティングシステムの `PATH` 環境変数は、JRE または JDK インストールの `bin` ディレクトリへのパスを含んでいる必要があります。例、`C:\Program Files (x86)\Java\jre1.8.0_51\bin`.
- Progress OpenEdge JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります。このサンプルでは、JDBC 接続は、**C:\Progress\OpenEdge\java** に OpenEdge SDK のインストールの一部として存在する**openedge.jar** と**pool.jar** ドライバーコンポーネントファイルにより与えられています。 .
- データベース接続の詳細があります :ホスト、データベースパス、またはエイリアス、ユーザー名、およびパスワード。

JDBC を使用して OpenEdge へ接続する

1. データベース接続ウィザードを開始します.
2. 『**JDBC 接続**』をクリックします。.
3. 『クラスパス』の横にデータベースへの接続を与える jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる jar ファイルは次のパスを使用して見つけることができます`C:\Progress\OpenEdge\java\openedge.jar;C:\Progress\OpenEdge\java\pool.jar;`。 jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、『クラスパス』テキストボックスを空白のままにすることができます。(次も参照 :CLASSPATH の構成)。

4. 「ドライバー」ボックスから、**com.ddtek.jdbc.openedge.OpenEdgeDriver** を選択します。このエントリは、有効な jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます（前のステップを参照してください）。



5. データベースへのユーザー名とパスワードを対応するテキストボックスに入力します。
6. ハイライトされた値を、使用するデータベースサーバーに適用する値と置き換えて、データベースURL テキストボックス内にデータベースサーバーへの接続文字列を入力します。

```
jdbc:datadirect:openedge://host:port;databaseName=db_name
```

7. **「接続」**をクリックします。

Sybase (JDBC) への接続

このトピックでは、JDBC を使用して Sybase データベースサーバーに接続する手順のサンプルを説明します。

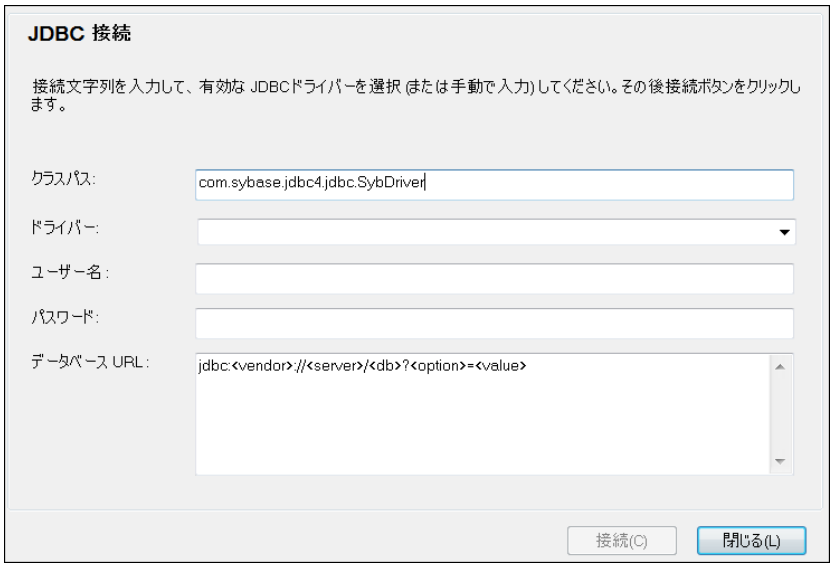
前提条件：

- Java Runtime Environment (JRE) がオペレーティングシステムにインストールされている必要があります。
- Sybase *jConnect* ドライバーがオペレーティングシステムにインストールされている必要があります（このサンプルでは、*Sybase Adaptive Server Enterprise PC Client* インストールの一部としてインストールされている *jConnect 7.0* が、使用されています）。ドライバーのインストールの手順に関しては、ドライバーに付随するドキュメント、または Sybase のドキュメントを参照してください）。
- データベース接続の詳細：ホスト、サーバーの名前、データベース、ポート、ユーザー名、およびパスワード

JDBC を使用して Sybase へ接続する：

1. データベース接続ウィザードを開始します。
2. **「JDBC 接続」**をクリックします。
3. 「クラスパス」の横にデータベースへの接続を与える jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる jar ファイルは次のパスを使用して見つけることができます：**C:\sybase\jConnect-7_0\classes\jconn4.jar**。jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照：CLASSPATH の構成)。
4. 「ドライバー」ボックスから、を選択します。「ドライバー」ボックスから、**com.sybase.jdbc4.jdbc.SybDriver** を選択します。このエントリは、有効な jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます（前のステップを参照してく

ださい)。



5. データベースのユーザー名とパスワードを対応するテキストボックスに入力します。
6. ハイライトされた値をデータベースサーバーに適用することのできる値と置き換えて、データベースURL テキストボックス内のデータベースサーバーの接続文字列を入力します。

```
jdbc:sybase:Tds:hostName:port/databaseName
```

7. **「接続」**をクリックします。

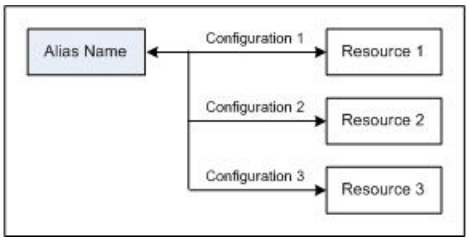
15.2 サポートされているデータベース

次のデータベースはサポートされています。各データベースの使用できるルート オブジェクトもリストされています。Altova は他のODBC/ADO データベースへのサポートの向上のための努力をしていますが、以下にリストされたデータベースのみで接続とデータ処理のテストに成功しています。使用中のAltova アプリケーションが、 64 ビット バージョンの場合、接続されているデータベース特定の 64 ビットデータベース ドライバーにアクセスしてください。Altova アプリケーションはIBM iSeries データベースのロジカル ファイルをサポートし、ロジカル ファイルを表示します。

データベース	ルートオブジェクト	メモ
Firebird 2.5.4	データベース	
IBM DB2 8.x, 9.1, 9.5, 9.7, 10.1, 10.5	スキーマ	
IBM DB2 for i 6.1, 7.1	スキーマ	ロジカルファイルはサポートされ、ビューとして表示されています。
IBM Informix 11.70	データベース	
Microsoft Access 2003, 2007, 2010, 2013	データベース	
Microsoft Azure SQL Database	データベース	SQL Server 2016 codebase
Microsoft SQL Server 2005, 2008, 2012, 2014, 2016	データベース	
MySQL 5.0, 5.1, 5.5, 5.6	データベース	
Oracle 9i, 10g, 11g, 12c	スキーマ	
PostgreSQL 8.0, 8.1, 8.2, 8.3, 9.0.10, 9.1.6, 9.2.1, 9.4, 9.5	データベース	PostgreSQL 接続は、ODBC またはJDBC などのインターフェイス (ドライバ) によりネイティブとドライバーベースの接続でサポートされています。ネイティブな接続はドライバを必要としません。
Progress OpenEdge 11.6	データベース	
SQLite 3.x	データベース	SQLite 接続は、ネイティブとしてサポートされます。SQLite データベースファイルへの直接の接続です。別途のドライバーは必要ありません。 Authentic ビューでは、SQLite からのデータを編集することはできません。 Authentic ビューで、SQLite からのデータを保存する場合は、既知の制限についてのメッセージが通知されます。
Sybase ASE15	データベース	

16 Altova グローバルリソース

Altova グローバルリソースはファイル、フォルダー、そしてデータベースリソースのエイリアス集合のことです。各エイリアスは複数の構成を持ち、各構成を1つのリソースにマッピングすることができます (下のスクリーンショット参照)。そのため、グローバルリソースを入力として使用することで、グローバルリソースを構成とともに切り替えることができます。この操作は、アクティブな構成を選択することのできるGUI 上で簡単に行うことができます。例えば、XML ドキュメントを変換するXSLT スタイルシートがグローバルリソース (エイリアス) により割り当てられている場合、このグローバルリソースに対して複数の構成を作成し、各構成が異なるXSLT ファイルを指すようにすることができます。このようにグローバルリソースを設定すると、構成を切り替えることで変換に使用されるXSLT ファイルを変更することができます。



グローバルリソースはAltova 製品内部におけるリソースの切り替えだけでなく、他のAltova 製品から生成、使用されるリソースに対しても使用することができます。そのため、1つのAltova 製品から他のAltova 製品で使用されるファイルを一括して生成することができます。このような機能により、開発ならびにテストスピードが飛躍的に向上します。例えば、XMLSpy にあるXSLT スタイルシートを使って、Altova MapForce で生成されたファイルを変換することができます。

Altova グローバルリソースの使用には、2つのステップがあります:

- グローバルリソースの定義: XML ファイルによりリソースが定義されます。これらのリソースは複数のAltova 製品間で共有することができます。
- グローバルリソースの使用: XMLSpy 内部ではファイルパスの代わりにグローバルリソースによりファイルを指定することができます。グローバルリソースを使用すると、XMLSpy にてアクティブな構成を切り替えることで、使用するリソースを簡単に変更することができるようになります。

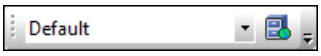
その他の Altova 製品におけるグローバルリソース

現在のところ、以下のAltova 製品でグローバルリソースを使用することができます:XMLSpy、StyleVision、MapForce、Authentic Desktop、MobileTogether Designer およびDatabaseSpy。

16.1 グローバル リソースの定義

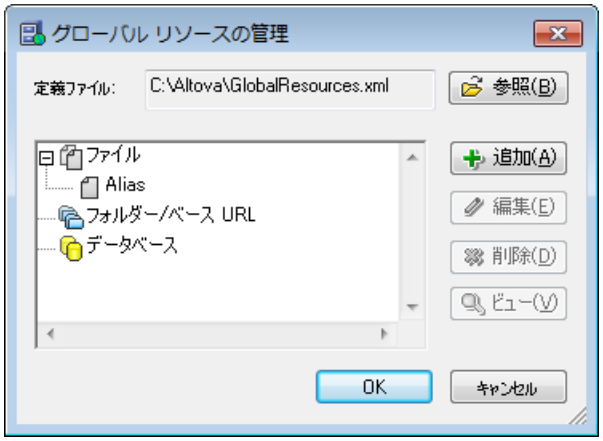
Altova グローバル リソースは、以下の方法によりアクセスすることのできるグローバル リソースの管理ダイアログで定義されます:

- メニュー コマンドの**[ツール| グローバル リソース]** を選択する。
- グローバル リソース ツールバーにある**[グローバル リソースの管理]** アイコンをクリックする(下のスクリーンショット)。



グローバル リソース定義ファイル

グローバル リソースに関する情報はグローバル リソース定義ファイルと呼ばれるXML ファイルに収められています。このファイルは、グローバル リソースの管理ダイアログにて初めてグローバル リソースが定義ならびに保存された際に作成されます (下のスクリーンショットを参照)。



グローバル リソースの管理ダイアログを初めて開くと、グローバル リソース定義ファイルのデフォルトの場所と名前が、定義ファイル テキスト ボックスにて指定されます (上のスクリーンショットを参照):

```
C:\Users\<username>\My Documents\Altova\GlobalResources.xml
```

このファイルは、すべてのAltova アプリケーションに対するデフォルトのグローバル リソース定義ファイルとしてセットされます。従って、任意のAltova アプリケーションからこのファイルに対して保存されたグローバル リソースは、その他すべてのAltova アプリケーションにて直ちにグローバル リソースとして使用することができることになります。グローバル リソースを定義して、グローバル リソース定義ファイルに保存するには、グローバル リソースの管理ダイアログにてグローバル リソースを追加し、**[OK]** ボタンをクリックしてください。

既に存在するグローバル リソース定義ファイルを、実行中のAltova アプリケーションにおけるアクティブな定義ファイルとして選択するには、定義ファイル テキスト ボックスの隣にある**[参照]** ボタンから、そのファイルを選択してください。(上のスクリーンショットを参照)

メモ: グローバル リソース定義ファイルの名前は自由に変更することができ、Altova アプリケーションからアクセスできる任意の場所へ配置することができます。この場合、各アプリケーションにて (定義ファイル テキスト ボックスから) このファイルをアプリケーションのグローバル リソース定義ファイルとして指定する必要があります。複数のAltova 製品にて同一の定義ファイルを指定することで、これらのAltova 製品にてリソースが利用できるようになります。

メモ: 複数のグローバル リソース定義ファイルを作成することもできます。この場合、各 Altova アプリケーションにて同時に選択できるアクティブな定義ファイルは 1 つだけで、そのファイルの中に含まれている定義だけがアプリケーションでは

利用できるということに注意してください。そのため、必要に応じて同じリソースを複数の定義ファイルで記述する必要が生じることもあります。

グローバル リソースの管理：追加、編集、削除、保存

グローバル リソースの管理ダイアログでは、選択されたグローバル リソース定義ファイルへグローバル リソースを追加することができるほか、選択されたグローバル リソースを編集または削除することができます。ファイル、フォルダー、そしてデータベースのグループに追加されたグローバル リソースは、グローバル リソース定義ファイルにより管理されます（上のスクリーンショットを参照）。

グローバル リソースを追加するには、[追加] ボタンをクリックして、表示される**[グローバル リソース]** ダイアログにてグローバル リソースを定義します（このセクション以下のサブセクションにある[ファイル]、[フォルダー]、[データベース] の記述を参照ください）。グローバル リソースを定義して（グローバル リソースの管理ダイアログにある**[OK]** ボタンをクリックすることで）保存すると、選択されたグローバル リソース定義ファイルのグローバル リソース定義ライブラリに、そのグローバル リソースが追加されます。グローバル リソースはエイリアスにより識別されます。

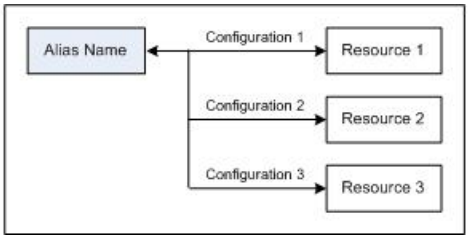
グローバル リソースを編集するには、**[編集]** ボタンをクリックします。関連するグローバル リソース ダイアログが表示され、必要な変更を行うことができます（[ファイル]、[フォルダー]、[データベース] サブセクションの記述を参照ください）。

グローバル リソースを削除するには、**[削除]** ボタンをクリックします。

グローバル リソースの追加、編集、または削除を終えた後には、グローバル リソースの管理ダイアログにある**[OK]** ボタンをクリックして、グローバル リソース定義ファイルに**変更点を保存**するようにしてください。

構成によりグローバル リソースをエイリアス名へ関連付ける

グローバル リソースを定義すると、エイリアス名をリソース（ファイル、フォルダー、データベース）へマッピングすることになります。1 つのエイリアス名から複数のリソースへのマッピングを行うことができ、各マッピングは構成と呼ばれます。そのため、単一のエイリアス名から、複数の構成を介して複数のリソースへの関連付けを行うことができます（以下の図を参照）。



Altova アプリケーションではファイルの代わりにエイリアスを割り当てることができます。アプリケーションのアクティブなグローバル リソース構成を切り替えることで、各エイリアスに割り当てられたリソースを切り替えることができます。例えば Altova XMLSpy アプリケーションにて、MyXML.xml という名前の XML ドキュメントに対して XSLT 変換を行う場合、MyXSLT という名前のエイリアスを使用される XSLT としてグローバル リソースに割り当てることができます。XMLSpy では、アクティブな構成を変更することで、異なる XSLT ファイルを使用することができます。構成 -1 にて First.xslt が MyXSLT へマッピングされており、構成 -1 がアクティブな構成として選択されている場合、First.xslt が変換に使用されます。これにより、複数の構成を使用することで、単一のエイリアスから複数のリソースへアクセスすることができるようになります。このメカニズムはリソースのテストや比較を行う際に便利な機能となります。更に、グローバル リソースは複数の Altova 製品間で使用することができるため、複数の Altova 製品にてリソースのテストや比較を行うことができます。

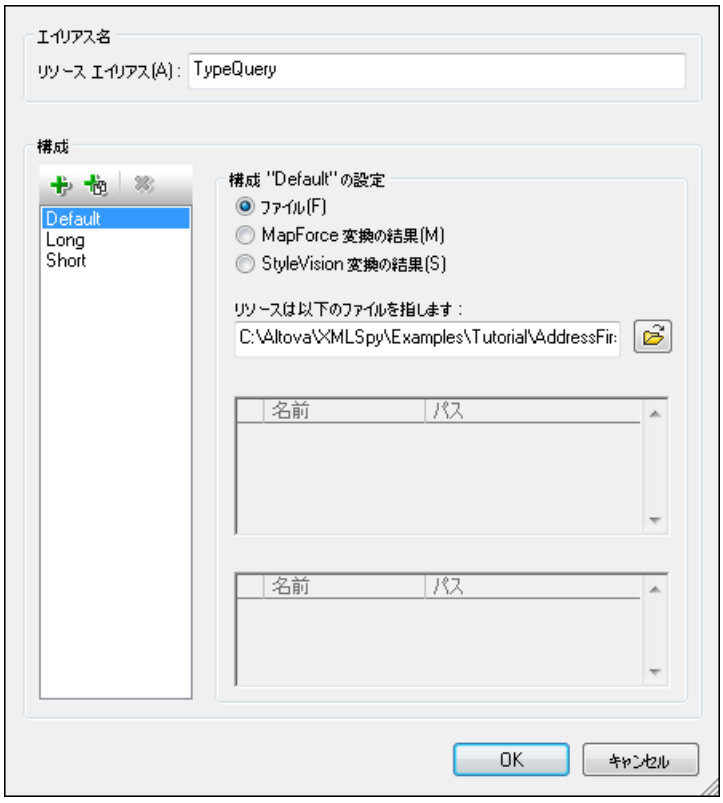
16.1.1 ファイル

[グローバル リソースの管理] ダイアログにある**[追加 | ファイル]** コマンドにより、ファイルのグローバル リソース ダイアログへアクセスすることができます (以下のスクリーンショットを参照)。グローバル リソース ダイアログでは、リソース エイリアス テキスト ボックスにて入力されたエイリアスの構成を定義することができます。以下に記される方法により構成のプロパティを指定して、**[OK]** をクリックしてエイリアスの定義を保存してください。

エイリアスの定義を保存した後には上記の操作を ([グローバル リソースの管理] ダイアログにある [追加 | ファイル] コマンドを選択するところから) 再度行うことで、別のエイリアスを追加することができます。

グローバル リソース ダイアログ

エイリアスはグローバル リソース ダイアログにて定義されます (以下のスクリーンショットを参照)。



グローバル リソース ダイアログのアイコン

構成の追加: 追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

構成をコピーして追加: 選択された構成のコピーとして追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

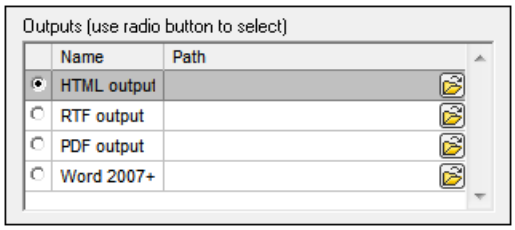
削除: 選択された構成を削除します。

開く グローバル リソースとして作成されるファイルを参照します。

エイリアスの定義

以下の操作により、エイリアスの名前と構成を定義することができます：

1. **エイリアスに名前を付ける** リソース エイリアス テキスト ボックスにエイリアスの名前を入力します。
2. **構成を追加する** 構成ペインには、デフォルトで (名前変更や削除を行うことができない) Default という名前の構成が表示されます。(i) 構成の追加または構成のコピーを追加アイコンをクリックして、(ii) 表示されるダイアログにて**構成名を入力**することで、必要なだけの**構成を追加**することができます。追加された各構成は構成リストに表示されます。上のスクリーンショットでは、Long と Short という名前の構成が構成リストに追加されています。構成のコピーを追加コマンドにより、選択されている構成をコピーして、修正することもできます。
3. **各構成に対してリソースの種類を選択する** 構成のリストから目的の構成を選択し、構成の設定ペインにて、(i) ファイル、(ii) MapFore 変換の結果、または (iii) StyleVision 変換の結果から、構成のリソースを指定します。対応するラジオ ボタンを選択してください。MapForce または StyleVision 変換の結果が選択された場合、mfd または sps ファイルと対応する入力ファイルを使って MapForce または StyleVision により変換が行われ、その変換結果がリソースとなります。
4. **リソースの種類に対するファイルを選択する** リソースの種類にファイルが選択された場合、リソースのファイル テキスト ボックスにてファイルを選択してください。リソースの種類に変換の結果が選択された場合、ファイルの選択テキスト ボックスにて (MapForce 変換の場合は) mfd ファイルまたは (StyleVision 変換の場合は) sps ファイルを選択してください。変換に対して複数の入力または出力が可能な場合、選択するためのオプションが表示されます。例えば、StyleVision 変換の出力オプションが、インストールされている StyleVision のエディションに合わせて表示されます (以下のスクリーンショットでは、Enterprise Edition の出力が示されています)。



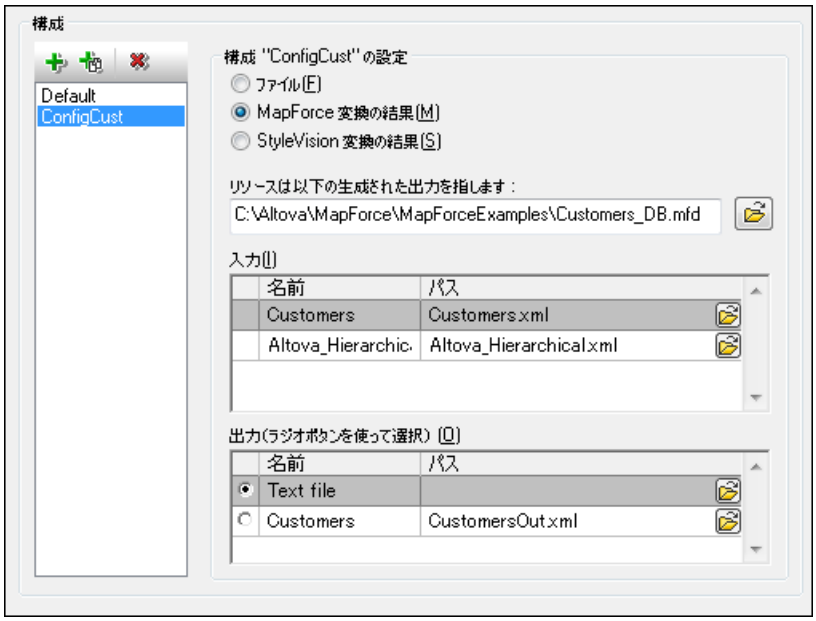
目的のオプションに対応するラジオ ボタンを選択します (上のスクリーンショットでは、[HTML 出力] が選択されています)。変換の結果をリソースとする場合、出力をファイルまたはグローバル リソースとして保存することができます。アイコンをクリックして、[他のグローバル リソースを選択] または、[参照] を選択してください。どちらのオプションも選択されていない場合、グローバル リソースが使用された際に変換の結果が一時ファイルとしてロードされます。
5. **必要に応じて複数の構成を定義する** その他の構成を必要な数だけ追加して、各構成に対してリソースを指定します。各構成に対して上記のステップ 3 と 4 を実行してください。エイリアスに対する新たな構成はいつでも追加することができます。

6. **エイリアスの定義を保存する** **[OK]** をクリックしてエイリアスと、すべての構成をグローバル リソースとして保存します。グローバル リソースが、[グローバル リソースの管理] ダイアログにあるファイル以下に表示されます。

MapForce 変換の結果

Altova MapForce では、(既存の) 入力ドキュメント スキーマから出力ドキュメント スキーマへのマッピングが行われます。MapForce ユーザーにより作成されたこのマッピングは、MapForce Design (MFD) と呼ばれます。入力スキーマに関連付けられた XML ファイル、テキスト ファイル、データベースは、データ ソースとして使用することができ、MapForce により出力ドキュメント スキーマに対応した出力データ ファイルが生成されます。[MapForce 変換の結果] が出力ドキュメントで、グローバル リソースになります。

MapForce により生成されたデータ ファイルをグローバル リソースとしてセットするには、グローバル リソース ダイアログにて、以下の設定を行う必要があります (以下のスクリーンショットを参照)。



- **.mfd (MapForce Design) ファイル:** [リソースは以下の生成された出力を指します] テキスト ボックスにて、mfd ファイルを指定する必要があります。
- **入力データ ファイル:** MFD ファイルが指定されると、そのファイルが解析され、中に含まれる入力スキーマ情報をベースにデフォルトのデータ ファイルが入力ペインに入力されます (上のスクリーンショットを参照)。他のファイルを指定することで、各入力スキーマに対するデフォルト ファイルの選択を変更することができます。
- **出力ファイル:** MFD ドキュメントに複数の出力スキーマが含まれる場合、出力ペインにそれらの出力すべてが表示されます (上のスクリーンショットを参照)。個々の出力スキーマに関連付けられた出力ファイルの場所が MFD ドキュメントにて指定されている場合、その出力スキーマに対するファイルの位置が出力ペインに表示されます。上のスクリーンショットからは、Customers 出力スキーマにデフォルトの XML データ ファイル (CustomersOut.xml) が与えられており、Text ファイル出力 スキーマにはそのような割り当てが行われていないことが理解できます。出力ペインに表示あるデフォルトのファイルの場所を使用することができるほか、自分で指定することができます。MapForce 変換の結果が、選択された出力スキーマがある場所へ保存され、そのファイルがグローバル リソースとして使用されます。

メモ: このオプション (MapForce の結果) により、グローバル リソースが呼び出された時に変換が行われるようになります。これで (入力ファイルから得られた) 最新のデータがグローバル リソースに含まれることが保証されます。

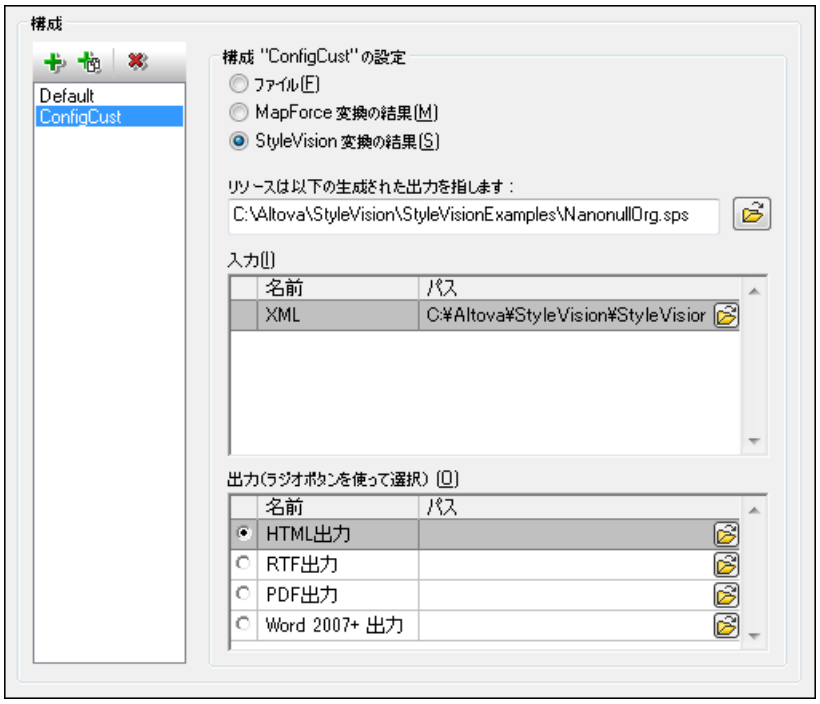
メモ: 変換を行うためには MapForce が使用されるため、この機能を使用するには Altova MapForce をインストールする必要があります。

StyleVision 変換の結果

Altova StyleVision は StyleVision Power Stylesheet (SPS) ファイルを作成するために使用され、これら SPS ファイルにより、XML ドキュメントを様々なフォーマット (HTML、PDF、RTF、Word 2007+ など) の出力ドキュメントへ変換するための XSLT スタイルシートが生成されます。[StyleVision 変換の結果] オプションを選択すると、StyleVision により作成された出力ドキュメントが、選択された構成に関連付けられたグローバル リソースとなります。

グローバル リソース ダイアログにて [StyleVision 変換の結果] オプションを選択すると、以下のファイルを指定する必要が

あります。



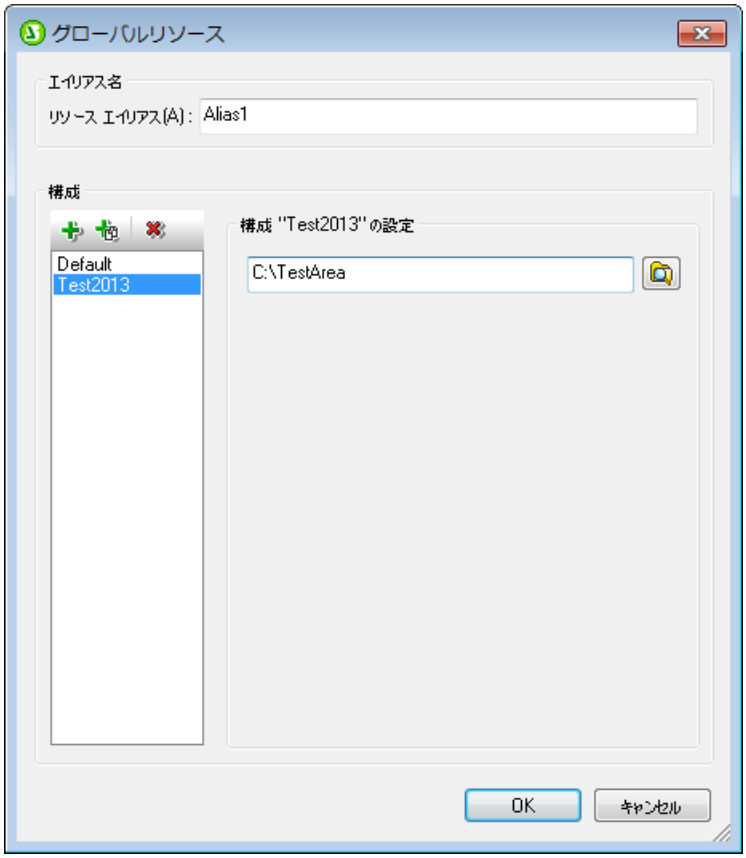
- **sps (SPS) ファイル：**[リソースは以下の生成された出力を指します] テキスト ボックスにて、ファイルを指定する必要があります（上のスクリーンショットを参照）。
- **入力ファイル：**入力ファイルがSPS ファイル内部で既に指定されている場合、SPS ファイルが選択された段階で、そのファイルが入力ペインに表示されます。このエントリーは変更することができるほか、エントリーが指定されていない場合、追加する必要があります。
- **出力ファイル：**出力ペインにて出力フォーマットを選択し、そのフォーマットに対する出力ファイルの場所を指定します。

メモ： このオプション（StyleVision の結果）により、グローバル リソースが呼び出された時に変換が行われるようになります。これで、（入力ファイルから得られた）最新のデータがグローバル リソースに含まれることが保証されます。

メモ： 変換を行うためにはStyleVision が使用されるため、この機能を使用するにはAltova StyleVision をインストールする必要があります。

16.1.2 フォルダー

フォルダーに対するグローバル リソース ダイアログでは、以下の操作によりフォルダー リソースを追加することができます（以下のスクリーンショットを参照）。



グローバル リソース ダイアログのアイコン

- **構成の追加**: 追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。
- **構成をコピーして追加**: 選択された構成のコピーとして追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。
- **削除**: 選択された構成を削除します。
- **開く** グローバル リソースとして作成されるファイルを参照します。

エイリアスの定義

以下の操作によりエイリアスを定義することができます ::

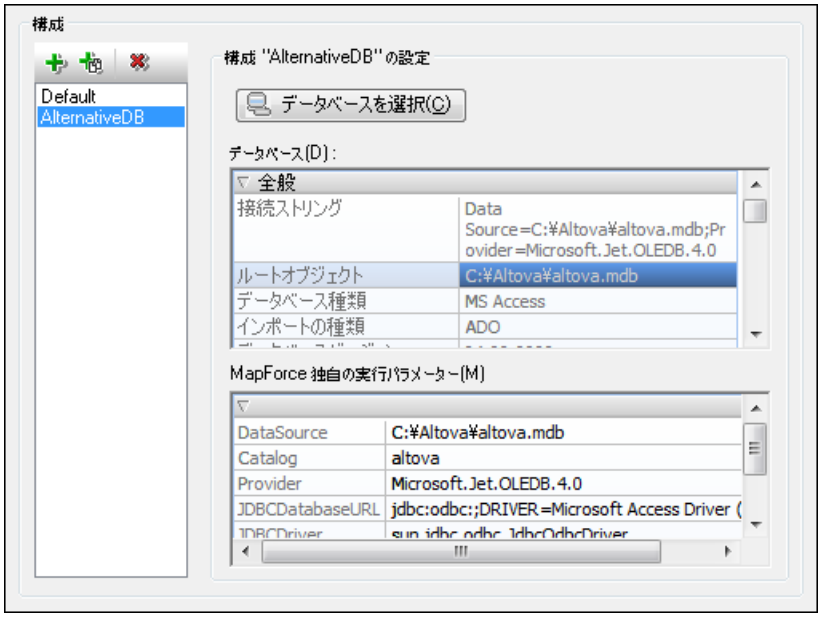
1. **エイリアスに名前を付ける** リソース エイリアス テキスト ボックスにエイリアスの名前を入力します。
2. **構成を追加する** 構成ペインには、デフォルトで (名前変更や削除を行うことができない) Default という名前の構成が表示されます。**構成の追加**または**構成のコピー**を追加アイコンをクリックして、表示されるダイアログにて構成名を入力することで、必要なだけの構成を追加することができます。構成の追加または構成のコピーを追加アイコンをクリックして、表示されるダイアログで構成の名前を入力することで、構成が追加されます。**[OK]** をクリックすることで新たな構成が構成ペインに表示されます。同じ作業を必要な数だけ繰り返してください。
3. 構成のリソースとなるフォルダーを選択する: 構成ペインに表示された構成を選択し、グローバル リソースとして作成するフォルダーを参照します。
4. 必要に応じて複数の構成を定義する: その他の構成を必要な数だけ追加して、各構成に対してリソースを指定します。各構成に対して上記のステップ 2 と 3 を実行してください。エイリアスに対する新たな構成はいつでも追加す

ることができます。

5. エイリアスの定義を保存する:**[OK]** をクリックしてエイリアスと、すべての構成をグローバル リソースとして保存します。グローバル リソースが、[グローバル リソースの管理] ダイアログにあるファイル以下に表示されます。

16.1.3 データベース

データベースに対するグローバル リソース ダイアログでは、以下の操作によりデータベースのリソースを追加することができます:



グローバル リソース ダイアログのアイコン

- **構成の追加**: 追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。
- **構成をコピーして追加**: 選択された構成のコピーとして追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。
- **削除**: 選択された構成を削除します。

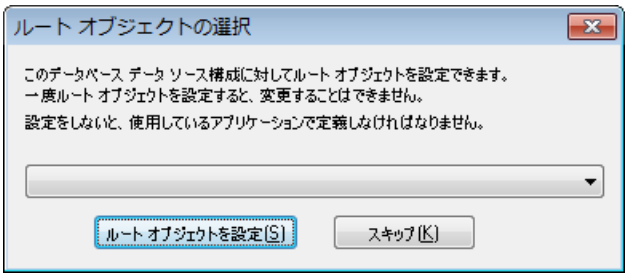
エイリアスの定義

以下の操作によりエイリアスを定義することができます:

1. **エイリアスに名前を付ける** リソース エイリアス テキスト ボックスにエイリアスの名前を入力します。
2. **構成を追加する** 構成ペインには、デフォルトで (名前変更や削除を行うことができない) Default という名前の構成が表示されます。構成の追加または構成のコピーを追加アイコンをクリックして、表示されるダイアログにて構成名を入力することで、必要なだけの構成を追加することができます。構成の追加または構成のコピーを追加アイコンをクリックして、表示されるダイアログで構成の名前を入力することで、構成が追加されます。[OK] をクリックすることで新たな構成が構成ペインに表示されます。同じ作業を必要な数だけ繰り返してください。
3. **構成のリソースとなるデータベースを選択する** 構成ペインに表示された構成を選択し、**[データベースを選択]** ボタンをクリックします。グローバル リソース接続を作成ダイアログが表示されます。
4. **データベースへ接続する** 接続ウィザード、既存の接続、 ADO 接続、ODBC 接続、または JDBC 接続を使用して、データベースへの接続を作成します。データベースへの接続のセクションで説明されている接続メソッドの定義を完成します。接続ウィザード、ADO 接続、または、ODBC 接続を使用することができます。 XMLSpy から既にデータベースへの接続が存在する場合は、既存の接続アイコンをクリックして、表示される接続リストからデ

ータベースを選択します。

5. **ルート オブジェクトの選択**: ルート オブジェクトを選択することのできるデータベースへ接続した場合、サーバー上にあるルート オブジェクトを選択するためのダイアログが表示されます (以下のスクリーンショットを参照)。ルート オブジェクトを選択して、**[ルート オブジェクトを設定]** をクリックしてください。選択されたルート オブジェクトが、この構成が使用された時にロードされるルート オブジェクトとなります。



ルート オブジェクトを選択しなかった場合 (**[スキップ]** ボタンをクリックした場合)、グローバル リソースがロードされた際にルート オブジェクトを選択することになります。

6. **複数の構成を必要に応じて定義する**: 作成されたその他の構成に対してデータベース リソースを指定します (作成された構成に対して上記のステップ 3 から 5 までを繰り返します)。新たな構成をエイリアス定義へ追加することもできます。
7. **ルート オブジェクトの選択**: ルート オブジェクトを選択することのできるデータベースへ接続した場合、サーバー上にあるルート オブジェクトを選択するためのダイアログが表示されます (以下のスクリーンショットを参照)。ルート オブジェクトを選択して、**[ルート オブジェクトを設定]** をクリックしてください。選択されたルート オブジェクトが、この構成が使用された時にロードされるルート オブジェクトとなります。

16.2 グローバルリソースを使用する

グローバルリソースには、ファイル型、フォルダー型、そしてデータベース型と、複数の種類が存在します。XMLSpy で使用することのできる様々なシナリオについて以下で紹介されています：ファイルとフォルダー および データベース。

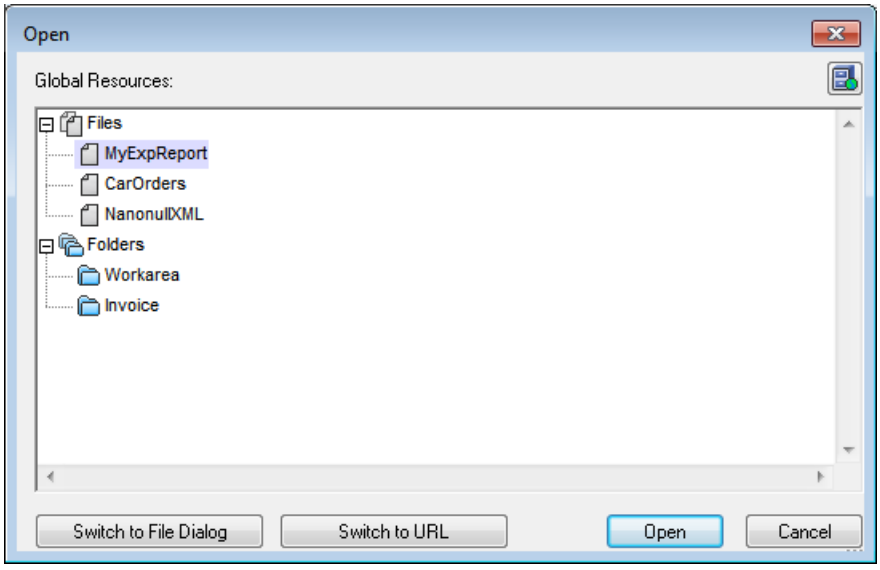
使用するリソースの選択

アプリケーション全体で使用することのできるグローバルリソースの選択、ならびに実際に使用されるグローバルリソースの選択を行うには方法が2つあります：

- [グローバルリソース] ダイアログにてアクティブなグローバルリソース XML ファイルを選択します。アクティブなグローバルリソース XML ファイル内に収められている利用可能なグローバルリソースの定義は、アプリケーション内の開かれているファイル全てで使用することができます。使用することができるのは、アクティブなグローバルリソース XML ファイルにある定義だけとなります。アクティブなグローバルリソース XML ファイルはいつでも変更することができ、新しいファイルに収められたグローバルリソース定義が、それまで使用されていたアクティブなファイルの内容に置き換えられます。従って、アクティブなグローバルリソース XML ファイルにより、(i) 割り当て可能なグローバルリソースと、(ii) 検索可能なグローバルリソースが決定されます（例えば、あるグローバルリソース XML ファイル内のグローバルリソースに割り当てが行われているが、現在アクティブなグローバルリソース XML ファイルにて、対応する名前のリソースがない場合、割り当てられたグローバルリソースの検索を行うことはできなくなります）。
- **[ツール] [アクティブな構成]** または グローバルリソース ツールバーからアクティブな構成 を選択します。メニューコマンド（またはツールバーのドロップダウンリスト）をクリックすることで、全てのエイリアスに対する構成のリストが表示されます。構成を選択することで、その構成がアプリケーションでアクティブな構成となります。この操作により、グローバルリソース（またはエイリアス）が使用される際には、各エイリアスに対して対応するアクティブな構成が適用されます。アクティブな構成は使用される全てのエイリアスに対して適用されます。アクティブな構成と同じ名前の構成をエイリアスが持たない場合、そのエイリアスに対してはデフォルトの構成が適用されます。アクティブな構成はリソースの割り当てに影響を与えません。アクティブな構成は、実際にリソースが使用されるときに重要になります。

16.2.1 ファイルとフォルダーの割り当て

このセクションでは、ファイル型ならびにフォルダー型のグローバルリソース割り当て方法について記述します。ファイル型とフォルダー型のグローバルリソースは異なる方法で割り当てられます。以下にある使用シナリオでは、**[グローバルリソースに切り替え]** ボタンにてグローバルリソースを開くダイアログが表示されます（以下のスクリーンショットを参照）。



グローバルリソースの管理: [グローバルリソースの管理] ダイアログが表示されます。

ファイル型のグローバルリソースを選択することで、ファイルが割り当てられます。**フォルダー型のグローバルリソース**を選択することで、ファイルを開くダイアログが表示され、目的のファイルを選択することができます。選択されたファイルへのパスは、フォルダーリソースからの相対パスとなります。フォルダー型のグローバルリソースに2つの構成が含まれ、各構成が異なるフォルダーを指す場合（そして、それらのフォルダー内にあるファイルの名前が同一の場合）、構成切り替えることで2つの異なるフォルダーをターゲットにすることができます。これは試験などを行う際に便利な機能となります。

ダイアログ下部にあるボタンをクリックすることで、ファイルダイアログまたはURL ダイアログに切り替えることもできます。**[グローバルリソースの管理]** アイコンをクリックすることで、**[グローバルリソースの管理]**ダイアログが表示されます。

使用できるシナリオ

ファイル型やフォルダー型のグローバルリソースに対して以下のような操作を行うことができます：

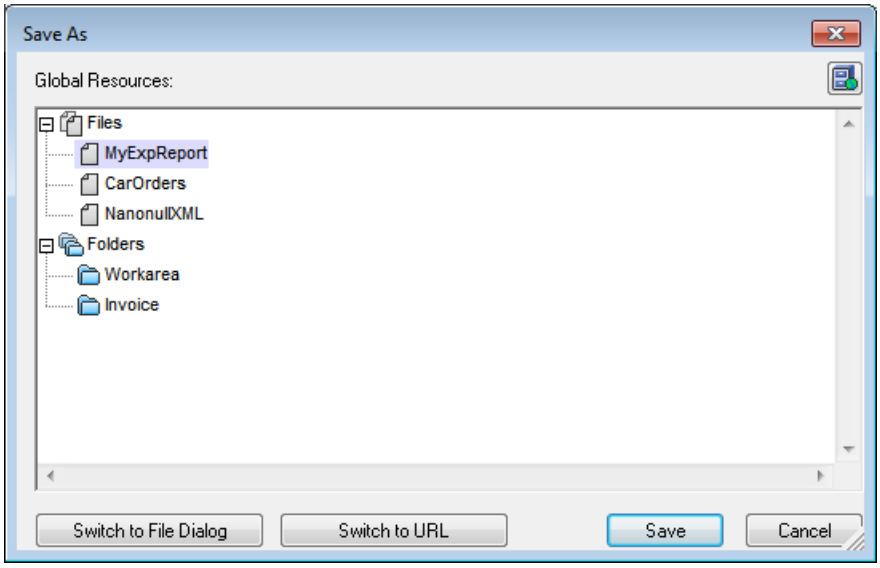
- グローバルリソースを開く
- グローバルリソースとして保存
- XSLT 変換にファイルを割り当てる
- XSLT 変換
- XQuery の実行
- SPS の割り当て

グローバルリソースを開く

グローバルリソースはXMLSpy の **[ファイル | 開く（グローバルリソースに切り替え）]** コマンドから開き編集することができます。ファイル型グローバルリソースの場合、ファイルが直接開かれます。フォルダー型グローバルリソースの場合、割り当てられたフォルダーが選択された状態でファイルを開くダイアログが表示され、フォルダー以下にある目的のファイルを選択することになります。グローバルリソースを通してファイルを選択し、編集する利点は、構成を切り替えるだけで、同一のグローバルリソース内にある（別構成の）ファイルへアクセスできることです。構成を変更する前に、編集がなされたファイルを保存を行う必要があります。

グローバルリソースとして保存

新たに作成されたファイルもグローバルリソースとして保存することができます。同様に、既存のファイルを開いて、グローバルリソースとして保存することもできます。**[ファイル | 保存」または「ファイル | 名前を付けて保存]** コマンドをクリックすると、保存ダイアログが表示されます。**[グローバルリソースに切り替え]** ボタンをクリックして、現在使用されているグローバルリソース XML ファイルにて定義されたエイリアスであるグローバルリソースへアクセスします（以下のスクリーンショットを参照）。



エイリアスを選択し、**[保存]** をクリックします。エイリアスがファイルエイリアスの場合、そのファイルに直接保存されます。エイリアスがフォルダーエイリアスの場合、ダイアログが表示され、保存するファイルの名前を入力するよう促されます。どちらの場合でも、ファイルは現在アクティブな構成により定義された位置に保存されます。

メモ: 各構成ではファイルの位置が構成の定義により指定されます。グローバルリソースとして保存しようとするファイルが、構成で定義された位置にあるファイルと同一の拡張子を持たない場合、このグローバルリソースが開かれたときに、XMLSpy にて編集エラーや検証エラーが発生する可能性があります。これは構成の定義により指定されたファイルの種類を想定して XMLSpy によりファイルが開かれるためです。

XSLT 変換にファイルを割り当てる

グローバルリソースを介して XSLT ファイルを XML ドキュメントに、XML ファイルを XSLT ドキュメントに割り当てることができます。XSLT ファイルを割り当てるコマンド **[XSL/ XQuery | XSL の割り当て]** ならびに **[XSL/ XQuery | XSL:FO の割り当て]** や XML ファイルを割り当てるコマンド **[XSL/ XQuery | サンプル XML の割り当て]** がクリックされると、割り当てダイアログが表示されます。**[参照]** ボタンをクリックすることで、ファイルを開くダイアログが表示され、グローバルリソースを開くダイアログに切り替え、目的のグローバルリソースを選択することができます。グローバルリソースを使用することで、XSLT 変換を行うために指定した XSLT（または XML）ファイルを、XMLSpy の構成を切り替えるだけで変更することができます。異なるファイルに対して変換が必要になった時も、ファイルの割り当て変更を行う必要はありません。XSLT 変換が開始されると、アクティブな構成で指定されるファイルが使用されます。

XSLT 変換 ならびに XQuery の実行

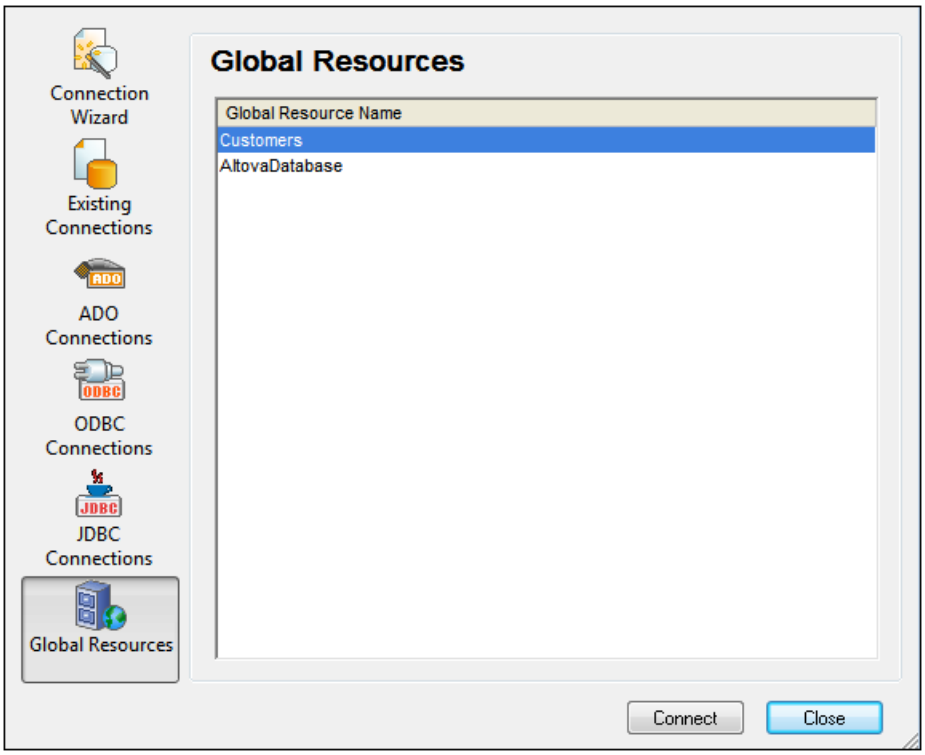
コマンド **[XSL/ XQuery | XSL 変換]** または **[XSL/ XQuery | XSL:FO 変換]** または **[XSL/ XQuery | XQuery 実行]** をクリックすると、XQuery または XML ファイルを指定するためのダイアログが表示されるので、XSLT、XQuery、または XML ファイルを選択します。**[参照]** ボタンをクリックして、**[グローバルリソースに切り替え]** ボタンをクリックすることで、グローバルリソースを開くダイアログが表示されます（セクション最上部のスクリーンショットを参照）。選択されたグローバルリソースの現在アクティブな構成に関連付けられているファイルが変換に使用されます。

SPS の割り当て

StyleVision スタイルシートをXML ファイルに割り当てる際 **[Authentic | SPS ファイルの割り当て]** 、スタイルシートを指定するためにグローバルリソースを使用することができます。**[参照]** ボタンをクリックして、グローバルリソースリソースに切り替えボタンをクリックすることで、グローバルリソース開くダイアログが表示されます(セクション最上部のスクリーンショットを参照)。グローバルリソースを割当先として選択することで、XMLSpy のアクティブな構成を切り替えるだけで、XML ドキュメントの Authentic View を変更することができます。

16.2.2 データベースの割り当て

DB からXMLSpy へ データや XML スキーマとして データ構造をインポートするコマンドを実行すると 例えば、**[変換 | データベースのデータをインポート]** コマンドを使用)、オプションとしてグローバルリソースを使用することができます 以下のスクリーンショットを参照)。メニューにあるデータベースに関連するコマンドでも、データベース型のグローバルリソースを使用することができます。メニューにあるデータベースに関連するコマンドでも、データベース型のグローバルリソースを使用することができます。



データベースを開くダイアログにてグローバルリソースアイコンをクリックすると、現在アクティブなグローバルリソースXML ファイルに定義されている全てのデータベース型グローバルリソースが表示されます。目的のグローバルリソースを選択して、**[接続]** をクリックします。選択されたグローバルリソースに2つ以上の構成が割り当てられている場合、現在アクティブな構成に対応したグローバルリソースが使用され **[ツール | アクティブな構成]** またはグローバルリソースツールバーを確認してください)、接続が確立されます。DB からXML スキーマを作成やDB データのインポートに記述されているように、使用するデータ構造やデータを選択する必要があります。

16.2.3 構成の切り替え

同時にアクティブにすることのできる構成は1つだけで、その設定はアプリケーション全体に対して適用されます。これは、アクティブな構成が、現在開いている全てのファイルやデータソース接続のエイリアスに対して有効であることを意味します。エイリアスにアクティブな構成と同じ名前の構成が割り当てられていない場合、デフォルトの構成がそのエイリアス対して使用されま

す。構成を変更する例として、XSLT ファイルが複数の構成を持つグローバルリソースを介して XML ドキュメントに割り当てられたと仮定しましょう。各構成は異なるファイルにマップされます。ですから、ファイルの選択はアプリケーションのアクティブな構成としてどの構成が選択されているかにより決まります。

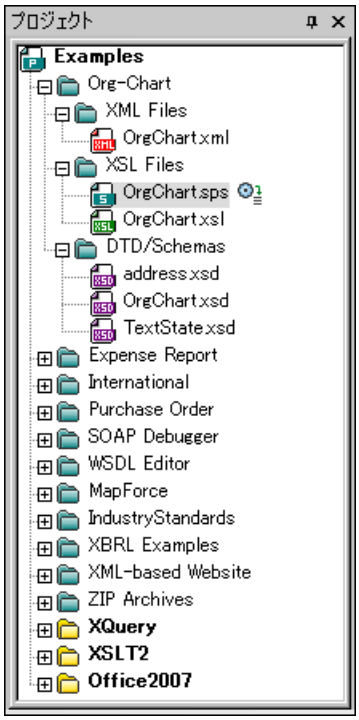
構成の切り替えるには2つの方法があります：

- メニューコマンド **[ツール| アクティブな構成]** を選択すると、グローバルリソース XML ファイル内にある全ての構成がサブメニューに表示されます。目的の構成を選択します。
- グローバルリソースコンボボックスのツールバーから、目的の構成を選択します（以下のスクリーンショットを参照）。

アクティブな構成を切り替えるという方法により、グローバルリソースを介して割り当てられたソースファイルを変更することができます。

17 プロジェクト

プロジェクトとは、何らかの形で関係するファイル同士の集合の事です。例えば、以下のスクリーンショットでは、`Examples` という名前のプロジェクトに、様々なサンプルファイルが、それぞれのフォルダーに収められており、各フォルダーにはサブフォルダーを作成することもできます。例えば`Examples` プロジェクト内の`OrgChart` サンプルフォルダーにはXML、XSL、そしてスキーマファイルを管理するためのサブフォルダーがあります。

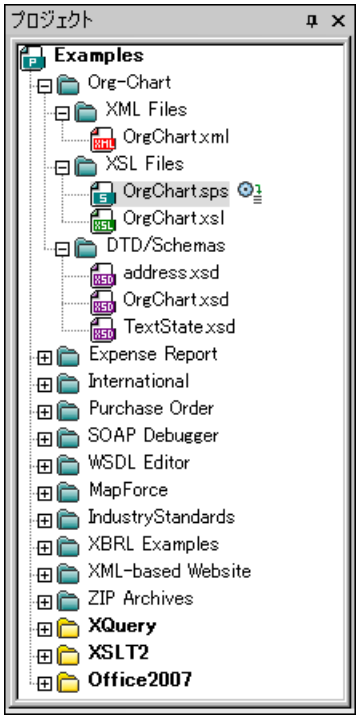


プロジェクトを使用することで、一緒に使用されるファイルをまとめて、ファイルへのアクセスを素早く行うことができます。更に、各フォルダーに対してスキーマやXSLT ファイルを定義することもでき、フォルダーに対するバッチ処理を行うこともできます。

このセクションでは[プロジェクトの作成と編集方法](#)ならびに[プロジェクトの使用方法](#)を説明します。

17.1 プロジェクトの作成と編集

プロジェクトは (以下のスクリーンショットにある) プロジェクトウィンドウとプロジェクトメニューにより管理されます。プロジェクトは一度に1つしか開くことができません。開かれたプロジェクトはプロジェクトウィンドウに表示されます。



プロジェクトの新規作成と既存プロジェクトを開く

メニューコマンド **[プロジェクト | 新規プロジェクトの作成]** により新たなプロジェクトが作成されます。既存のプロジェクトは **[プロジェクト | プロジェクトを開く]** メニューコマンドにより開くことができます。新たに開かれたプロジェクト (新規プロジェクトか既存のプロジェクトかに関わらず) は、それまでプロジェクトウィンドウで開かれていたプロジェクトを置き換えます。それまで開かれていたプロジェクトに (下のスクリーンショットにあるように、フォルダーの隣のアスタリスクとして示される) 保存されていない変更が含まれている場合、変更を保存するかどうかを決定することになります。

プロジェクトの名前付けと保存

新規プロジェクトの名前は保存時に決定されます。プロジェクトの保存には、**[プロジェクト | プロジェクトの保存]** コマンドを使用し、プロジェクトは `.spp` というファイル拡張子で保存されます。プロジェクトが変更されると、その変更を反映するためにプロジェクトを保存しなければなりません。 (プロジェクトウィンドウのトップレベルにある) プロジェクトの名前は、Windows のファイルシステムを使ってしか変更することができず、XMLSpy の GUI を使っての名前変更はできないことに注意してください。サブフォルダーの名前は GUI により変更することができます。

プロジェクトの構成

プロジェクトはフォルダーとファイルから成るツリー構造により管理されます。フォルダーとファイルはどの階層にも作成することができ、その深さにも制限はありません。サブフォルダーやファイル、またはリソースを作成するには、プロジェクトウィンドウ内のフォルダーを選択し、**「プロジェクト」** メニューにあるコマンド、またはコンテキストメニューを使用します。プロジェクトに追加されたファ

イル、フォルダー、そしてリソースを削除したり、ドラッグによりプロジェクトツリー内の他の場所に移動することができます。

新規プロジェクトが作成されると、ファイルをその種類（XML や XSL など）により管理するデフォルトの構造が表示されます（以下のスクリーンショットを参照）。

フォルダーに定義されたプロパティにより、各フォルダーにはファイル拡張子が割り当てられています。ファイルがフォルダーに追加されると、そのファイル拡張子に従って、ファイルは自動的に対応するフォルダーに追加されます。各フォルダーに対して、対応するファイル拡張子を割り当てることができます。

プロジェクトに追加することのできるもの

フォルダー、ファイル、そしてその他のリソースは、トップレベルのプロジェクトフォルダーまたはプロジェクト内にある任意の階層のフォルダーに追加することができます。フォルダーには、(i) プロジェクトフォルダー、(ii) 外部フォルダー、(ii) 外部 Web フォルダー、の3種類あります。

オブジェクトを追加するには、関連するフォルダーを選択し、必要なコマンドをそのフォルダーのコンテキストメニュー、または **プロジェクト** メニューから選択します。以下のオブジェクトをプロジェクトフォルダーに追加することができます：

- *プロジェクトフォルダー（緑色）* はプロジェクトのコンテンツを構造化するためにプロジェクトに追加されるフォルダーです。プロジェクトフォルダーには（そのフォルダーのプロパティから）対応するファイルの拡張子を割り当てることができます。ファイルがフォルダーに追加されると、そのファイルの拡張子にマッチした最初のフォルダー内に配置されます。従って、複数のファイルがフォルダーへ追加されると、それぞれのファイルの拡張子に従って、対応するフォルダーに配置されることになります。
- *外部フォルダー（黄色）* はファイルシステムのフォルダーです。外部フォルダーがプロジェクトに追加されると、その外部フォルダー内にある全てのファイル、サブフォルダー、そしてサブフォルダーのファイルがプロジェクトに追加されます。外部フォルダーに対してファイル拡張子を定義すると、そのフォルダー内のファイルをフィルタリングする効果を持ちます。
- *外部 Web フォルダー* は外部フォルダーに似ていますが、Web サーバー上にあるフォルダーを指定し、アクセスのためにユーザー認証を行う必要があります。外部の Web フォルダーに対してファイル拡張子を定義すると、そのフォルダー内のファイルをフィルタリングする効果を持ちます。
- *ファイル* はフォルダーの選択を行ない、以下の3つのファイル追加コマンドのどれかを選択することでフォルダーに追加することができます：(i) **[プロジェクトにファイルを追加]** コマンドにより開かれるダイアログにて追加するファイルを選択するか、(ii) **[プロジェクトにアクティブなファイルを追加]** コマンドによりメインウィンドウでアクティブなファイルをプロジェクトに追加する、または (ii) プロジェクトに **[アクティブなファイルに関連するファイルを追加]** コマンドによりアクティブな XML ファイルに関連するファイル（例えば XML スキーマや DTD）も一緒にプロジェクトにファイルを追加します。（例えば XSLT ファイルのような）処理命令により割り当てられているファイルは関連するファイルとは見なされないことに注意してください。
- *グローバルリソース* はファイル、フォルダー、そしてデータベースリソースへのエイリアスです。その定義方法と使用方法については、グローバルリソースのセクションにて記述されます。*URL* を使えば URL によりオブジェクトを指定することができます。
- *Altova スクリプトプロジェクト* は asprj 拡張子を持つファイルで、XMLSpy プロジェクトに関連付けることができます。これによりマクロやその他のスクリプトがプロジェクトで利用できるようになります。スクリプトプロジェクトの作成方法と XMLSpy プロジェクトへの割り当て方法はスクリプティングのセクションを参照ください。.

プロジェクトプロパティ

フォルダーのプロパティは、各フォルダーのプロパティダイアログに保管されています。まずフォルダーを選択し、**「プロジェクト」**メニューまたはコンテキストメニュー（フォルダーを右クリック）の**[プロパティ]**コマンドを選択することでアクセスすることができます。プロパティはトップレベルプロジェクトフォルダーだけではなく、プロジェクト内の任意の階層にあるフォルダーに対しても定義することができます。プロパティダイアログでは、以下のプロパティをフォルダーに対して定義、編集することができます。

- *Folder フォルダー名*：トップレベルのプロジェクトフォルダーに対しては編集できません（フォルダー名の代わりにファイルパスが表示されます）。
- *ファイル拡張子*：プロジェクトのトップレベルフォルダーに対しては編集できません。
- *検証*：フォルダー内のXML ファイルの検証において使用されるDTD またはXML スキーマを指定します。
- *変換*：(i) フォルダー内のXML ファイルの変換において使用されるXSLT ファイルと、(ii) フォルダー内のXSLT ファイルが変換を行うXML ファイルを指定します。
- *出力先*：変換の結果生成されるファイルを保存するフォルダーと拡張子を指定します。*Authentic ビューの SPS ファイル*：フォルダー内のXML ファイルをAuthentic ビューにより閲覧、編集するためのSPS ファイルを指定します。

詳細に関しては、**[プロジェクト] プロパティ**コマンドの説明を参照してください。

プロジェクト内のソース管理

プロジェクトではMicrosoft Visual SourceSafe 互換のソース管理システムがサポートされます。この機能の使用方法については、マニュアル内の対応するユーザーリファレンスのセクションを参照ください。

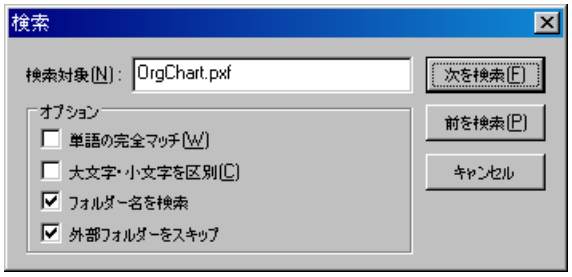
プロジェクトの保存

ファイルの追加や削除、プロジェクトプロパティの修正といったプロジェクトの変更は、**[プロジェクトの保存]**コマンドにより保存されなければなりません。

プロジェクト内の検索

ファイルの名前、またはファイル名の一部からプロジェクト内部にあるファイルやフォルダーを検索することができます。検索が行われると、該当するファイルやフォルダーが1つずつハイライトされた状態で表示されます。

検索を行うには、検索を行うプロジェクトフォルダーをプロジェクトウィンドウにて選択し、**[編集 | 検索]**メニューコマンドを選択するか、「Ctrl」+F キーを押下します。検索ダイアログが表示されるので、検索を行う文字列を入力し、目的に従うかたちで検索オプションを指定します（以下のスクリーンショットを参照）。検索オプションについては以下を参照ください。



以下の検索オプションが利用できます：

- 単語の完全マッチ：入力された検索語とファイルまたはフォルダー名が完全に一致するかチェックされ、厳格な検索を行うことができます。ドット (".") の前後に入力された文字列も単語として認識され、例えばファイルの拡張子だけを入力して検索を行うことができます。
- 大文字小文字を区別：検索語として入力された文字列の大文字と小文字を区別するかどうか指定することができます。
- フォルダー名を検索：フォルダー名を検索の対象に含めます。選択されていない場合、ファイル名だけが検索されます。
- 外部フォルダーをスキップ：外部フォルダーを検索の対象に加える、または対象から外す事ができます。外部フォルダーはシステムまたはネットワーク上に実在しているフォルダーで、プロジェクト内部で論理的に作成されたプロジェクトフォルダーとは異なるものです。

検索が正常に行われた場合、最初にマッチしたアイテムがプロジェクトウィンドウにてハイライトされます。検索ダイアログにある「次を検索」ならびに[前を検索] ボタンをクリックすることで、その他のマッチを参照することができます。

プロジェクトの更新

外部フォルダーに対して変更がなされた場合、プロジェクトが再ロードされるまでその変更はプロジェクトウィンドウに反映されません。

17.2 プロジェクトを使用する

プロジェクトを使うことで同時に複数のファイルに対して設定を行ったり、バッチコマンドを設定したり、ワークスペースの管理を効率的に行うことができます。プロジェクトを使用することで、作業のスピードが飛躍的に向上します。

プロジェクトを使う利点

プロジェクトを使用する利点を以下に示します：

- ファイルの拡張子などの条件により、複数のファイルやフォルダーを1つのフォルダーにグループ化することができます。
- スキーマや XSLT ファイルをフォルダーに割り当てることができます。複数のスキーマやファイルを使って XML ファイルの検証を素早く行うことができます。XML ファイルを複数のフォルダーに追加して、各フォルダーにて異なるスキーマや XSLT を定義してください。
- 各フォルダーに対して、バッチ処理を適用することができます。バッチ処理で使用することのできるコマンドは以下に記されます。
- 変換の出力先となるフォルダーを指定することができます。

素早いアクセスを実現するリソース管理

フォルダーやファイルリソースをツリー構造で管理することができ、プロジェクト内にある様々なフォルダーやファイルの全容を確認し、目的のファイルへ素早く辿りつけることができます。プロジェクトウィンドウで表示されているファイルをダブルクリックして開いてください。プロジェクトへのファイルやフォルダーの追加は、必要に応じて素早く行うことができ、必要でないファイルやフォルダーを削除することもできます。他のプロジェクトで作業を行ないたい場合は、現在開かれているプロジェクトを閉じて、目的のプロジェクトを開いてください。

バッチ処理

フォルダー内にあるファイルのバッチ処理は、それがトップレベルのプロジェクトフォルダーであるかどうかに関わらず、**そのフォルダーのコンテキストメニューから利用できます**（フォルダーにて右クリック）。バッチ処理のステップは以下のとおりです：

1. フォルダーのプロパティダイアログにて検証や変換に使用するファイルを定義します。
2. 変換の出力先となるフォルダーを指定します。出力先のフォルダーが指定されなかった場合、プロジェクトツリー上の親フォルダーにて指定されている出力先フォルダーが使用されます。
3. コンテキストメニューからバッチ処理コマンドを使用します。メニューオプションの XML、**DTD/スキーマ**、または **XSL/XQuery** から対応するコマンドを選択すると、メインウィンドウにてアクティブになっているドキュメントに対してのみコマンドが実行され、プロジェクトウィンドウのプロジェクトフォルダーには影響を与えません。

（トップレベルまたはその他の）プロジェクトフォルダーのコンテキストメニューでは、以下のコマンドがバッチ処理で利用できます：

- *整形式をチェック*：バッチ処理中に発見したエラーは、全てメッセージウィンドウに表示されます。
- *検証*：バッチ処理中に発見したエラーは、全てメッセージウィンドウに表示されます。
- *変換*：フォルダーのプロパティダイアログにて指定した出力フォルダーに変換の出力が保存されます。フォルダーが指定されなかった場合、プロジェクトツリー上の親フォルダーにて指定されている出力先フォルダーが使用されます。親（または祖先）フォルダーにて出力先フォルダーが定義されていない場合、ドキュメントウィンドウが開かれ、各変換の結果が表示されます。XSL-FO 変換では XML または FO ドキュメントを PDF へ変換します。
- *DTD/スキーマの生成*：スキーマが生成される前に、出力先フォルダーを指定するよう促されます。生成されたスキーマファイルはそのフォルダーに保存され、個別のウィンドウにて表示されます。

メモ　バッチ処理を実行するには、プロジェクトウィンドウの対応するフォルダーにてコンテキストメニューを使用してください。メニューオプションの XML、DTD/スキーマ、または XSL/XQuery から対応するコマンドを使用しないでください。これらのコマンドは、メインウィンドウにてアクティブになっているドキュメントに対して実行されます。

RaptorXML Server を使用しての検証とXSLT/XQuery

プロジェクトフォルダーのコンテキストメニューコマンドを使用して、RaptorXML Server をハイパフォーマンスXML 検証とXSLT/XQuery 変換に使用することができます。詳細に関してはRaptorXML Serverのセクションを参照してください。

18 RaptorXML Server

Altova RaptorXML Server が、インストールされ、ネットワーク上でライセンスが与えられており、XMLSpy のインストールにアクセスが与えられている場合、XML および XBRL* ドキュメントを検証し、XSLT とXQuery の変換. の実行をするために、RaptorXML Server (RaptorXML または Raptor と以下省略されます) を使用することができます。XMLSpy プロジェクト フォルダー内のアクティブなドキュメント または すべての ドキュメントを検証することができます。検証の結果 は、メッセージウィンドウの GUI に表示されます。

XMLSpy を使用して以下を行うことができます (i) ドキュメントの検証 (ii) XMLSpy のエンジンまたは RaptorXML Server を使用して、XSLT/XQuery の変換の実行。Raptor を使用する利点の一つは、広範囲に及ぶ検証の個別の検証を構成できることです。更に、Raptor オプションのセットを XMLSpy 内の '構成 ' として、保管することができ、特定の Raptor 検証のための構成を選択します。Raptor の使用は、大きなデータコレクションを検証する際にとても役に立ちます。

メモ: 実際のパフォーマンスは、検証のために RaptorXML Server により使用される PC プロセッサーコアの数によります。使用するコア数が多いほど、処理は速くなります。

*** メモ:** Raptor には、2 つのエディションがあります。: (XML 検証のための) *RaptorXML Server* および (XML および XBRL 検証のための) *RaptorXML+XBRL* 。XBRL ドキュメントを検証する場合は、RaptorXML +XBRL Server を使用します。RaptorXML(+XBRL) Server の詳細に関しては、 Altova Web サイト およびユーザーマニュアル: *RaptorXML Server* および *RaptorXML+XBRL Server* を参照してください。

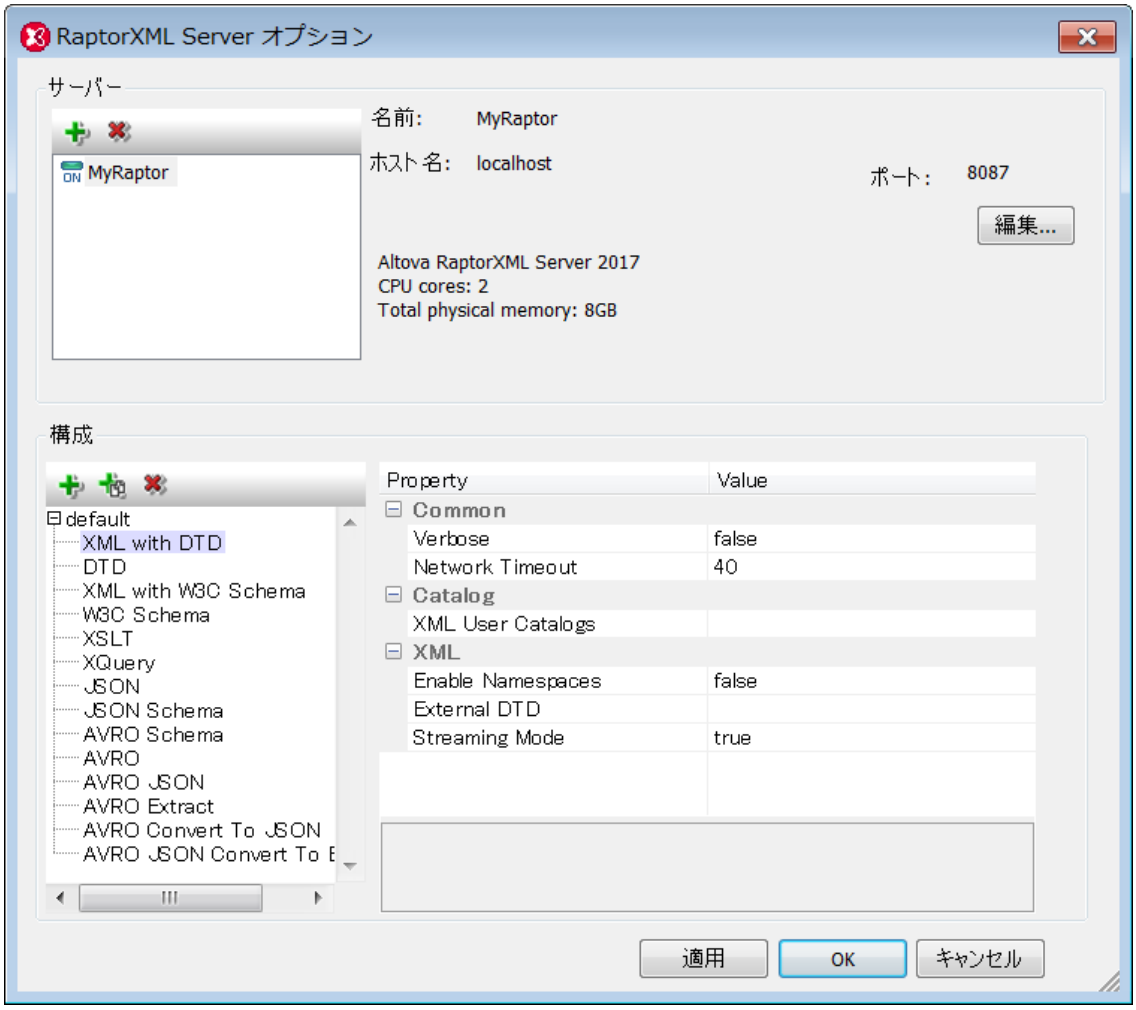
RaptorXML Server を使用して検証または変換する方法

RaptorXML Server を使用して、XML または XBRL ドキュメントを検証、または、XSLT や XQuery 変換を実行するには、XMLSpy は、サーバーへのアクセス方法、検証のため Raptor にパスするオプションなどの使用する RaptorXML Server についての情報を必要とします。XMLSpy 内でこの情報は以下のとおり管理されます :

1. *Raptor サーバーのプールにサーバーを追加する*. このステップでは、,RaptorXML Servers が、プールに追加され、XMLSpy 内に補完されている各サーバーの情報にアクセスすることができます。各サーバーは、名前により識別されています。
2. *各 サーバーのための構成の定義*. 構成は、Raptor 検証 オプションのセットです。各サーバー は、複数の構成を持つことができます。検証に関しては、構成を1つ選択すると、選択された構成がアクティブな構成になります。
3. *検証に使用する サーバー 構成 の選択* 。サーバーとその構成をアクティブな 構成に選択します。アクティブな構成は、後のすべての検証で使用されます。
4. *検証 または Raptor を使用して XSLT/XQuery 変換の実行*。

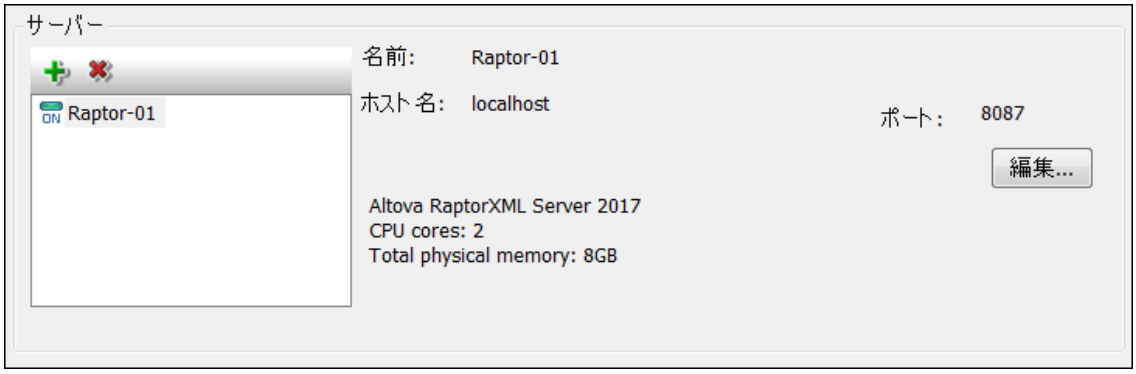
18.1 サーバーとサーバー構成の追加

RaptorXML Server オプションダイアログ (下のスクリーンショット、**[ツール| Raptor Servers の管理]**)内で、Raptor Servers で使用することのできるプールに複数の Raptor Servers の追加 し、各サーバーのための複数の構成の定義 に追加することができます。各サーバーのために定義された構成と共にサーバーを追加するには、**[ツール| Raptor Servers および構成]** サブメニューを使用します。このサブメニューで、Raptor 検証のために使用するを選択します。



Raptor Server を追加する

ダイアログの サーバーペイン (下のスクリーンショット) 内から、**[サーバーの追加]** アイコンをクリックします。アイコンをクリックして、Raptor Server、がインストールされているマシンの名前 (ホスト名) および Raptor Server のポートを認識するための名前を入力します。**[OK]** をクリックして設定を保存します。



- *名前*：選択されるすべての文字列。XMLSpy 内では、特定のRaptorXML Server を認識するために使用されています。
- *ホスト名*：Raptor Server がインストールされているマシンのネットワークの名前 またはIP アドレス。処理はホスト名よりもIP アドレスを使用した場合のほうが速くなります。`localhost`（ローカルマシン）に対応するIP アドレスは`127.0.0.1` です。
- *ポート*：Raptor Server がアクセスされるポート。このポートは、（`server_config.xml` という名前の）Raptor の構成ファイル内で指定されています。ポートは、固定され、サービスへのリクエストが対応できるように知らされている必要があります。Raptor 構成ファイルの詳細は、*RaptorXML Server* および*RaptorXML +XBRL Server* のユーザーマニュアルを参照してください。

サーバー情報を入力すると、**[OK]** をクリックします。入力したサーバー名が（ペインの左側の）サーバーリストに表示されます。サーバーの名前の横に緑のアイコンが表示され、サーバーが開始され、実行中であることを示しています。サーバーの詳細はペインに表示されています（*上のスクリーンショット参照*）。赤いアイコンはサーバーがオフラインであることを示します。サーバーを検出することができない場合、エラーメッセージが表示されます。

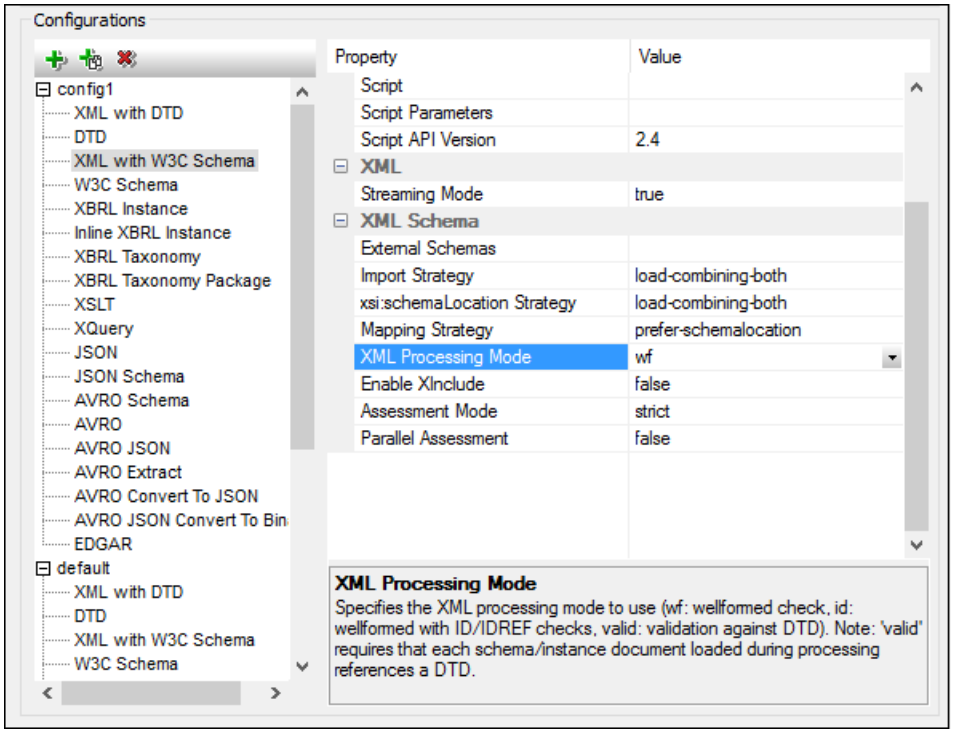
メモ： Raptor Server は、サーバーが追加される際に実行中である必要があります。これはXMLSpy がサーバーの情報を取得し、保管するために必要です。サーバーが追加された後、サーバがオフラインまたは検出されない場合、これらのシナリオは赤いアイコンまたはエラーメッセージで通知されます。

サーバーの名前、ホスト名、またはポートを編集する場合、左側のペインのサーバーを選択して、**[編集]** ボタンをクリックし、表示されるダイアログ内の情報を変更します。プールからサーバーを削除する場合は、サーバーを選択して**[選択されたサーバーを削除]** アイコンをクリックします。

Server 構成

構成とは、RaptorXML 検証オプションのセットです。サーバーが追加されると、構成を持ち、`default` と名前が付けられます。これはRaptorXML オプションのセットに設定されたデフォルトの値です。これらの値を編集することができ、他のオプションの値を含む新しい構成を追加することもできます。複数のサーバー構成を追加した後、アクティブな構成になる1つの構成を選択します。これが**[サーバー上で検証]** コマンドが実行される時に使用される構成です。

構成ペインは2つの部分により構成されています：(i) 検証される構成とドキュメントの型を表示する左側のペイン、(i) 左側のペインで選択された検証型のグループ化されたオプションを表示する右側のペイン。右側のペインの下には選択されたオプションの説明が表示されています（*上のスクリーンショット参照*）。



構成の追加
RaptorXML Server オプションダイアログの構成ペイン内から (*上のスクリーンショット*) **[構成の追加]** をクリックします。デフォルトの オプション値に新規構成が追加されます。**[選択された構成をコピー]** をクリックすることにより新規構成を作成することもできます。これによりコピーされた構成のオプション値と同じ値を持つ新しい構成が作成されます。新たしい構成は、型 `config<X>`のデフォルトの名前と共に作成されます。構成の名前をダブルクリックして新しい名前を入力して編集します。構成のオプション値のすべてを編集することができます。

構成の オプション値の編集
左側のペインから検証ドキュメントを選択します。これにより右側のペイン内にそのグループのオプションが表示されます。オプションの値を編集するには、(*オプションの値の型に従い*)以下を行います ：

- 値が定義済みの値のセットの1つの場合、そのオプションの値の列のコンボボックスから値を選択します。
- 値に制約がある場合、オプションの値フィールドをクリックして、値を入力します。
- 値がファイルパスの場合、値を入力できるだけでなく、オプションの値の列の **[参照]** ボタンを押して、使用するファイルを参照します。

オプションを選択すると、右側のペインの下のボックスに詳細が表示されます。各オプションの詳細は、RaptorXML Server および RaptorXML(+XBRL) Server ユーザーマニュアルのコマンドラインインターフェイスのセクションを参照してください。

構成の削除
左側のペインから、削除する構成を選択して、**[選択された構成を削除]** をクリックします。

Visual Studio および Eclipse 内の XMLSpy

Visual Studio および Eclipse, にXMLSpy が統合されると、IDE のアクティブの構成がXMLSpy のスタンドアロンバージョン内のアクティブな構成として設定されます。

18.2 RaptorXML Server を使用して検証

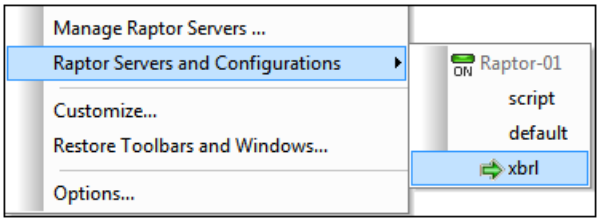
RaptorXML Server を使用して、XML および XBRL* ドキュメントを検証することができます。検証には 2 つの ステップが含まれます :

- 検証のために使用されるサーバーとサーバー構成を選択します。
- (**サーバー上での検証** コマンドを使用した;*以下を参照*)検証の実行

***メモ:** Raptor には、2 つのエディションがあります。: (XML 検証のための)*RaptorXML Server* および (XML および XBRL 検証のための)*RaptorXML+XBRL* 。XBRL ドキュメントを検証する場合は、RaptorXML +XBRL Server を使用します。RaptorXML(+XBRL) Server の詳細に関しては、 Altova Web サイト およびユーザーマニュアル:*RaptorXML Server* および *RaptorXML+XBRL Server* を参照してください。

使用するサーバー構成の選択

複数のサーバーで複数の構成を定義した場合、サーバーとそのアクティブな構成として1つの構成を選択することができます。アクティブな構成は、以降の検証に使用されます。カーソルを **[ツール | Raptor Servers および構成]** コマンドにポイントして、追加されたすべてのサーバーとその詳細が含まれるサブメニューから、を含む表示されるサブメニューが表示されます (*下のスクリーンショット参照*)。アクティブな構成と設定するサーバー構成を選択します。したのスクリーンショットでは、`Raptor-01` という名前の サーバーの 緑色の矢印で示されている)`xbrl` 構成がアクティブな構成として選択されています。



メモ: **[サーバー上で検証]** アイコン のドロップダウンメニュー内のアクティブな構成を選択することもできます。このメニューには、アクティブなサーバー上でEDGAR を検証するコマンドも搭載しています。

RaptorXML Server を使用して検証する

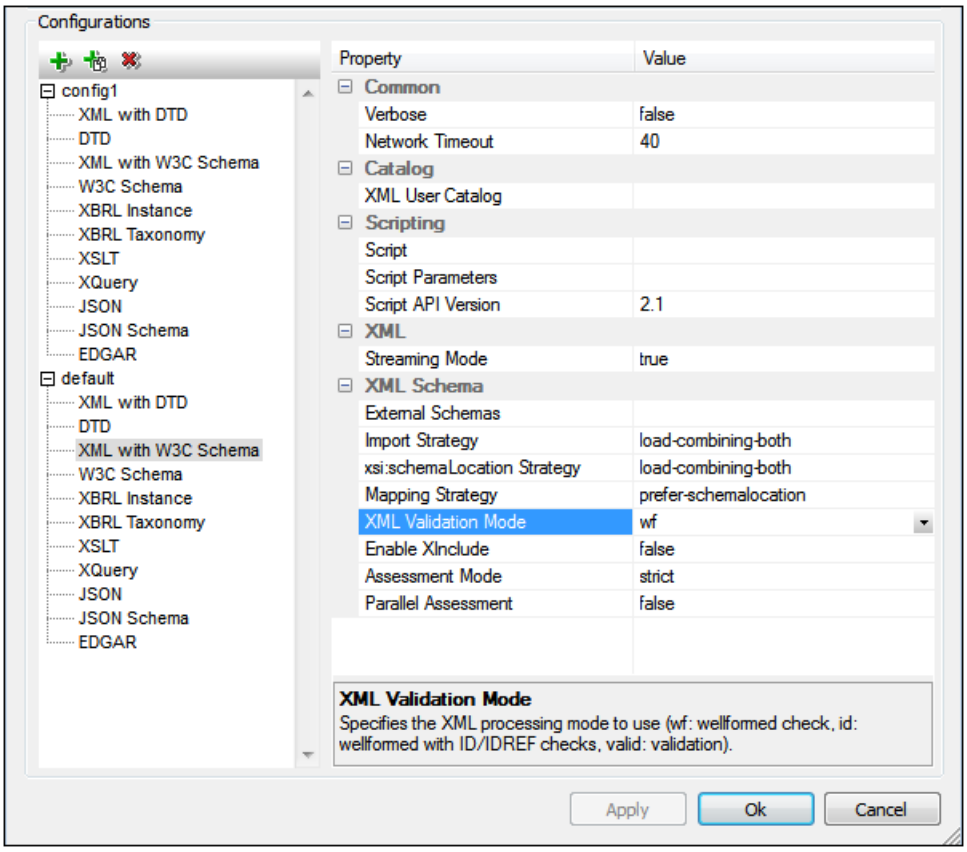
XMLSpy の検証エンジンまたは、RaptorXML Server を使用して XML および XBRL ドキュメントを検証することができます。以下の方法を使用してください :

- ツールバーアイコン **サーバー上で検証する**をクリックします。
- コマンド **[XML | サーバー上で XML の検証する (ハイパフォーマンス)(Ctrl+F8)** を選択します。
- プロジェクト フォルダーまたはファイル内の入力ヘルパーのプロジェクトを右クリックし、**[サーバー上の XML を検証する (ハイパフォーマンス)]**を選択して、選択されたオブジェクト内の XML または XBRL データを検証します。

メモ: テキストビュー、スキーマビュー、XBRL ビュー、および グリッドビュー内で Raptor 検証を使用することができます。

19 検証オプション

このセクションは、検証されるドキュメントの型により整理されています（下のスクリーンショットの左側のペインを参照してください）。例えば、*W3C スキーマを持つXML は、* XML ドキュメントをW3C XML スキーマに対して検証します。左側のペインで検証が選択されると、検証の型で使用することのできるRaptorXML Server 検証 オプションが右側のペインに表示されます。これらの オプションは、*スクリプト*および*XML スキーマ*などのグループ別に整理されています（*下のスクリーンショット参照*）。



このセクションのサブセクションは、それぞれのRaptorXML Server 検証 オプションの説明へのリンクを含みます。

19.1 DTD に対して XML

DTD に対して XML データを検証するためのオプション

- 共通

  詳細
  ネットワークのタイムアウト

- カタログ

  XML ユーザーカタログ

- XML

  名前空間を有効化する
  外部 DTD
  ストリーミングモード

19.2 DTD

DTD の検証のためのオプション

- 共通

  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

- スクリプト

  スクリプト
  スクリプトパラメーター
  スクリプトAPI バージョン

19.3 W3C スキーマに対して XML

XML スキーマに対して XML データを検証するためのオプション

- 共通
  - 詳細
  - ネットワークのタイムアウト
- カタログ
  - XML ユーザー カタログ
- スクリプト
  - スクリプト
  - スクリプト パラメーター
  - スクリプト API バージョン
- XML
  - ストリーミング モード
- XML スキーマ
  - 外部 スキーマ (xsd)
  - インポート ストラテジー (schema-imports)
  - xsi:schemaLocation Strategy (schemalocation-hints)
  - ストラテジーのマッピング (schema-mapping)
  - XML 検証モード (xml-mode)
  - XInclude の有効化 (xinclude)
  - 評価モード
  - パラレル評価

19.4 W3C スキーマ

XML スキーマの検証のためのオプション

- 共通

  詳細
  ネットワークのタイムアウト

- カタログ

  XML ユーザー カタログ

- スクリプト

  スクリプト
  スクリプト パラメーター
  スクリプト API バージョン

- XML スキーマ

  インポートストラテジー (schema-imports)
  xsi:schemaLocation Strategy (schemalocation-hints)
  ストラテジーのマッピング (schema-mapping)
  XML 検証モード (xml-mode)
  XInclude の有効化 (xinclude)

19.5 XBRL インスタンス

XBRL インスタントドキュメントの検証のためのオプション

- 共通

  詳細
  ネットワークのタイムアウト

- カタログ

  XML ユーザー カタログ

- スクリプト

  スクリプト
  スクリプト パラメーター
  スクリプトAPI バージョン

- XML スキーマ

  インポートストラテジー (schema-imports)
  xsi:schemaLocation Strategy (schemalocation-hints)
  ストラテジーのマッピング (schema-mapping)
  XInclude の有効化 (xinclude)
  パラレル評価

- XBRL

  ディメンション拡張子の有効化 (dimensions)
  拡張可能な列挙 拡張子の有効化 (extensible-enumerations)
  ユニットレジストリ名前空間の有効化 (utr)
  XBRL スペックスキーマのプリロード (preload-xbrl-schemas)
  タクソノミパッケージ
  参照されているDTS のみを検証 (validate-dts-only)
  XBRL 矛盾をエラーとして扱う (treat-inconsistencies-as-errors)

- XBRL フォーミュラ

  フォーミュラ拡張子の有効化 (formula)
  アサーション重要度の拡張の有効化 (assertion-severity)
  フォーミュラスペックスキーマのプリロード (preload-formula-schemas)
  条件を満たさないアサーションのレポート
  検証メッセージ言語 (message-lang)
  検証メッセージの役割 (message-role)

- XBRL テーブル

  テーブル拡張子の有効化 (table)
  テーブルスペックスキーマのプリロード (preload-table-schemas)
  テーブルリンクベース名前空間

19.6 XBRL タクソノミ

XBRL タクソノミの検証のためのオプション

- 共通
  - 詳細
  - ネットワークのタイムアウト
- カタログ
  - XML ユーザー カタログ
- スクリプト
  - スクリプト
  - スクリプト パラメーター
  - スクリプトAPI バージョン
- XML スキーマ
  - インポートストラテジー (schema-imports)
  - xsi:schemaLocation Strategy (schemalocation-hints)
  - ストラテジーのマッピング (schema-mapping)
  - XInclude の有効化 (xinclude)
- XBRL
  - ディメンション拡張子の有効化 (dimensions)
  - Enable 拡張可能な列挙 拡張子 (extensible-enumerations)
  - XBRL スペックスキーマのプリロード (preload-xbrl-schemas)
  - タクソノミパッケージ
  - XBRL 矛盾をエラーとして扱う (treat-inconsistencies-as-errors)
- XBRL フォーミュラ
  - フォーミュラ拡張子の有効化 (formula)
  - フォーミュラスペックスキーマのプリロード (preload-formula-schemas)
- XBRL Table
  - テーブル拡張子の有効化 (table)
  - テーブルスペックスキーマのプリロード (preload-table-schemas)
  - テーブルリンクベース名前空間

19.7 XSLT

XSLT ドキュメントの検証のためのオプション

- 共通

  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザー カタログ

- XML スキーマ

  インポートストラテジー (schema-imports)
  xsi:schemaLocation ストラテジー (schemalocation-hints)
  マッピングストラテジー (schema-mapping)
  XML 検証モード (xml-mode)
  XInclude の有効化 (xinclude)

- Java 拡張子

  Java 拡張子の無効化 (javaext-disable)
  バーコード拡張子のロケーション (javaext-barcode-location)

- チャート拡張子

  チャート拡張子の無効化 (chartext-disable)

- .NET 拡張子

  .NET 拡張子の無効化 (dotnetext-disable)

- XEngines 共通

  XML を PSVI と共にロード (load-xml-with-psvi)

- XSLT

  XSLT エンジン バージョン (xslt-バージョン)
  テンプレートモード
  テンプレートエントリポイント

19.8 XQuery

XQuery ドキュメントの検証のためのオプション

- 共通

  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

- XML スキーマ

  インポートストラテジー (schema-imports)
  xsi:schemaLocation ストラテジー (schemalocation-hints)
  マッピングストラテジー (schema-mapping)
  XML 検証モード (xml-mode)
  XInclude の有効化 (xinclude)

- Java 拡張子

  Java 拡張子の無効化 (javaext-disable)
  バーコード拡張子のロケーション (javaext-barcode-location)

- チャート拡張子

  チャート拡張子の無効化 (chartext-disable)

- .NET 拡張子

  .NET 拡張子の無効化 (dotnetext-disable)

- XEngines 共通

  XML をPSVI と共にロード (load-xml-with-psvi)

- XQuery

  XQuery エンジンバージョン (xquery-version)
  XML 宣言の削除

19.9 JSON

JSON (インスタンス) ドキュメントの検証のためのオプション

- 共通

  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

- JSON

  JSON スキーマパス

19.10 JSON スキーマ

JSON スキーマドキュメントの検証のためのオプション.

- 共通

  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

19.11 EDGAR

EDGAR (エドガー、Electronic Data Gathering, Analysis, and Retrieval) は、米国証券取引委員会SEC (Securities and Exchange Commission) に企業により提出されたファイルを自動的に収集、検証、および分類するシステムです。EDGAR を介して検証する場合、Raptor は、内部 EDGAR スクリプトを使用してXBRL インスタンスドキュメントを検証します。以下の追加オプションの設定することことができます。

EDGAR スクリプト パラメーター

EDGAR スクリプトは、EDGAR Filing Manual Volume II: EDGAR Filingで説明されているとおり、追加のチェックを行います。スクリプトは、以下のスクリプト パラメーターを指定することを許可します :

`CIK`	登録者名のCIK
`submissionType`	EDGAR 送信の型、例えば :`'10-K'`
`cikList`	カンマで区切られたCIK のリスト :`','`
`cikNameList`	`'|Edgar|'`により区切られる、`cikList` 内の各 CIK のための公式の登録者名のリスト
`forceUtrValidation`	UTR 検証を強制的に有効化するために`true` に設定
`edbody-url`	埋め込まれたHTML フラグメントの検証に使用される`edbody.dtd` へのパス
`edgar-taxonomies-url`	カンパニー拡張タクソノミタクソノミからの参照を許可するファイルのリストを含む `edgartaxonomies.xml` へのパス

共通

詳細
ネットワーク タイムアウト

カタログ

XML ユーザーカタログ

XML スキーマ

インポートストラテジー (schema-imports)
xsi:schemaLocation ストラテジー (schemalocation-hints)
マッピングストラテジー (schema-mapping)
XInclude の有効化 (xinclude)
パラレル評価

XBRL

ディメンション拡張子の有効化 (dimensions)
拡張可能な列挙 拡張子の有効化 (extensible-enumerations)
XBRL スペックスキーマのプリロード (preload-xbrl-schemas)
タクソノミパッケージ
XBRL 非整合性をエラーとして扱う (treat-inconsistencies-as-エラー)

XBRL フォーミュラ

フォーミュラ拡張子の有効化 (formula)

フォーミュラスペックスキーマのプリロード (preload-formula-schemas)

- XBRL テーブル

テーブル拡張子の有効化 (table)
テーブルスペックスキーマのプリロード (preload-table-schemas)
テーブルリンクベース名前空間

20 RaptorXML Server使用しての XSLT と XQuery

RaptorXML Server を使用して以下を行うことができます (i) XSLT 変換 (ii) XML ドキュメント上のXQuery アップデートの実行。これらのアクションはプロジェクトのみで使用することができ、以下の3つのステップを含みます :

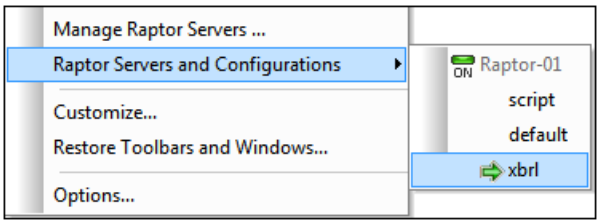
- ジョブに使用されるサーバーとサーバーの構成の選択。
- プロジェクトフォルダーの設定と(プロジェクトプロパティダイアログ内で)使用するXSLT/XQuery ファイルの指定。XSLT/XQuery ファイルは、プロジェクトフォルダー内のすべてのXML ファイルのXSLT とXQuery 変換に使用されたファイルであるフォルダーのプロジェクトプロパティダイアログ内に割り当てられます。プロジェクトフォルダー内の個々のXML ファイルにXSLT/XQuery ファイルを割り当てることはできません。XSLT/XQuery ファイルはフォルダー全体のみに割り当てることができます。
- XSLT 変換またはXQuery 更新および実行

メモ: XSLT またはXQuery ドキュメントがJava 拡張関数 または NET 拡張関数を使用する場合、ファイルパスを使用して、JAR ファイル (Java) または外部 (登録されていない) アセンブリファイル (NET)を検索します。これは、XMLSpy およびRaptorXML Serverを使用して、同じXSLT/XQuery ドキュメントが変換および実行に使用される場合、JAR ファイルへのまたはアセンブリファイルへのファイルパスがこれらのファイルを正確に検索する必要があります。

メモ: RaptorXML Server がXMLSpy と同じマシン上にある場合、パフォーマンス向上のために、サーバー設定を `true` の値を持つ`server.unrestricted-filesystem-access` と設定します。更に詳しい情報は、RaptorXML Server 構成ファイルのドキュメントを参照してください。

使用するサーバー構成の選択

複数のサーバーで複数の構成を定義した場合、サーバーとそのアクティブな構成として1つの構成を選択することができます。アクティブな構成は、以降の検証に使用されます。カーソルを **[ツール| Raptor Servers および構成]** コマンドにポイントして、追加されたすべてのサーバーとその詳細が含まれるサブメニューから、を含む表示されるサブメニューが表示されます (*下のスクリーンショット参照*)。アクティブな構成と設定するサーバー構成を選択します。したのスクリーンショットでは、`Raptor-01` という名前のサーバーの緑色の矢印で示されている)`xbrl` 構成がアクティブな構成として選択されています。



メモ: **[サーバー上で検証]** アイコン のドロップダウンメニュー内のアクティブな構成を選択することもできます。このメニューには、アクティブなサーバー上でEDGAR を検証するコマンドも搭載しています。

XSLT 変換の実行

XMLSpy またはRaptorXML Server のXSLT エンジンを使用してXSLT 変換 の実行をするには以下を行います :

- 変換されるXML ファイルが存在するプロジェクトフォルダーを右クリックします。このフォルダーはプロジェクトフォルダー全体またはプロジェクトの階層内の個別のフォルダーであることができます。
- 表示されるメニューから、**[サーバー上でのXSL 変換 (ハイパフォーマンス)]** を選択します。

メモ: プロジェクトフォルダー内の個々のXML ファイルにXSLT/XQuery ファイルを割り当てることはできません。XSLT/XQuery ファイルはフォルダー全体のみに割り当てることができます。セクションの始めを参照してください。

詳細に関しては、XSLT およびXSLT 変換のセクションを参照してください。

XQuery の更新 /実行の実行

XMLSpy またはRaptorXML Server のXSLT エンジンを使用してXQuery 更新 変換 の実行をするには以下を行います:

- 更新および実行するrXQuery またはXML ファイルが存在するプロジェクトフォルダーを右クリックします。このフォルダーはプロジェクトフォルダー全体またはプロジェクトの階層内の個別のフォルダーであることができます。
- 表示されるメニューから **[サーバー上でのXQuery 更新 実行 (ハイパフォーマンス)]** を選択します。

メモ: プロジェクトフォルダー内の個々のXML ファイルにXSLT/XQuery ファイルを割り当てることはできません。XSLT/XQuery ファイルはフォルダー全体のみに割り当てることができます。セクションの始めを参照してください。

詳細に関しては、XQuery およびXQuery/Update 実行のセクションを参照してください。

21 ファイル /ディレクトリの比較

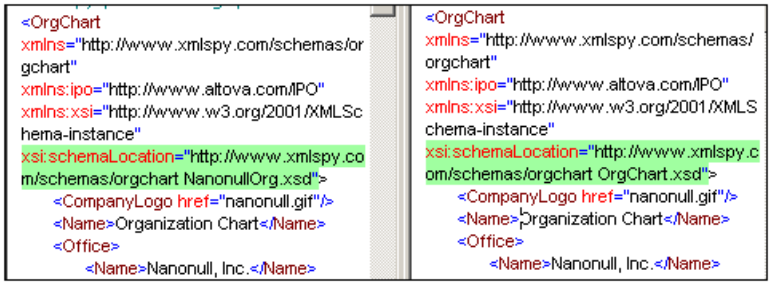
XMLSpy にはファイル比較ならびにディレクトリ比較機能が搭載されています。ファイル比較とディレクトリ比較は、**[ツール]** メニューの **[開いているファイルと比較]** ならびに **[ディレクトリを比較]** コマンドからそれぞれアクセスできます。ファイルの比較オプションは、メニューオプションの **[ツール | 比較オプション]** コマンドによりアクセスすることのできる設定ダイアログにて設定することができます。

各コマンドの詳細はユーザーリファレンスのセクションにて記述されます。このセクションのサブセクションではファイル比較とディレクトリ比較メカニズムの概要について説明します。

21.1 ファイルの比較

ファイル比較機能により、ファイルを開くダイアログやグローバルリソースから選択されたファイルと、アクティブなファイルとの比較を行うことができます。以下ではそのメカニズムの概要を説明します。詳細についてはユーザーリファレンスのセクションを参照ください。

- ファイル比較セッションが開始されると、比較オプションダイアログにて指定された設定が、そのセッションにて有効になります。
- ファイルはXML ファイル（ドキュメントの構造も比較の対象になります）、またはテキストファイルとして比較することができます。設定ダイアログにて、(i) グリッドビューまたはテキストビューを選択し、テキストの比較だけを行うのチェックを外すことでXML 比較を、(i) テキストビューを選択し、テキストの比較だけを行うにチェックを入れることでテキストの比較を行うことができます。
- 選択されたビュー（グリッドビューまたはテキストビュー）にて2つのファイルが隣接して開かれ、両方のファイル間の差異がハイライトされます（以下のスクリーンショットを参照）。

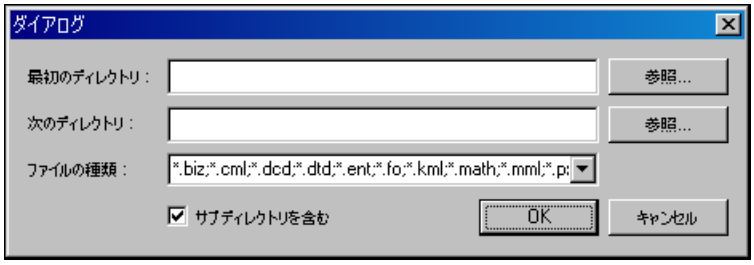


ファイルの比較コントロールウィンドウが表示され、差異のナビゲーションやマージを行うことができます。

設定ダイアログでは、どのXML コンポーネントを比較の対象とし、何を無視するか設定することができます。詳細については、ユーザーリファレンスの比較オプションセクションを参照ください。

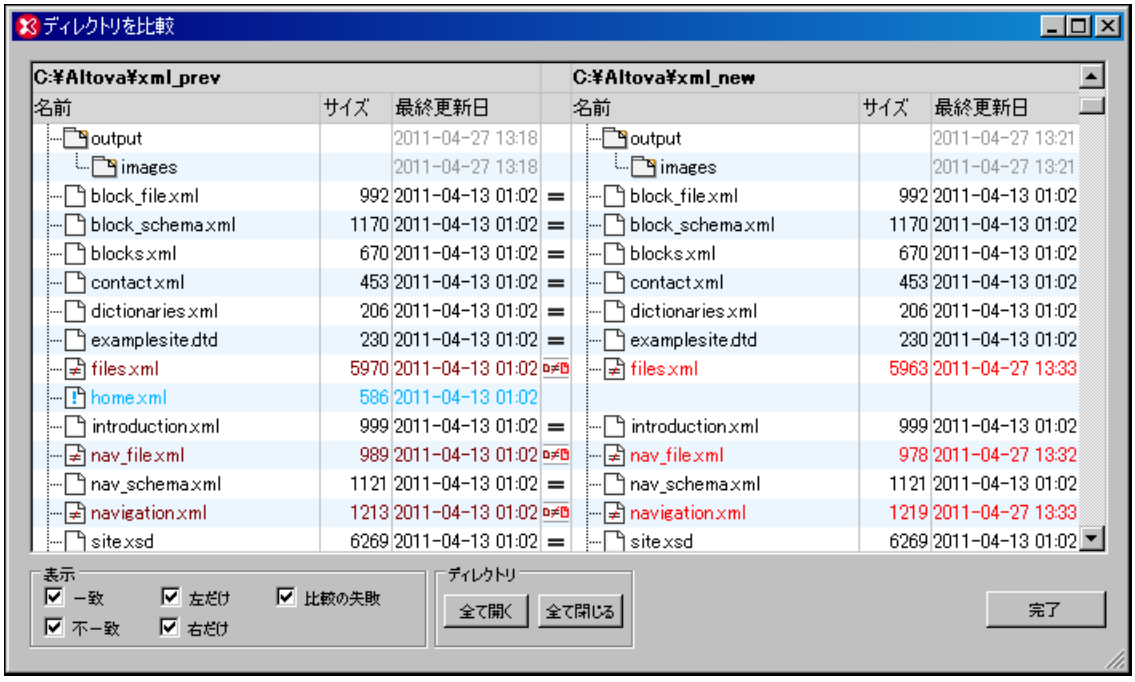
21.2 ディレクトリの比較

ディレクトリの比較機能を使うことで、フォルダーダイアログにて指定したディレクトリ同士を比較することができます。サブディレクトリも比較の対象とするか選択することができ、比較の対象となるファイルの種類を指定することができます。



ファイルの種類のリストの中にZIP ファイル拡張子を加えることで、ZIP アーカイブも比較の対象とすることができます。

ディレクトリの比較では、ファイルが欠けているか、同名のファイルが同じものであるかが比較されます。ファイル同士の比較は設定ダイアログにて指定された設定をベースに行われます。ディレクトリ比較の結果は別のウィンドウにて表示されます（以下のスクリーンショットを参照）。



ディレクトリを比較ウィンドウ内のシンボルの意味や、管理の方法については、ユーザーリファレンスの**[ディレクトリを比較]** コマンドに関する記述を参照ください。ファイルの行をダブルクリックすると、ファイルの比較を開始することができます。

22 Source Control

The source control support in XMLSpy is available through the Microsoft Source Control Plug-in API (formerly known as the MSSCCI API), versions 1.1, 1.2 and 1.3. This enables you to run source control commands such as "Check in" or "Check out" directly from XMLSpy to virtually any source control system that lets native or third-party clients connect to it through the Microsoft Source Control Plug-in API.

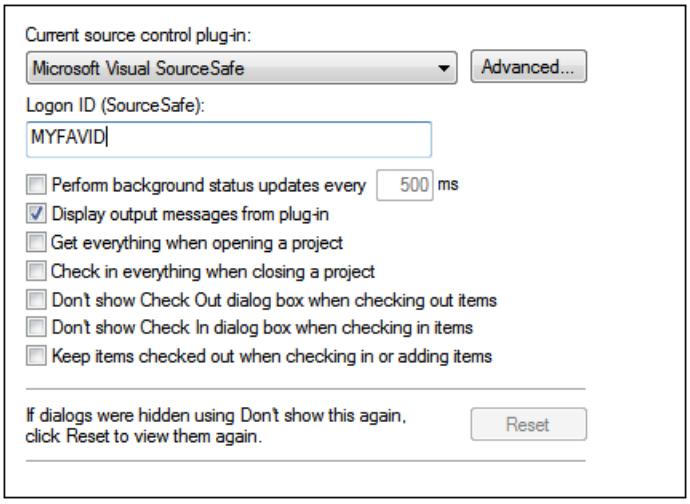
You can use as your source control provider any commercial or non-commercial plug-in that supports the Microsoft Source Control Plug-in API, and can connect to a compatible version control system. For the list of source control systems and plug-ins tested by Altova, see Supported Source Control Systems.

Installing and configuring the source control provider

To view the source control providers available on your system, do the following:

1. On the **Tools** menu, click **Options**.
2. Click the **Source Control** tab.

Any source control plug-ins compatible with the Microsoft Source Code Control Plug-in API are displayed in the **Current source control plug-in** drop-down list.



If a compatible plug-in cannot be found on your system, the following message is displayed:

```
"Registration of installed source control providers could not be found or is
incomplete."
```

Some source control systems might not install the source control plug-in automatically, in which case you will need to install it separately. For further instructions, refer to the documentation of the respective source control system. A plug-in (provider) compatible with the Microsoft Source Code Control Plug-in API is expected to be registered under the following registry entry on your operating system:

```
HKEY_LOCAL_MACHINE\SOFTWARE\SourceCodeControlProvider\InstalledSCCProviders
```

Upon correct installation, the plug-in becomes available automatically in the list of plug-ins available to XMLSpy.

Accessing the source control commands

The commands related to source control are available in the **Project | Source Control** menu.

Resource / Speed issues

Very large source control databases might be introducing a speed/resource penalty when automatically performing background status updates.

You might be able to speed up your system by disabling (or increasing the interval of) the **Perform background status updates every ... seconds** option in the **Source Control** tab accessed through **Tools | Options**.

Note: The **64-bit** version of your Altova application automatically supports any of the supported 32-bit source control programs listed in this documentation. When using a 64-bit Altova application with a 32-bit source control program, the **Perform background status updates every ... seconds** option is automatically grayed-out and cannot be selected.

Differencing with Altova DiffDog

You can configure many source control systems (including Git and TortoiseSVN) so that they use Altova DiffDog as their differencing tool. For more information about DiffDog, see http://www.altova.com/diffdog.html. For DiffDog documentation, see http://www.altova.com/documentation.html.

22.1 Setting Up Source Control

The mechanism for setting up source control and placing files in a XMLSpy project under source control is as follows:

1. If this hasn't been done already, install the source control system (see Supported Source Control Systems) and set up the source control database (repository) to which you wish to save your work.
2. Create a local workspace folder that will contain the working files that you wish to place under source control. The folder that contains all your workspace folders and files is called the local folder, and the path to the local folder is referred to as the local path. This local folder will be bound to a particular folder in the repository.
3. In your Altova application, create an application project folder to which you must add the files you wish to place under source control. This organization of files in an application project is abstract. The files in a project reference physical files saved locally, preferably in one folder (with sub-folders if required) for each project.
4. In the source control system's database (also referred to as source control or repository), a folder is created that is bound to the local folder. This folder (called the bound folder) will replicate the structure of the local folder so that all files to be placed under source control are correctly located hierarchically within the bound folder. The bound folder is usually created when you add a file or an application project to source control for the first time. See the section, Application Project, for information about the repository's folder structure.
5. Project files are added to source control using the command **Project | Source Control | Add to Source Control**. When you add a project or a file in a project for the first time to source control, the correct bindings and folder structure will be created in the repository.
6. Source control actions, such as the checking in and out of files, and the removing of files from source control, can be carried out via commands in the **Project | Source Control** submenu. These commands are described in the Project menu subsection of the User Reference.

Note: If you wish to change the current source control provider, this can be done in one of two ways: (i) via the Source Control options (**Tools | Options | Source Control**), or (ii) in the Change Source Control dialog (**Project | Source Control | Change Source Control**).

22.2 Supported Source Control Systems

The list below shows the Source Control Servers (SCSs) supported by XMLSpy, together with their respective Source Control Clients (SCCs). The list is organized alphabetically by SCS. Note the following:

- Altova has implemented the Microsoft Source Control Plug-in API (versions 1.1, 1.2, and 1.3) in XMLSpy, and has tested support for the listed drivers and revision control systems. It is expected that XMLSpy will continue to support these products if, and when, they are updated.
- Source Code Control clients not listed below, but which implement the Microsoft Source Control Plug-in API, should also work with XMLSpy.

Source Control System	Source Code Control Clients
AccuRev 4.7.0 Windows	AccuBridge for Microsoft SCC 2008.2
Bazaar 1.9 Windows	Aigenta Unified SCC 1.0.6
Borland StarTeam 2008	Borland StarTeam Cross-Platform Client 2008 R2
Codice Software Plastic SCM Professional 2.7.127.10 (Server)	Codice Software Plastic SCM Professional 2.7.127.10 (SCC Plugin)
Collabnet Subversion 1.5.4	• Aigenta Unified SCC 1.0.6 • PushOK SVN SCC 1.5.1.1 • PushOK SVN SCC x64 version 1.6.3.1 • TamTam SVN SCC 1.2.24
ComponentSoftware CS-RCS (PRO) 5.1	ComponentSoftware CS-RCS (PRO) 5.1
Dynamsoft SourceAnywhere for VSS 5.3.2 Standard/Professional Server	Dynamsoft SourceAnywhere for VSS 5.3.2 Client
Dynamsoft SourceAnywhere Hosted	Dynamsoft SourceAnywhere Hosted Client (22252)
Dynamsoft SourceAnywhere Standalone 2.2 Server	Dynamsoft SourceAnywhere Standalone 2.2 Client
Git	PushOK GIT SCC plug-in (see Source Control with Git)
IBM Rational ClearCase 7.0.1 (LT)	IBM Rational ClearCase 7.0.1 (LT)
March-Hare CVSNT 2.5 (2.5.03.2382)	Aigenta Unified SCC 1.0.6
March-Hare CVS Suite 2008	• Jalindi Igloo 1.0.3 • March-Hare CVS Suite Client 2008 (3321) • PushOK CVS SCC NT 2.1.2.5 • PushOK CVS SCC x64 version 2.2.0.4 • TamTam CVS SCC 1.2.40
Mercurial 1.0.2 for Windows	Sergey Antonov HgSCC 1.0.1
Microsoft SourceSafe 2005 with CTP	Microsoft SourceSafe 2005 with CTP
Microsoft Visual Studio Team System 2008/2010 Team Foundation Server	Microsoft Team Foundation Server 2008/2010 MSSCCI Provider

Source Control System	Source Code Control Clients
Perforce 2008 P4S 2008.1	Perforce P4V 2008.1
PureCM Server 2008/3a	PureCM Client 2008/3a
QSC Team Coherence Server 7.2.1.35	QSC Team Coherence Client 7.2.1.35
Reliable Software Code Co-Op 5.1a	Reliable Software Code Co-Op 5.1a
Seapine Surround SCM Client/Server for Windows 2009.0.0	Seapine Surround SCM Client 2009.0.0
Serena Dimensions Express/CM 10.1.3 for Win32 Server	Serena Dimensions 10.1.3 for Win32 Client
Softimage Alienbrain Server 8.1.0.7300	Softimage Alienbrain Essentials/Advanced Client 8.1.0.7300
SourceGear Fortress 1.1.4 Server	SourceGear Fortress 1.1.4 Client
SourceGear SourceOffsite Server 4.2.0	SourceGear SourceOffsite Client 4.2.0 (Windows)
SourceGear Vault 4.1.4 Server	SourceGear Vault 4.1.4 Client
VisualSVN Server 1.6	• Aigenta Unified SCC 1.0.6 • PushOK SVN SCC 1.5.1.1 • PushOK SVN SCC x64 version 1.6.3.1 • TamTam SVN SCC 1.2.24

22.3 Local Workspace Folder

The files you will be working with should be saved in a hierarchy inside a local workspace folder (*see diagram below*).

```
Local Workspace Folder
|
|-- MyProject.spp
|-- QuickStart
|  |-- QuickStart.css
|  |-- QuickStart.xml
|  |-- QuickStart.xsd
|-- Grouping
|  |-- Persons
|  |  |-- Persons.xml
```

The application project file (`.spp` file) typically will be located directly inside the local workspace folder (*see diagram above*).

When one or more files in this (workspace) folder are placed under source control, the local workspace folder's structure is partly or wholly reproduced in the repository. For example, if the file `Persons.xml` from the local folder shown above is placed under source control, then the path to it in the repository will be:

```
[RepositoryFolder]/MyProject/Grouping/Persons/Persons.xml
```

The `MyProject` folder in the repository folder is bound to the local folder. Typically it would be the name of the project, but you could give it any name.

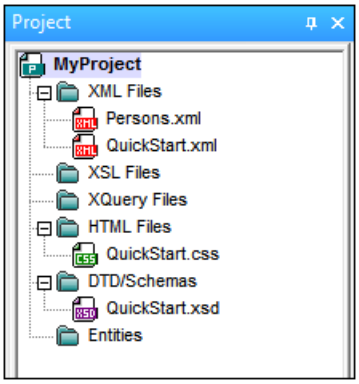
If the entire application project is placed under source control (by selecting the project name in the Projects window and placing it under source control), then the entire local folder structure is recreated in the repository.

Note: Files from outside the local workspace folder can be added to the application project. But whether you can place such a file under source control depends upon the source control system you are using. Some source control systems could have a problem placing a file from outside the local folder into the repository. We therefore recommend that all project files you wish to place under source control be located in the local workspace folder.

22.4 Application Project

Create or load the Altova application project you wish to place under source control. If you wish to place a single file under source control, this file must be included in a project—since source control can only be accessed via a project.

For example, consider a project in Altova's XMLSpy application. The project's properties are saved in a `.spp` file. In the application, the project is displayed in the application's Project window (*see screenshot below*). The project in the screenshot below is named `MyProject` and the project's properties are saved in the file `MyProject.spp`.



You can place the entire project (all files in the project) or only some project files under source control. **Only files that are in the project can be placed under source control.** So you will need to add files to the project before you can place them under source control. The project file (`.spp` file) will automatically be placed under source control as soon as a file from within the project is placed under source control.

The entire project, or one or more project files, is placed under source control via the command **Project | Source Control | Add to Source Control** (*see next section below*).

Note, however, that the folder structure of the repository corresponds not to the project's folder structure (*screenshot above*) but to the structure of the local workspace folder (*see folder diagram below*). In the diagram below, notice that the `MyProject` folder in the repository has a folder structure corresponding to that of the local workspace folder. Note that the bound folder occurs within the repository folder.

```
Local Workspace Folder          Repository
|                               |
|-- MyProject.spp               |-- MyProject (bound to Local
                                Workspace)
|-- QuickStart                  ||-- MyProject.spp
|   |-- QuickStart.css          ||-- QuickStart
|   |-- QuickStart.xml          ||   |-- QuickStart.css
|   |-- QuickStart.xsd          ||   |-- QuickStart.xml
|-- Grouping                    ||   |-- QuickStart.xsd
|   |-- Persons                 ||-- Grouping
|   |   |-- Persons.xml         ||   |-- Persons
                                ||   |   |-- Persons.xml
```

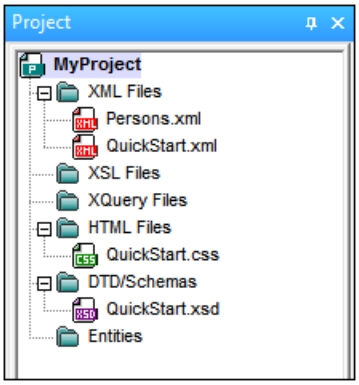
Note: An application project can contain project folders (green) and external folders (yellow). Only files in (green) project folders can be placed under source control. Files in (yellow) external folders cannot be placed under source control.

Note: Files from outside the local workspace folder can be added to the application project. But whether you can place such a file under source control depends upon the source control system you are using. Some source control systems could have a problem placing a file from outside the local folder into the repository. We therefore recommend that all project files you wish to place under source control be located in the local workspace folder.

22.5 Add to Source Control

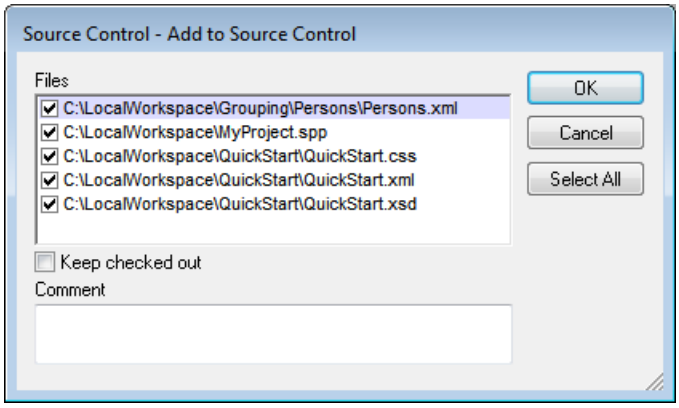
Adding the project to source control will automatically create the correct bindings and repository structure before adding the project file (`.spp` file) or individual files to source control. Add the project to source control as follows.

Select the project in the Project window (`MyProject` in the screenshot below) so that it is highlighted (*as in the screenshot below*). Alternatively select a single file, or select multiple files by clicking them with the **Ctrl** key pressed. Adding a single file to source control will automatically add the project file (`.spp` file) to source control as well.



Next, select the menu command **Project | Source Control | Add to Source Control**. This pops up the connection and configuration dialogs of the currently selected source control system. (You can change the source control system via the Change Source Control dialog (**Project | Source Control | Change Source Control**).)

Follow the source control system's instructions to make the connection and configuration. After this has been completed, all the files selected for addition plus the project file (`.spp` file) are displayed in an Add to Source Control dialog (*screenshot below*). Select the files you wish to add and click **OK**.



The files will be added to the repository and be either checked in or checked out depending on whether the *Keep Checked Out* check box has been checked or not.

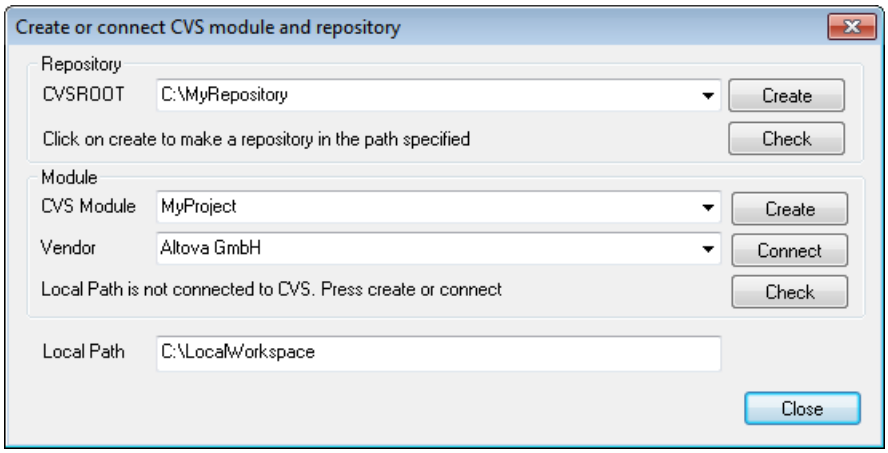
Configuration notes

You might be prompted to create a folder in the repository for the project if it has not already

been created. If you are, go ahead and create it. The local workspace folder will be bound to this folder created in the repository (*see diagrams below*).

```
Local Workspace Folder          Repository
|                               |
|-- MyProject.spp               |-- MyProject (bound to Local
                                    Workspace)
|-- QuickStart                  ||-- MyProject.spp
|   |-- QuickStart.css          ||-- QuickStart
|   |-- QuickStart.xml          ||   |-- QuickStart.css
|   |-- QuickStart.xsd          ||   |-- QuickStart.xml
|-- Grouping                    ||   |-- QuickStart.xsd
|   |-- Persons                 ||-- Grouping
|   |   |-- Persons.xml         ||   |-- Persons
                                ||   |   |-- Persons.xml
```

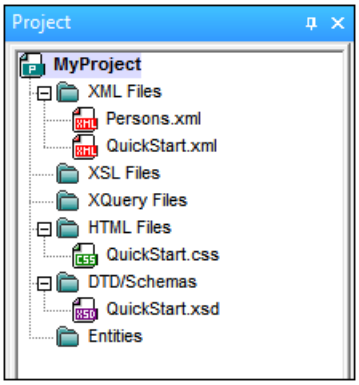
The configuration dialog of Jalindi Igloo is show below. The CVSROOT field is the path to the repository folder.



In the screenshot above, the local path locates the local workspace folder, which corresponds to the CVS module, `MyProject`, and is bound to it.

22.6 Working with Source Control

To work with source control, select the project, a project folder, or a project file in the Project window (*screenshot below*) and then select the command you want in the **Project | Source Control** menu. The **Check In** and **Check Out** commands are available as context menu commands of Project window items.



In this section, we describe the main source control features in detail:

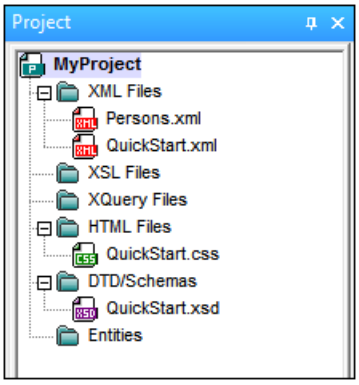
- Add to, Remove from Source Control
- Check Out, Check In
- Getting Files as Read-Only
- Copying and Sharing from Source Control
- Changing Source Control

Additional commands in the **Project | Source Control** menu are described in the User Reference section of the manual. For information specific to a particular source control system, please see the user documentation of that system.

22.6.1 Add to, Remove from Source Control

Adding

After a project has been added to source control, you can place files either singly or in groups under source control. This is also known as adding the files to source control. Select the file in the Project window and then click the command **Project | Source Control | Add to Source Control**. To select multiple files, keep the **Ctrl** key pressed while clicking on the files you wish to add. Running the command on a (green) project folder (*see screenshot below*) adds all files in the folder and its sub-folders to source control.



When files are added to source control, the local folder hierarchy is replicated in the repository (it is not the project folder hierarchy that is replicated). So, if a file is in a sub-folder X levels deep in the local folder, then the file's parent folder and all other ancestor folders are automatically created in the repository.

When the first file from a project is added to source control, the correct bindings are created in the repository and the project file (`.spp` file) is added automatically. For more details, see the section Add to Source Control.

Source control symbols

Files and the project folder display certain symbols, the meanings of which are given below.

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

Removing

To remove a file from source control, select the file and click the command **Project | Source Control | Remove from Source Control**. You can also remove: (i) files in a project folder by executing the command on the folder, and (ii) the entire project by executing the command on the project.

22.6.2 Check Out, Check In

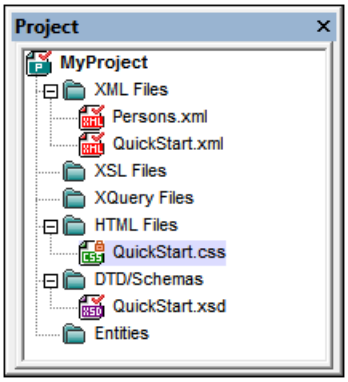
After a project file has been placed under source control, it can be checked out or checked in by selecting the file (in the Project window) and clicking the respective command in the **Project | Source Control** menu: **Check Out** and **Check In**.

When a file is checked out, a copy from the repository is placed in the local folder. A file that is checked out can be edited. If a file that is under source control is not checked out, it cannot be edited. After a file has been edited, the changes can be saved to the repository by checking in the file. Even if the file is not saved in the application, checking it in will save the changes to the repository. Whether a file is checked out or not is indicated with a tick or lock symbol in its Project window icon.

Files and the project folder display certain symbols, the meanings of which are given below.

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

Selecting the project or a folder within the project selects all files in the selected object. To select multiple objects (files and folders), press the **Ctrl** key while clicking the objects. The screenshot below shows a project that has been checked out. The file `QuickStart.css` has subsequently been checked in.



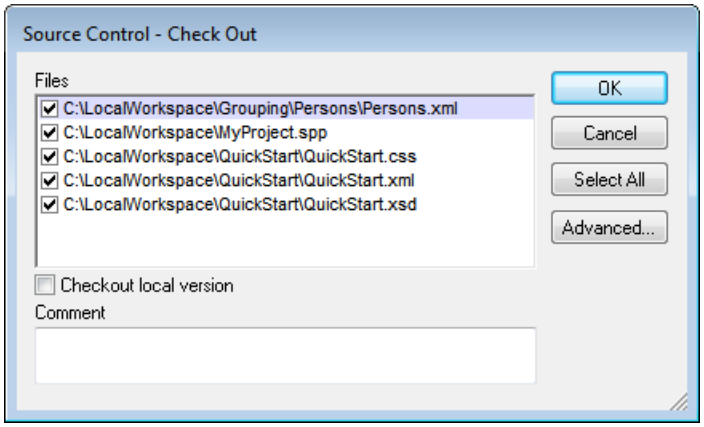
Saving and rejecting editing changes

Note that, when checking in a file, you can choose to leave the file checked out. What this does is save editing changes to the repository while continuing to keep the file checked out, which is useful if you wish to periodically save editing changes to the repository and then continue editing.

If you have checked out a file and made editing changes, and then wish to reject these changes, you can revert to the document version saved in the repository by selecting the command **Project | Source Control | Undo Check Out**.

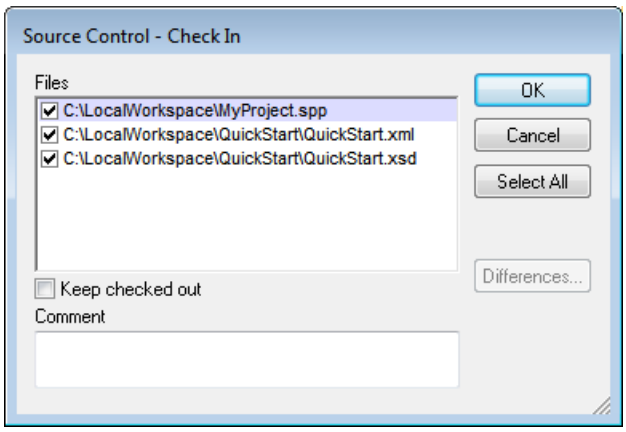
Checking out

The Check Out dialog (*screenshot below*) allows you: (i) to select the files to check out, and (ii) to select whether the repository version or the local version should be checked out.



Checking in

The Check In dialog (*screenshot below*) allows you: (i) to select the files to check in, and (ii) if you wish, to keep the file checked out.

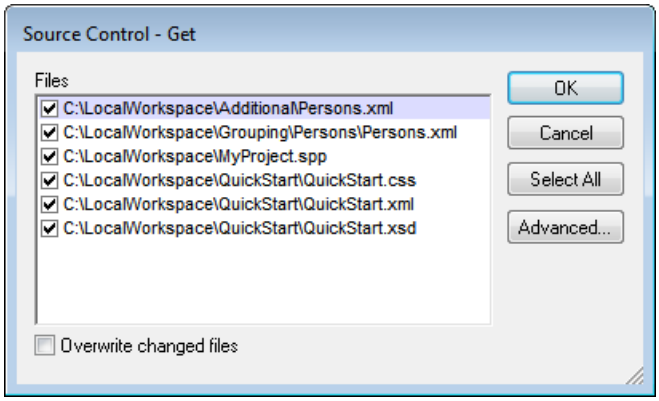


Note: In both dialogs (Check Out and Check In), multiple files appear if the selected object (project or project folder/s) contain multiple files.

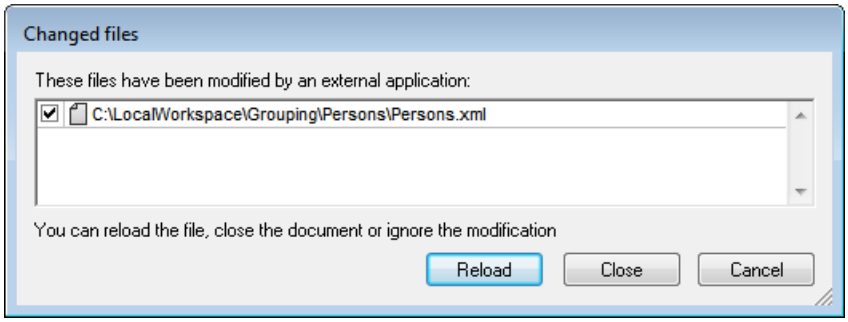
22.6.3 Getting Files as Read-Only

The **Get** command (in the **Project | Source Control** menu) retrieves files from the repository as read-only files. (To be able to edit a file, you must check it out.) The Get dialog lists the files in the object (project or folder) on which the **Get** command was executed (*see screenshot below*). You can select the files to retrieve by checking them in the Get dialog list.

Note: The **Get Folders** command allows you to select individual sub-folders in the repository if this is allowed by your source control system, .

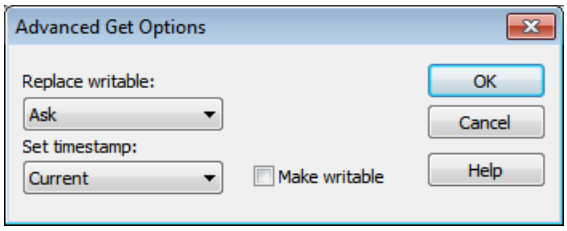


You can choose to overwrite changed checked-out files by checking this option at the bottom of the Get dialog. On clicking **OK**, the files will be overwritten. If any of the overwritten files is currently open, a dialog pops up (*screenshot below*) asking whether you wish to reload the file/s (**Reload** button), close the file/s (**Close**), or retain the current view of the file (**Cancel**).



Advanced Get Options

The Advanced Get Options dialog (*screenshot below*) is accessed via the **Advanced** button in the Get dialog (*see first screenshot in this section*).



Here you can set options for (i) replacing writable files that are checked out, (ii) the timestamp, and (iii) whether the read-only property of the retrieved file should be changed so that it will be writable.

Get latest version

The **Get Latest Version** command (in the **Project | Source Control** menu) retrieves and places the latest source control version of the selected file(s) in the working directory. The files are retrieved as read-only and are not checked out. This command works like the **Get** command (*see above*), but does not display the Get dialog.

If the selected files are currently checked out, then the action taken will depend on how your source control system handles such a situation. Typically, the source control system will ask whether you wish to replace, merge with, or leave the checked-out file as it is.

Note: This command is recursive when performed on a folder, that is, it affects all files below the current one in the folder hierarchy.

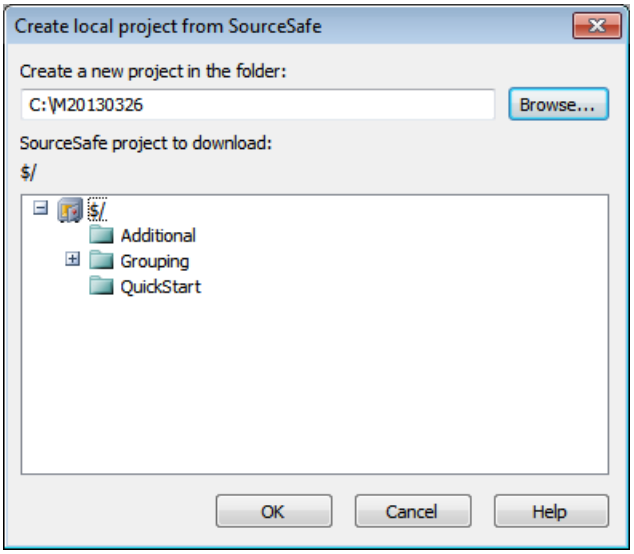
22.6.4 Copying and Sharing from Source Control

The **Open from Source Control** command creates a new application project from a project under source control.

Create the new project as follows:

1. Depending on the source control system used, it might be necessary, before you create a new project from source control, to make sure that no file from the source-controlled project is checked out.
2. No project need be open in the application, but can be.
3. Select the command **Project | Source Control | Open from Source Control**.

4. The source control system that is currently set will pop up its verification and connection dialogs. Make the connection to the bound folder in the repository that you want to copy.
5. In the dialog that pops up (*screenshot below*), browse for the local folder to which the contents of the bound folder in the repository (that you have just connected to) must be copied. In the screenshot below the bound folder is called `MyProject` and is represented by the `$` sign; the local folder is `C:\M20130326`.

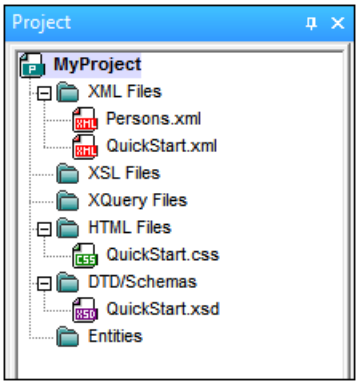


6. Click **OK**. The contents of the bound folder (`MyProject`) will be copied to the local folder `C:\M20130326`., and a dialog pops up asking you to select the project file (`.spp` file) that is to be created as the new project.
7. Select the `.spp` file that will have been copied to the local folder. In our example, this will be `MyProject.spp` located in the `C:\M20130326` folder. A new project named `MyProject` will be created in the application and will be displayed in the Project window. The project's files will be in the folder `C:\M20130326`.

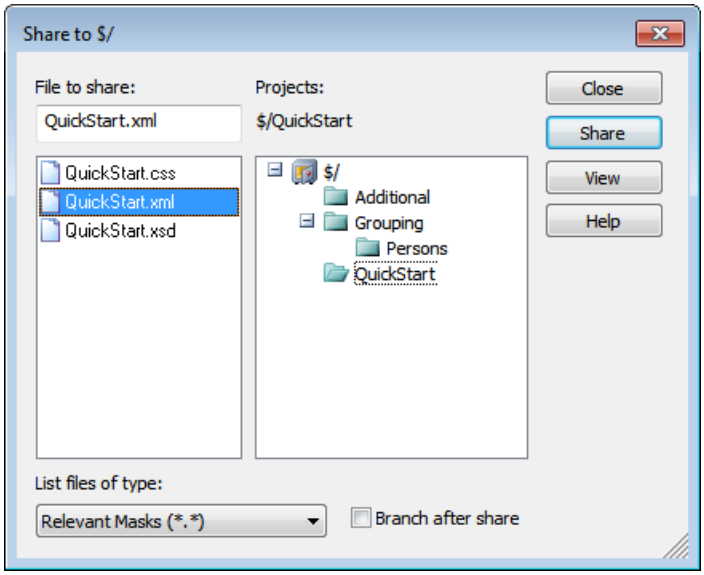
Sharing from source control

The **Share from Source Control** command is supported when the source control system being used supports shares. You can share a file, so that it is available at multiple local locations. A change made to one of these local files will be reflected in all the other "shared" versions.

In the application's Project window first select the project (*highlighted in the screenshot below*). Then click the **Share from Source Control**.

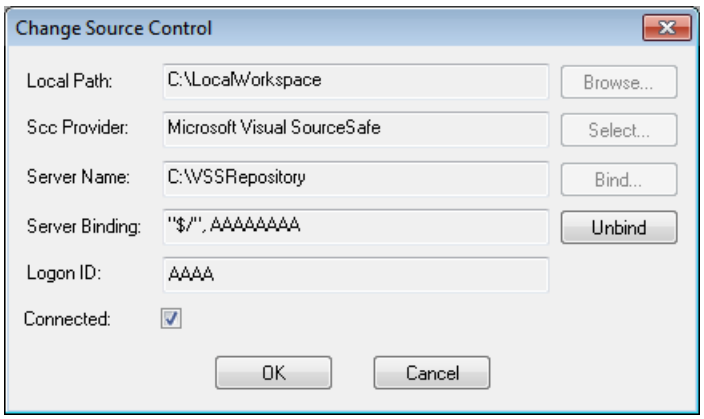


The Share To [Folder] dialog (*screenshot below*) pops up.



To select the files to share, first choose, in the project tree in the right-hand pane of the dialog (*see screenshot above*), the folder in which the files are. The files in the chosen folder are displayed in the left-hand pane. Select the file you wish to share (multiple files by pressing the **Ctrl** key and clicking the files you want to share). The selected file/s will be displayed in the *Files to Share* text box (*at top left*). The files disappear from the left hand pane. Click **Share** and then **Close** to copy the selected file/s to the local share folder. When you click **Close**, the files to share will be copied to the selected local location.

The share folder is noted in the name of the Share to [Folder] dialog. In the screenshot above it is the local folder (since the `$` sign is the folder in the repository to which the local folder is bound). You can see and set the share folder in the Change Source Control dialog (*screenshot below*, **Change Source Control**) by changing the local path and server binding.



For more details about sharing using your source control system, see the source control system's user documentation.

22.6.5 Changing Source Control

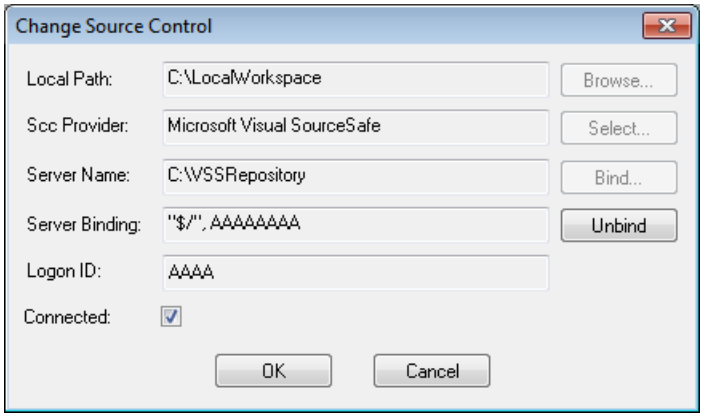
Source control settings can be changed via two commands in the **Project | Source Control** menu:

- **Source Control Manager**, which opens the source control system application and allows you to set up databases and configure bindings.
- **Change Source Control**, which pops up the Change Source Control dialog, in which you can change the source control system being used by the Altova application and the current binding. This dialog is described below.

The current binding is what the active application project will use to connect to the source control database. The current binding is correct when the application project file (`.spp` file) is in the local folder and the bound folder in the repository is where this project's files are stored. Typically the bound folder and its sub-structure will correspond with the local workspace folder and its sub-structure.

In the Change Source Control dialog (*screenshot below*), you can change the source control system (*SCC Provider*), the local folder (*Local Path*), and the repository binding (*Server Name* and *Server Binding*).

Only after undoing the current binding can the settings be changed. Undo the current binding with the **Undo** button. All the settings are now editable.



Change source control settings as follows:

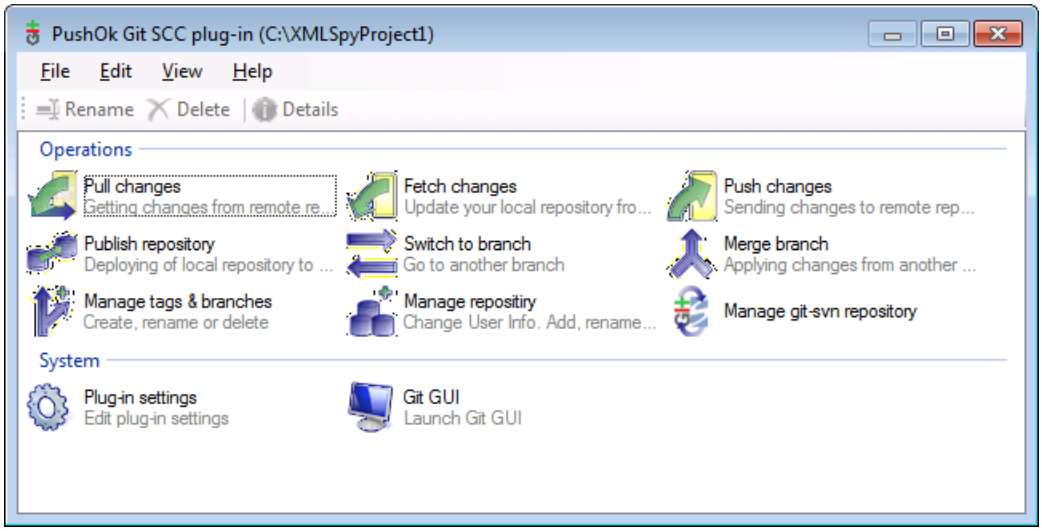
1. Use the **Browse** button to browse for the local folder and the **Select** button to select from among the installed source control systems.
2. After doing this you can bind the local folder to a repository database. Click the **Bind** button to do this. This pops up the connection dialog of your source control system.
3. If you have entered a *Logon ID*, this will be passed to the source control system; otherwise you might have to enter your logon details in the connection dialog.
4. Select the database in the repository that you wish to bind to this local folder. This setting might be spread over more than one dialog.
5. After the setting has been created, click **OK** in the Change Source Control dialog.

22.7 Source Control with Git

Support for Git as a source control system in XMLSpy is available through a third-party plug-in called **GIT SCC plug-in** (http://www.pushok.com/software/git.html).

> At the time when this documentation is written, the **GIT SCC plug-in** plug-in is available for experimental use. Registration with the plug-in publisher is required in order to use the plug-in.

The GIT SCC plug-in enables you to work with a Git repository using the commands available in the **Project | Source Control** menu of XMLSpy. Note that the commands in the **Project | Source Control** menu of XMLSpy are provided by the Microsoft Source Control Plug-in API (MSSCCI API), which uses a design philosophy different from Git. As a result, the plug-in essentially intermediates between "Visual Source Safe"-like functionality and Git functionality. On one hand, this means that a command such as **Get latest version** may not be applicable with Git. On the other hand, there are new Git-specific actions, which are available in the "Source Control Manager" dialog box provided by the plug-in (under the **Project | Source Control | Source Control Manager** menu of XMLSpy).



The Source Control Manager dialog box

Other commands that you will likely need to use frequently are available directly under the **Project | Source Control** menu.

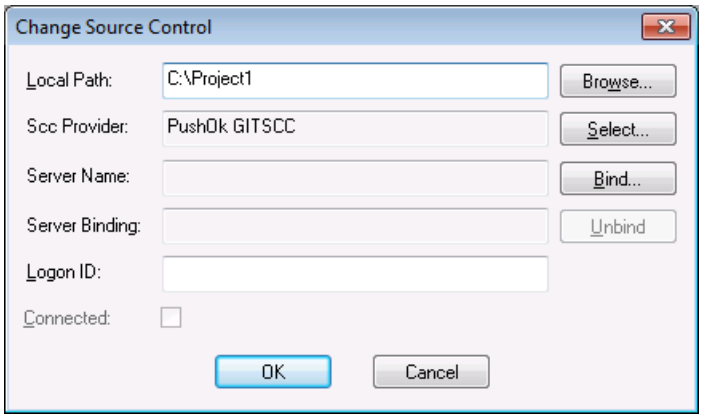
The following sections describe the initial configuration of the plug-in, as well as the basic workflow:

- Enabling Git Source Control with GIT SCC Plug-in
- Adding a Project to Git Source Control
- Cloning a Project from Git Source Control

22.7.1 Enabling Git Source Control with GIT SCC Plug-in

To enable Git source control with XMLSpy, the third-party **PushOK GIT SCC plug-in** must be installed, registered, and selected as source control provider, as follows:

1. Download the plug-in installation file from the publisher's website (http://www.pushok.com), run it, and follow the installation steps.
2. On the **Project** menu of XMLSpy, click **Change Source Control**, and make sure **PushOk GITSCC** is selected as source control provider. If you do not see **Push Ok GITSCC** in the list of providers, it is likely that the installation of the plug-in was not successful. In this case, check the publisher's documentation for a solution.



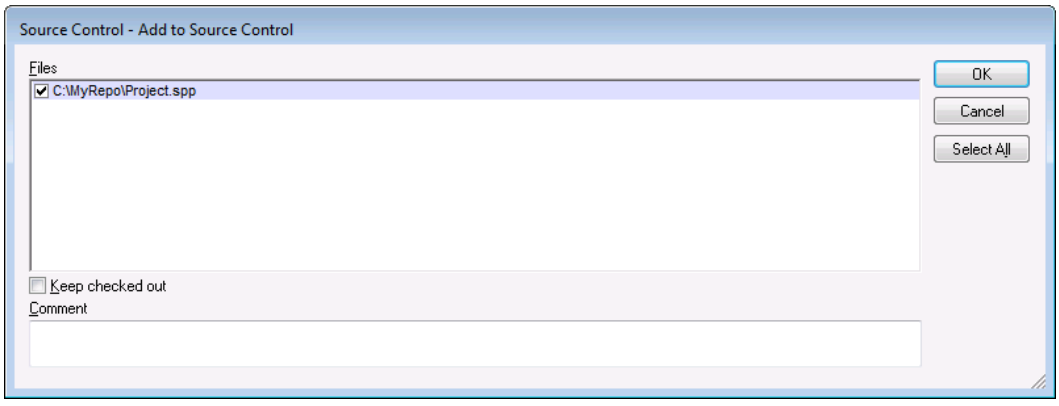
3. When a dialog box prompts you to register the plug-in, click **Registration** and follow the wizard steps to complete the registration process.

22.7.2 Adding a Project to Git Source Control

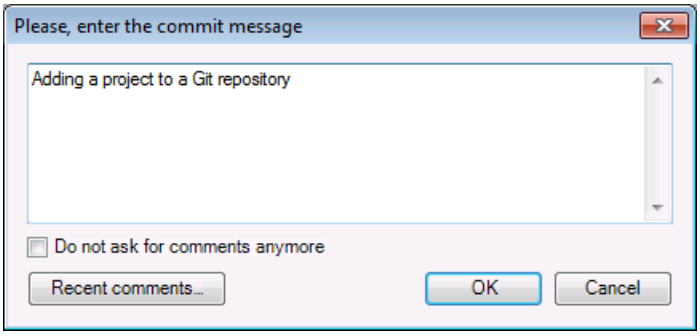
You can save XMLSpy projects as Git repositories. The structure of files or folders that you add to the project would then correspond to the structure of the Git repository.

To add a project to Git source control:

1. Make sure that **PushOK GIT SCC Plug-in** is set as source control provider (see Enabling Git Source Control with GIT SCC Plug-in).
2. Create a new project using the menu command **Project | Create Project**.
3. Save the project to a local folder, for example `C:\MyRepo\Project.spp`
4. On the **Project** menu, under **Source Control**, click **Add to Source Control**.

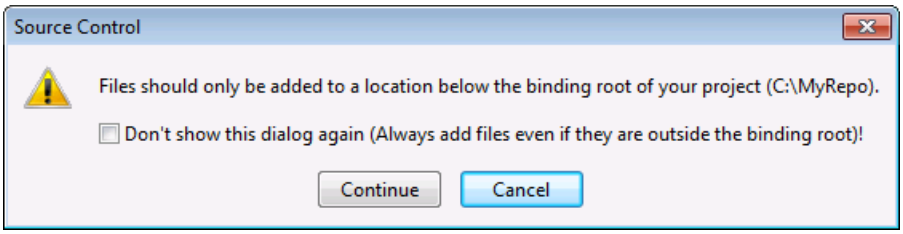


5. Click **OK**.



6. Enter the text of your commit message, and click **OK**.

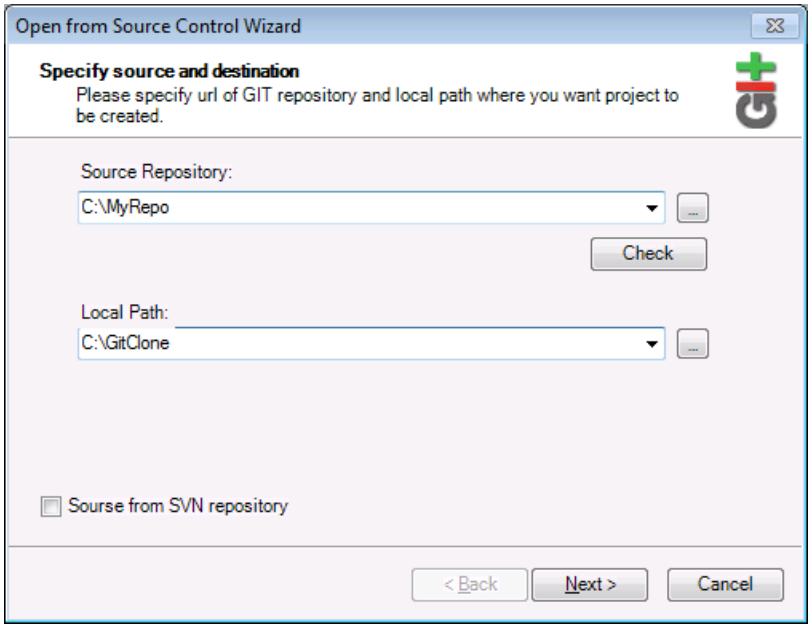
You can now start adding files and folders to your project. Note that all project files and folders must be under the root folder of the project. For example, if the project was created in the `C:\MyRepo` folder , then only files under `C:\MyRepo` should be added to the project. Otherwise, if you attempt to add to your project files that are outside the project root folder, a warning message is displayed:



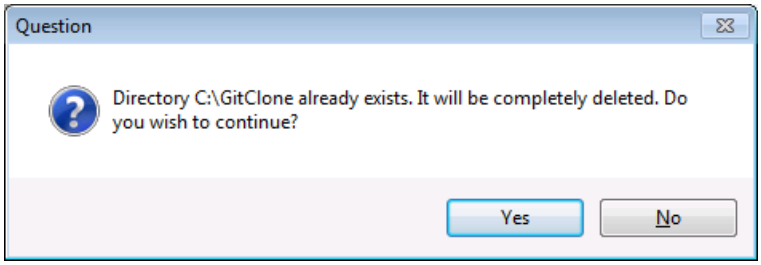
22.7.3 Cloning a Project from Git Source Control

Projects that have been previously added to Git source control (see Adding a Project to Git Source Control) can be opened from the Git repository as follows:

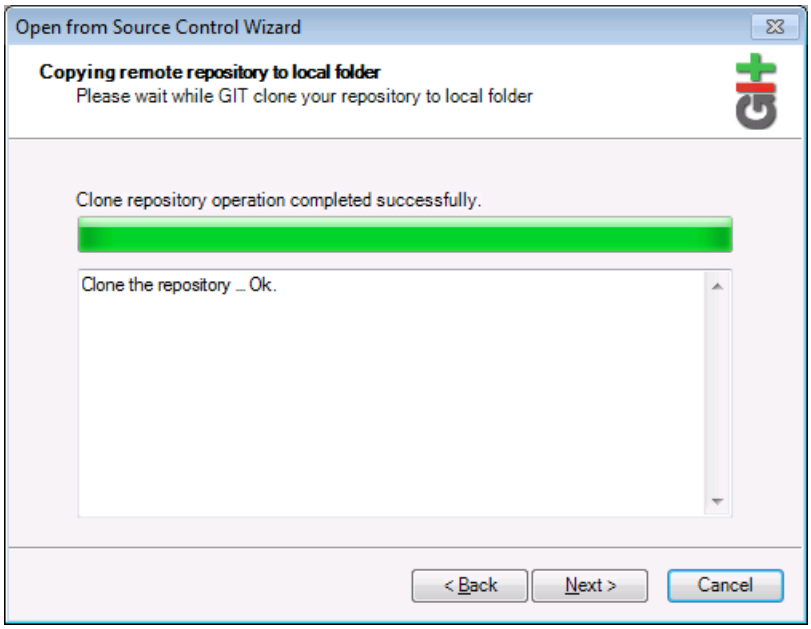
1. Make sure that **PushOK GIT SCC Plug- in** is set as source control provider (see Enabling Git Source Control with GIT SCC Plug- in).
2. On the **Project** menu, click **Source Control | Open from Source Control**.
3. Enter the path or the URL of the source repository. Click **Check** to verify the validity of the path or URL.



4. Under **Local Path**, enter the path to local folder where you want the project to be created, and click **Next**. If the local folder exists (even if it is empty), the following dialog box opens:



5. Click **Yes** to confirm, and then click **Next**.



6. Follow the remaining wizard steps, as required by your specific case.
7. When the wizard completes, a Browse dialog box appears, asking you to open the XMLSpy Project (*.spp) file. Select the project file to load the project contents into XMLSpy.

23 Visual Studio 内の XMLSpy

XMLSpy は Microsoft Visual Studio IDE のバージョン 2005/2008/2010/2012/2013/2015/2017 に統合することができます。この機能により、高度な XML 編集機能を、Visual Studio の高度な開発環境内で使用することができるようになります。

このセクションでは、以下のように構成されます：

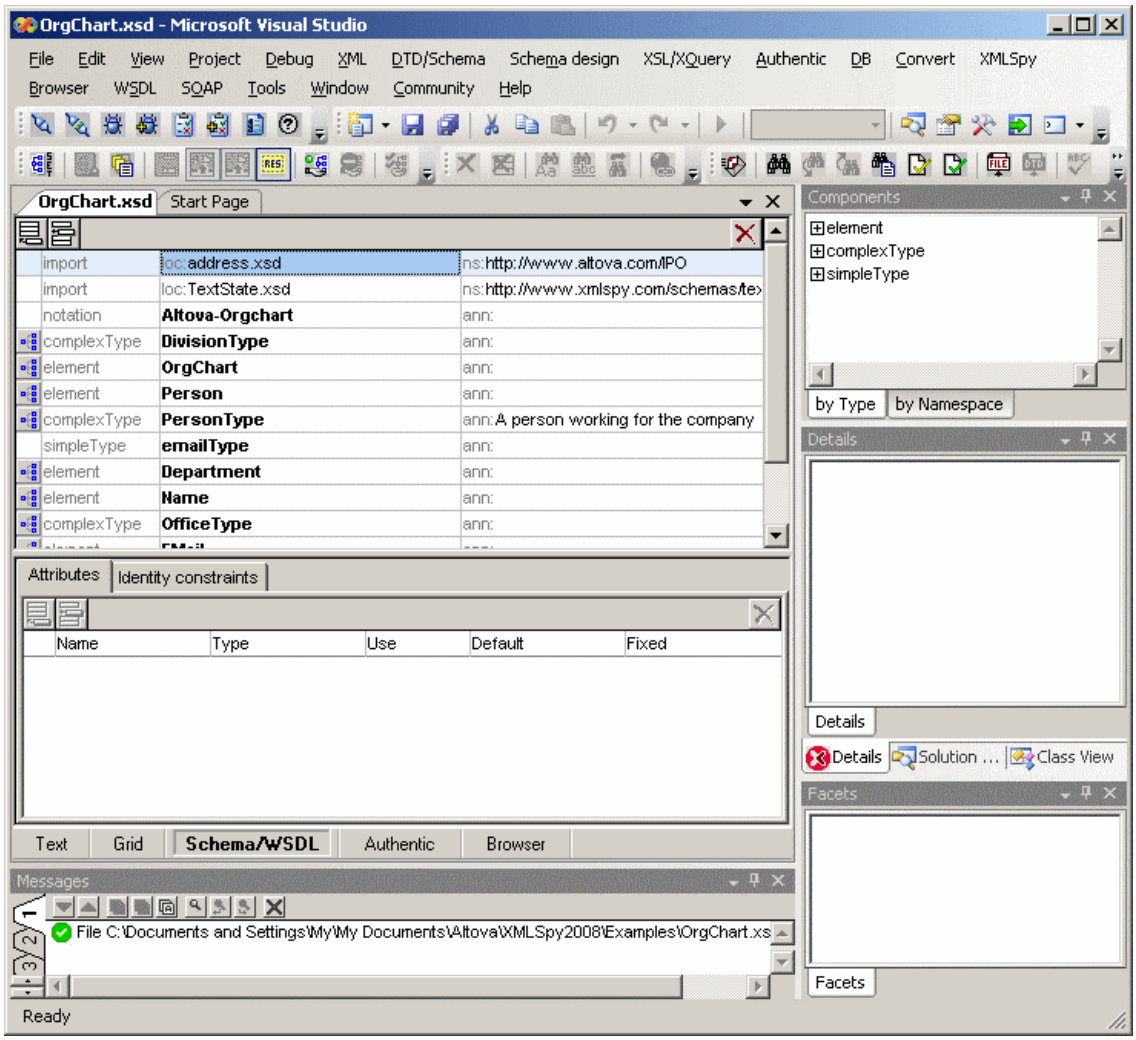
- 大まかなインストールの流れと、XMLSpy プラグインを Visual Studio へ統合
- Visual Studio バージョンとスタンドアロンバージョンの違い
- Visual Studio 内の XMLSpy デバッガー

メモ このセクションで使用されているスクリーンショットは Visual Studio バージョン 2005 のものです。他のバージョンを使用されている場合、実際に表示される画面と異なる場合があります。

23.1 XMLSpy プラグインのインストール

Visual Studio で使えるXMLSpy プラグインをインストールするには、以下の手順を踏みます：

1. Microsoft Visual Studio 2005/2008/2010/2012/2013/2015/2017 をインストール
2. XMLSpy (Enterprise または Professional Edition) をインストール
3. XMLSpy のMicrosoft Visual Studio Integration Package をダウンロードして実行。パッケージは www.altova.com にある <XMLSpy（Enterprise または Professional Edition）ダウンロードページにてご利用になれます。**注意**：Integration Package のバージョンは、お使いのXMLSpy のバージョンと同一のものでなければなりません。最新バージョンは 2017 です。



Integration Package をインストールすると、Visual Studio 環境からXMLSpy を使用することができるようになります。

プラグインを有効にする方法

インストールにてプラグインが自動的に有効化されなかった場合、以下の手順を踏んでください：

1. Visual Studio IDE 実行ファイルがインストールされているディレクトリ (例：`C:\Program Files\MS Visual Studio\Common7\IDE`) に移動します。
2. コマンドプロンプトにて以下のコマンドを入力します：`devenv.exe /setup`.
3. 処理が終わるのを待って、Visual Studio 内のアプリケーションの使用を開始してください。

メモ このセクションで使用されているスクリーンショットは Visual Studio バージョン 2005 のものです。他のバージョンを使用されている場合、実際に表示される画面と異なる場合があります。

23.2 XMLSpy スタンドアロンとの違い

このセクションではVisual Studio バージョンとスタンドアロンバージョンのXMLSpy の違いについて説明します。まずはVisual Studio バージョンでサポートされていない機能について、そしてスタンドアロンバージョンのVisual Studio 間で動作が異なる点について説明します。

- Visual Studio ではサポートされない機能
- Visual Studio で追加されたXMLSpy メニュー
- 入力ヘルパー (Visual Studio のツールウィンドウ)
- 異なるコマンドによる同等の機能
- Visual Studio コマンドとして使えるXMLSpy コマンド

Visual Studio ではサポートされない機能

Visual Studio では以下のXMLSpy の機能がご利用になれません：

- スクリプティング環境 **[ツール | XMLSpy オプション]** メニューオプションの **[スクリプト]** タブは現在サポートされていません。
- メニューオプションの **[ツール | ツール]** の **[ビュー]** タブにある、ブラウザービューを別のウィンドウに表示するオプションがサポートされていません。つまりテキストビューとテキストビューは常に同じウィンドウで表示されます。
- Authentic View の状態アイコンがサポートされていません。
- 全てのソース管理機能がサポートされていません。
- スタンドアロンバージョンでは **[ツール]** メニューオプションから利用できる全ての比較機能がサポートされていません。

Visual Studio で追加されたXMLSpy メニュー

以下のコマンドはVisual Studio バージョンのXMLSpy だけでご利用になれます：

- **表示 | XMLSpy ツールウィンドウ**
- **表示 | XMLSpy 表示**
- **XMLSpy** (グローバルリソースメニューアイテムが含まれます)
- ツール | **XMLSPY オプション**

入力ヘルパー (Visual Studio のツールウィンドウ)

XMLSpy の入力ヘルパーはVisual Studio のツールウィンドウとして使用することができます。入力ヘルパーに関して、以下の点にご注意してください。入力ヘルパーとXMLSpy のGUI に関しては、イントロダクションを参照ください。

- 入力ヘルパーウィンドウは開発環境内のどのような場所にでもドラッグすることができます。
- 入力ヘルパータブを右クリックすることで、インターフェースを更にカスタマイズすることができます。入力ヘルパーのオプションは：フローティング、ドッキング可能、タブ付きドキュメント、自動的に隠す、です。

異なるコマンドによる同等の機能

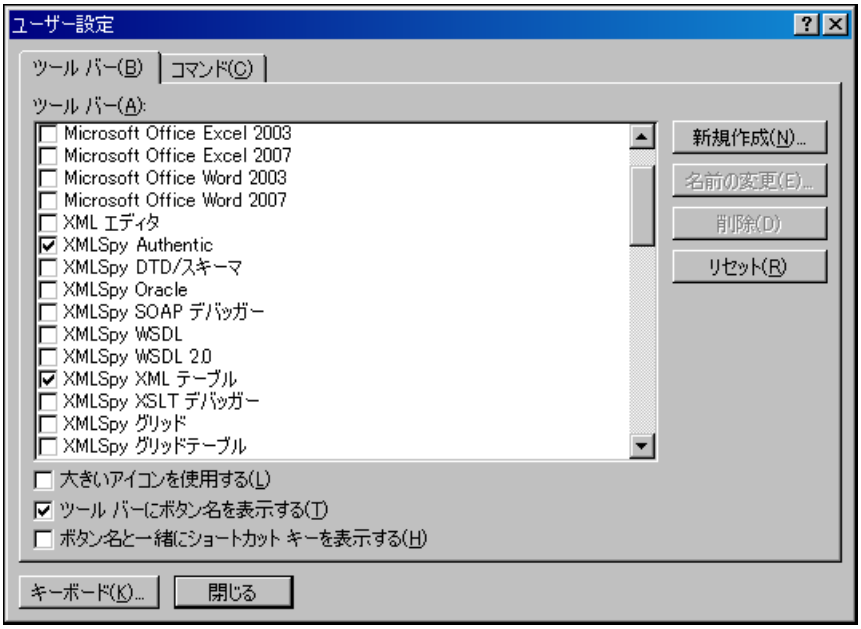
以下の表に示されるように、XMLSpy の機能によっては、Visual Studio にて異なるコマンド名で表示されます。

XMLSpy	*Visual Studio*	*機能*
ファイル	開く| URL にスイッチ	ファイル	開く| Web サイト	URL からファイルを開く
ファイル	名前を付けて保存	URL にスイッチ	ファイル	XMLSpy ファイルを URL に保存	ファイルを URL に保存

Visual Studio コマンドとして使える XMLSpy コマンド

いくつかの XMLSpy コマンドは Visual Studio コマンドとして Visual Studio GUI に表示されます：

- ***元に戻す、やり直し***：これらの Visual Studio コマンドは、Visual Studio 開発環境全てに対して適用されます。
- ***プロジェクト***: XMLSpy プロジェクトは Visual Studio プロジェクトとして扱われます。
- ***ツールバーのカスタマイズ、コマンドのカスタマイズ***：以下のスクリーンショットにある) ユーザー設定ダイアログ (**[ツール| ユーザー設定]**) 内のツールバーならびにコマンドタブには Visual Studio と XMLSpy のコマンドが含まれます。



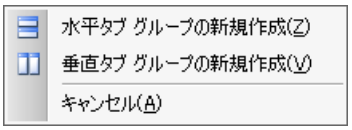
- ***表示***：**表示**メニュー以下の XMLSpy ***ツールウィンドウ***と XMLSpy ***ビュー***の2つのコマンドには、入力ヘルパーや他のサイドバー、編集ビュー間の切り替え、そして特定の入力ガイドの有効/無効を切り替えるオプションが含まれています。
- ***XMLSpy ヘルプ***:この XMLSpy メニューは Visual Studio **[ヘルプ]** メニューのサブメニューに表示されます。

メモ このセクションで使用されているスクリーンショットは Visual Studio バージョン 2005 のものです。他のバージョンを使用されている場合、実際に表示される画面と異なる場合があります。

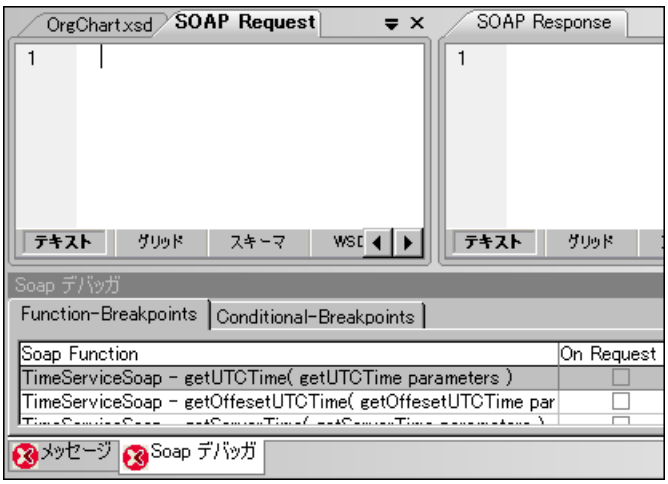
23.3 Visual Studio での XMLSpy デバッガー

XMLSpy にはXSLT/XQuery デバッガー（Enterprise および Professional Edition）と SOAP デバッガー（Enterprise Edition）が搭載されています。デバッガーでは（例えばXML、XSLT、XSLT 出力ファイルなど）複数のファイルが表示されますが、これらのファイルはVisual Studio にて1つにタブグループ化されます。Visual Studio では新たにタブグループを作成して、デバッグを見やすく、処理の経過をより簡単にたどることができます：

1. 単一のタブグループから分離したいタブをクリックします。そのタブを、現在アクティブなタブのどこかにドラッグします。この操作により、新たに作成するタブグループの種類を決定するポップアップメニューが表示されます。



2. **[垂直タブグループの新規作成]** を選択すると、選択されたタブのみが含まれるタブグループが新たに作成されます（以下のスクリーンショットを参照）。



メモ このセクションで使用されているスクリーンショットはVisual Studio バージョン2005 のものです。他のバージョンを使用されている場合、実際に表示される画面と異なる場合があります。

24 Eclipse 内の XMLSpy

Eclipse はオープンソースのフレームワークで、様々なアプリケーション開発環境をプラグインという形で統合します。

Eclipse 用の XMLSpy プラグインを使えば、Eclipse 4.4 / 4.5 / 4.6 プラットフォームから XMLSpy の機能にアクセスすることができるようになります。プラグインは Windows プラットフォームにてご利用になれます。このセクションでは Eclipse 用の XMLSpy プラグインのインストール方法と、XMLSpy パースペクティブならびに XMLSpy デバッガーパースペクティブの設定方法について説明します。これらの操作を行うと、XMLSpy GUI と XMLSpy メニューコマンドを Eclipse GUI から使用することができるようになります。

注意：スタンドアロンバージョンで使用することのできるソース管理機能は Eclipse バージョンにてご利用になれません。

24.1 Eclipse 用の XMLSpy プラグインをインストール

Eclipse 用のXMLSpy プラグインをインストールする前に、以下が既にインストールされていることを確認してください：

- XMLSpy Enterprise またはProfessional Edition。
- Eclipse に必要な、バージョン 5.0以降のJava SE 実行環境 (JRE 5.0)。。詳細についてはEclipse のウェブサイトを参照ください。XMLSpy のバージョン（32ビットまたは64ビット）と等しいバージョンのJRE（32ビットまたは64ビット）
- Eclipse プラットフォーム4.4 / 4.5 / 4.6 。XMLSpy のバージョン（32-ビットまたは64-ビット）と等しいバージョンのEclipse（32ビットまたは64ビット）をインストールしてください。

これらがインストールされた状態で、XMLSpy Integration Package に収められているEclipse 用のXMLSpy プラグインをインストールすることができます（下記を参照）。

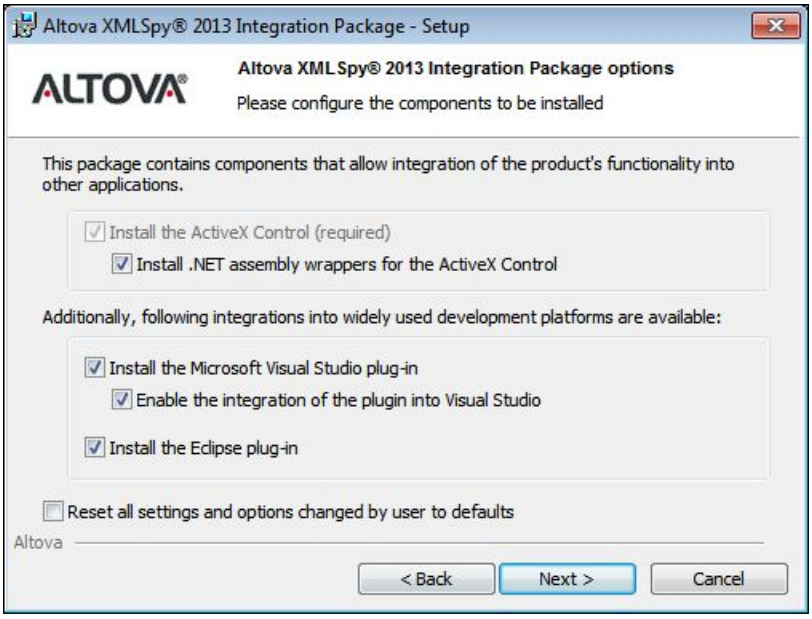
> Eclipse インストーラーを使用して、Eclipse 4.5 をインストールした場合、XMLSpy の 32 bit と 64 bit バージョンを同じマシンで作動することができます。これはEclipse インストーラーに搭載されており、Eclipse (32-ビットと64ビット)の両方のバージョンを手動でインストールする場合にはお適用されません。

メモ: 自動アップデートはWindows XP とWindows Server 2003 のためには操作されません。機能が正確に作動するためには、Windows 再頒布可能パッケージを手動でインストールする必要があります。

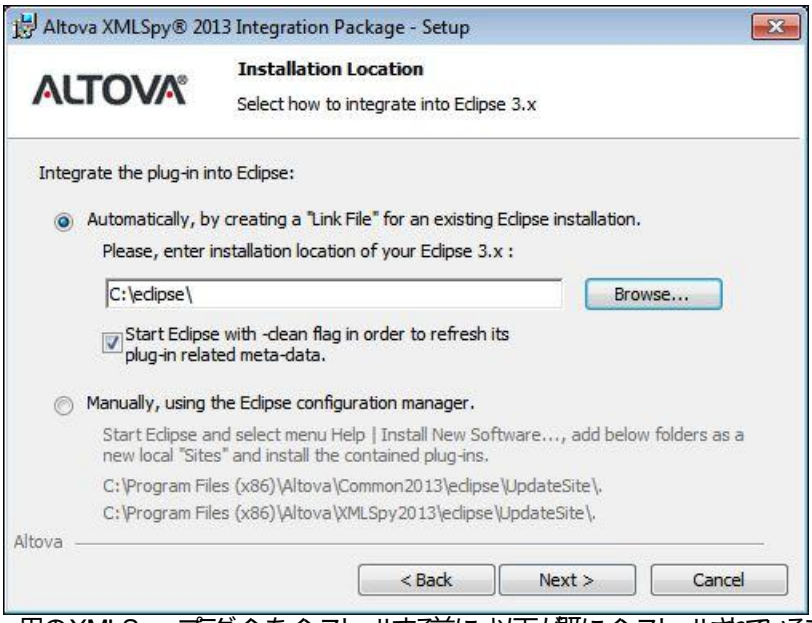
XMLSpy Integration Package

Eclipse 用のXMLSpy プラグインはXMLSpy Integration Package の中に含まれており、以下のようにXMLSpy Integration Package のインストーラーからインストールすることができます：

1. XMLSpy（Enterprise またはProfessional Edition）、JRE、そしてEclipse が既にインストールされていることを確認してください（上記参照）。
2. Altova ウェブサイトのコンポーネントダウンロードページから、XMLSpy Integration Package をダウンロード、インストールします。このインストールでは重要なステップが2つあり、それぞれステップ 3 と 4 で説明されます。
3. XMLSpy Integration Package のインストールでは、Eclipse 用のXMLSpy プラグインをインストールするかどうかを尋ねられます（以下のスクリーンショットを参照）。オプションにチェックを入れて、**次へ**をクリックしてください。



4. その次に表示される (以下のスクリーンショットの) ダイアログにて、インストールウィザードによりEclipse へXMLSpy プラグインの統合を行う (自動) か、ご自身の手で (Eclipse GUI から) Eclipse 用のXMLSpy プラグインを統合することができます (手動)。



Eclipse 用のXMLSpy プラグインをインストールする前に、以下が既にインストールされていることを確認してください:

- XMLSpy Enterprise または Professional Edition。
- Eclipse に必要な、バージョン 5.0 以降の Java SE 実行環境 (JRE 5.0)。。詳細については Eclipse のウェブサイトを参照ください。XMLSpy のバージョン (32ビットまたは64ビット) と等しいバージョンの JRE (32ビットまたは64ビット)
- Eclipse プラットフォーム4.4 / 4.5 / 4.6 。XMLSpy のバージョン (32-ビットまたは64-ビット) と等しいバージョンのEclipse (32ビットまたは64ビット) をインストールしてください。

これらがインストールされた状態で、XMLSpy Integration Package に収められているEclipse 用のXMLSpy プラ

グインをインストールすることができます（下記を参照）。

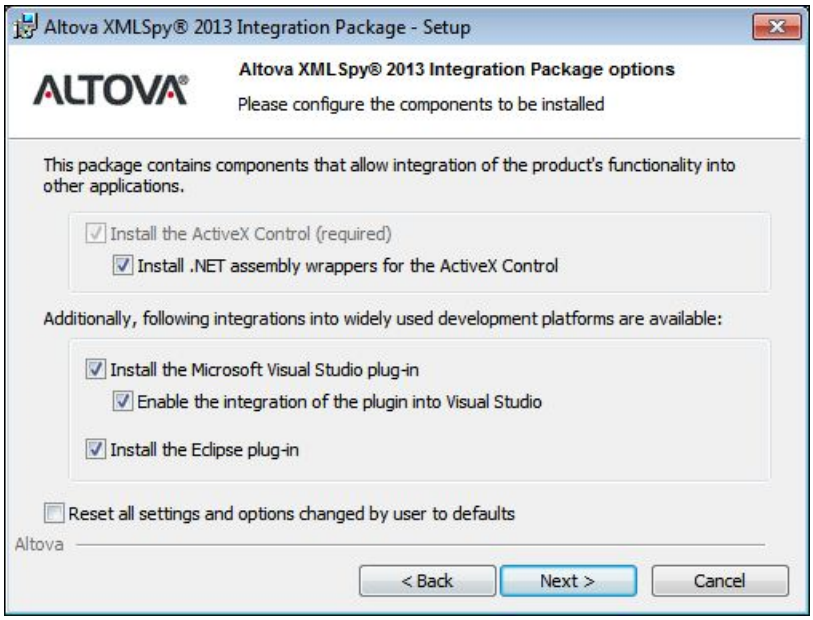
> Eclipse インストーラーを使用して、Eclipse 4.5 をインストールした場合、XMLSpy の 32 bit と 64 bit バージョンを同じマシンで作動することができます。これは Eclipse インストーラーに搭載されており、Eclipse (32-ビットと 64-ビット)の両方のバージョンを手動でインストールする場合にはお適用されません。

メモ: 自動アップデートは Windows XP と Windows Server 2003 のためには操作されません。機能が正確に作動するためには、Windows 再頒布可能パッケージを手動でインストールする必要があります。

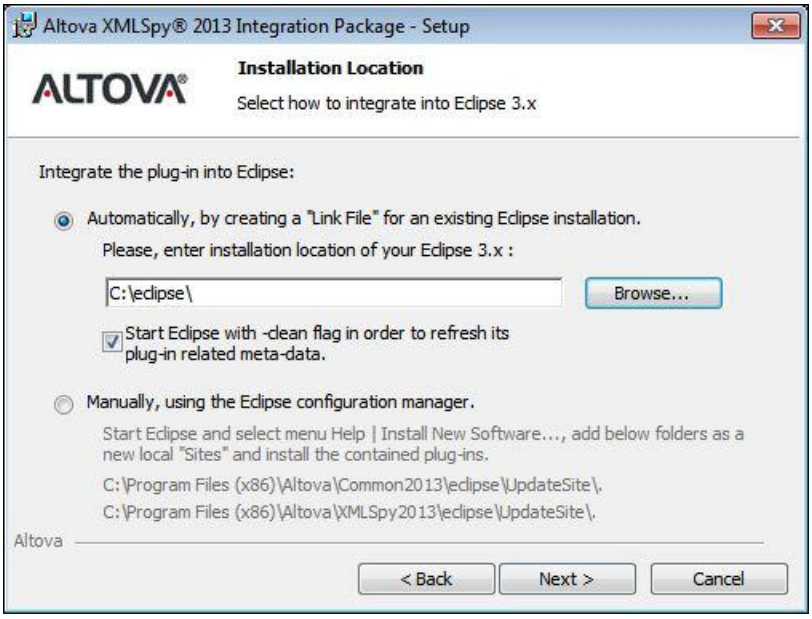
XMLSpy Integration Package

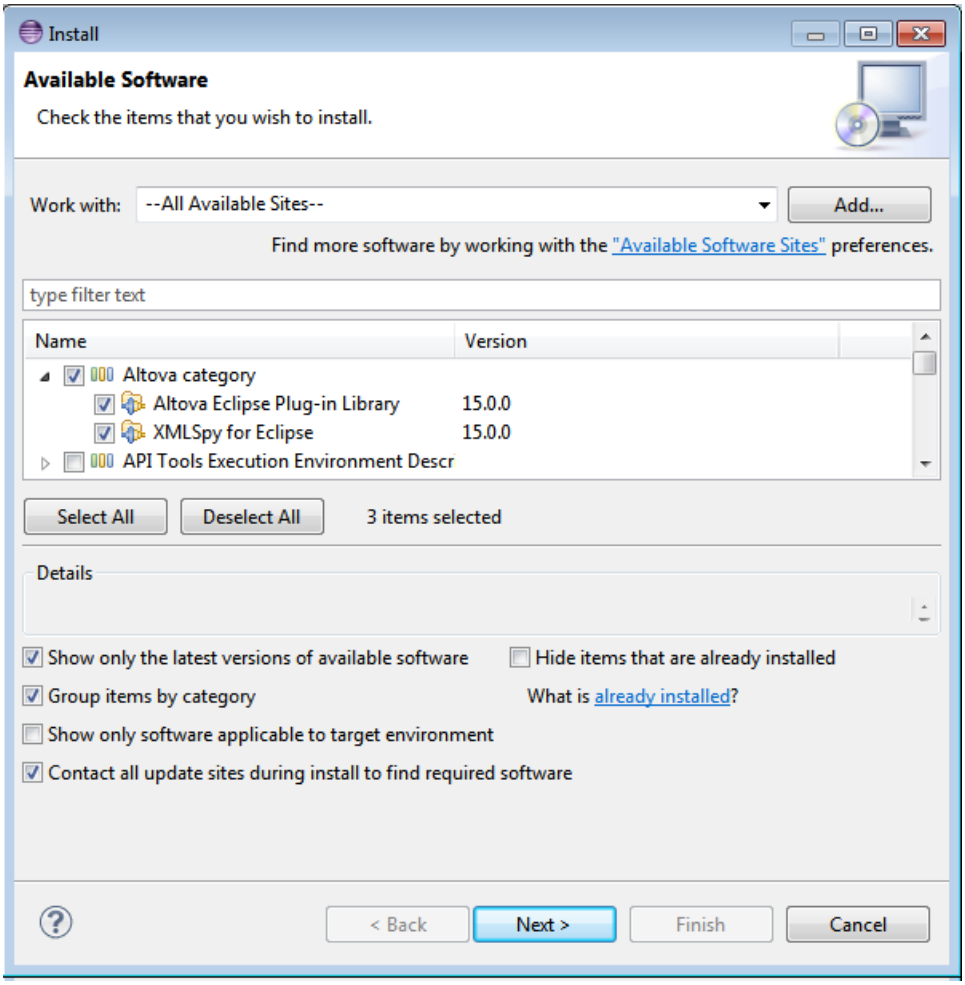
Eclipse 用の XMLSpy プラグインは XMLSpy Integration Package の中に含まれており、以下のように XMLSpy Integration Package のインストーラーからインストールすることができます：

1. XMLSpy（Enterprise または Professional Edition）、JRE、そして Eclipse が既にインストールされていることを確認してください（上記参照）。
2. Altova ウェブサイトのコンポーネントダウンロードページから、XMLSpy Integration Package をダウンロード、インストールします。このインストールでは重要なステップが 2 つあり、それぞれステップ 3 と 4 で説明されます。
3. XMLSpy Integration Package のインストールでは、Eclipse 用の XMLSpy プラグインをインストールするかどうかを尋ねられます（以下のスクリーンショットを参照）。オプションにチェックを入れて、**次へ**をクリックしてください。



4. その次に表示される（以下のスクリーンショットの）ダイアログにて、インストールウィザードにより Eclipse へ XMLSpy プラグインの統合を行う（自動）か、ご自身の手で（Eclipse GUI から）Eclipse 用の XMLSpy プラグインを統合することができます（手動）。





7. インストール詳細画面にてインストールするアイテムを確認します。**次へ**をクリックします。
8. 表示されるライセンスのレビュー画面にて、**使用条件の条項に同意します**を選択し、**完了**ボタンをクリックしてイン

ストールを行います (XMLSpy Enterprise または Professional Edition ライセンスに加えて XMLSpy プラグインのライセンスへ新たに同意する必要はありません)。

アイコンが表示されないなどの問題が発生した場合は、-clean フラグ付きで Eclipse を起動してみてください。

現在インストールされているバージョン

現在インストールされている XMLSpy プラグインのバージョンを確認するには、Eclipse のメニューオプションから「**ヘルプ | Eclipse について**」を選択します。そこから XMLSpy アイコンを選択してください。

24.2 Eclipse 内の XMLSpy エントリポイント

Eclipse 内にある以下のエントリポイントを使って XMLSpy の機能にアクセスすることができます：

- XMLSpy パースペクティブから XMLSpy の GUI 機能に Eclipse GUI からアクセスすることができます。
- XMLSpy ツールバーボタンから XMLSpy のヘルプならびにドキュメントの新規作成機能にアクセスすることができます。

XMLSpy パースペクティブ

Eclipse におけるパースペクティブとは、特定の機能に結び付けられた GUI ビューのことです。Eclipse 用の XMLSpy プラグインが Eclipse に統合されると、XMLSpy パースペクティブも自動的に作成されます。このパースペクティブは、編集ビュー、メニュー、入力ヘルパー、そしてサイドバーなど、XMLSpy の GUI 要素を含むパースペクティブとなります。

XMLSpy と関連付けられている種類のファイル（例えば .xml）が開かれた時、このファイルを XMLSpy パースペクティブで編集することができます。同様に、他の種類のファイルも Eclipse の別のパースペクティブで開くことができます。更に、アクティブなファイルに対しては、パースペクティブを変更することができ、異なる環境で同じファイルの編集を行うことができます。従って、パースペクティブには2つの利点があります：

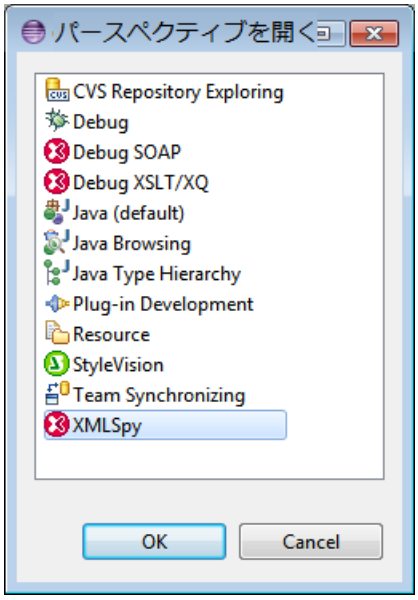
1. アクティブなファイルに対して、作業環境を素早く変更することができる。
2. 新たな開発環境を開くこと無くファイルの切り替えを行うことができる（パースペクティブに結び付けられた環境も有効になります）。

XMLSpy パースペクティブを使った作業には以下のようなものがあります：

- XMLSpy パースペクティブへの切り替え
- XMLSpy パースペクティブの環境設定
- XMLSpy パースペクティブのカスタマイズ

XMLSpy パースペクティブへの切り替え

Eclipse にて、メニューコマンドから「**ウィンドウ| パースペクティブを開く| その他**」を選択します。表示されるダイアログにて XMLSpy を選択し、**OK** をクリックします（以下のスクリーンショットを参照）。



空のドキュメント、またはアクティブなドキュメントに対してXMLSpy パースペクティブが開かれます。この方法でユーザーはパースペクティブの切り替えを行うことができます。目的のパースペクティブを**パースペクティブを開く**サブメニュー の**その他**アイテムの上）に表示させることで、パースペクティブにより素早くアクセスすることができます。この設定はカスタマイズダイアログにて行うことができます。

ファイルが開かれた時、またはアクティブになったときにパースペクティブを自動的に切り替えることもできます。そのファイルが最初に開かれたときには、ファイルの種類に関連付けられているアプリケーションのパースペクティブが自動的に開かれ、このファイルの種類に対して、このパースペクティブをデフォルトのパースペクティブとして割り当てるか尋ねられます。この種類のファイルを開くたびに尋ねられたくない場合は、*今後このメッセージを表示しない*にチェックを入れ**OK** をクリックします。
。この種類のファイルを開くたびに尋ねられたくない場合は、*今後このメッセージを表示しない*にチェックを入れ**OK** をクリックします。

XMLSpy パースペクティブの環境設定

パースペクティブの設定には、() 関連付けられた種類のファイルが開かれたときに、自動的にパースペクティブを切り替える、(ii) 各 XMLSpy ツールバーを含めるオプションが含まれます。設定ダイアログにアクセスするには、**ウィンドウ| 設定**」メニューコマンドを選択します。左のペインにあるリストからXMLSpy を選択し、必要な設定を選択します。**OK** ボタンをクリックして完了します。

XMLSpy パースペクティブのカスタマイズ

カスタマイズオプションを使うことで、ショートカットやコマンドをパースペクティブに加えることができます。（以下のスクリーンショットにある）パースペクティブのカスタマイズダイアログにアクセスするには、（XMLSpy パースペクティブの場合は）パースペクティブをアクティブにして、メニューコマンドから**ウィンドウ| パースペクティブのカスタマイズ**」を選択します。

ツール・バー可視性タブとメニュー可視性タブでは、表示するツールバーとメニューを指定することができます。コマンド・グループ可用性タブでは、コマンドグループを親メニューやツールバーへ追加することができます。コマンドグループを有効にするには、対応するチェックボックスにチェックを入れてください。カスタマイズ パースペクティブダイアログのショートカットタブでは、サブメニュ

ーのショートカットをセットすることができます。サブメニューコンボボックスから目的のサブメニューを選択し、ショートカットのカテゴリを選択し、パースペクティブに追加したいショートカットにチェックを入れます。**OK** をクリックしてカスタマイズを完了します。

XMLSpy ツールバーボタン

ツールバーには XMLSpy-に関連する2つのボタンが追加されます (以下のスクリーンショットを参照)。

これらのボタンは、() XMLSpy ヘルプを開くと (i) 新規 XMLSpy ドキュメントの作成のために使用されます。。

24.3 XMLSpy のデバッガーパースペクティブ

Enterprise Edition の XMLSpy には2つのデバッガー (XSLT/XQuery ならびに SOAP) が搭載されており、Professional Edition の XMLSpy には1つのデバッガー (XSLT/XQuery) が搭載されています。これらデバッガーのパースペクティブも、現在インストールされている XMLSpy のエディションに応じて Eclipse で利用することができます。

デバッガーのパースペクティブに切り替えるには、メニューコマンドから **[ウィンドウ| パースペクティブを開く| その他]** を選択します。表示されるダイアログ (以下のスクリーンショット) にて、デバッガーを選択 (例えば、XSLT/XQをデバッグ) し、**「OK」** をクリックします。

入力ヘルパーまたはアクティブなドキュメントに対して選択されたデバッガーのパースペクティブが適用されます。この方法でユーザーはパースペクティブの切り替えを行うことができます。目的のパースペクティブを **[パースペクティブを開く]** サブメニュー の **[その他]** アイテムの上) に表示させることで、パースペクティブにより素早くアクセスすることができます。この設定はカスタマイズダイアログにて行うことができます。

デバッガーを使用する方法については、このドキュメンテーション内の対応するセクションを参照ください: XSLT とXQuery、WSDL とSOAP。

25 Code Generator

XMLSpy includes a built-in code generator which can generate Java, C++ or C# class files from XML schemas. The generated code consists of strongly-typed schema wrapper libraries that enable you to create software applications that process XML data. The schema wrapper libraries enable you to work with XML data in an abstract way, without too much concern for the underlying XML Application Program Interface (API), such as MSXML, Apache Xerces, Microsoft System.Xml, or Java Application for XML Processing (JAXP).

25.1 Introduction to code generator

The generated code supports the following operations:

- Read XML files into a Document Object Model (DOM) in-memory representation
- Write XML files from a DOM representation back to a system file
- Convert strings to XML DOM trees and vice versa.

The generated code is expressed in C++, Java or C# programming languages.

Target Language	C++	C#	Java
Development environments	Microsoft Visual Studio 2010 Microsoft Visual Studio 2008 Microsoft Visual Studio 2005	Microsoft Visual Studio 2010 Microsoft Visual Studio 2008 Microsoft Visual Studio 2005	Eclipse 3.4 or later Apache Ant (build.xml file)
XML DOM implementations	MSXML 6.0 or Apache Xerces 2.6 or later	System.Xml	JAXP

C++

You can configure whether the C++ generated output should use MSXML 6.0 or Apache Xerces 2.6 or later. XMLSpy generates complete project (.vcproj) and solution (.sln) files for Visual Studio 2005/ 2008/ 2010. The generated code optionally supports MFC.

Note: When building C++ code for Visual Studio 2005/2008/2010 and using a Xerces library precompiled for Visual C++, a compiler setting has to be changed in all projects of the solution:

1. Select all projects in the Solution Explorer.
2. On the **Project** menu, click **Properties**.
3. Click **Configuration Properties | C/C++ | Language**.
4. In the list of configurations, select *All Configurations*.
5. Change **Treat wchar_t as Built-in Type** to **No (/ Zc:wchar_t-)**

C#

The generated C# code uses the .NET XML classes (System.Xml) and can be used from any .NET capable programming language, such as VB.NET, Managed C++, or J#. Project files can be generated for Visual Studio 2005, 2008, and 2010.

Java

The generated Java output is written against the industry-standard Java API for XML Processing (JAXP) and includes an Ant build file and project files for Eclipse. Java 7 or higher is supported.

Generated output

The designated destination folder for the generated code includes all the libraries and files required to manipulate XML files programmatically, namely:

- Standard Altova libraries
- Schema wrapper libraries

- An empty test application with sample source code. The test application skeleton is a compilable application that calls an empty Example() method. You can add your test code into this method for easy and quick testing of your new generated library.

Code generator templates

Output code is completely customizable via a simple yet powerful template language (SPL, from Spy Programming Language) which gives full control in mapping XML Schema built-in data-types to the primitive datatypes of a particular programming language. SPL allows you to easily replace the underlying parsing and validating engine, customize code according to your company's writing conventions, or use different base libraries such as the Microsoft Foundation Classes (MFC) and the Standard Template Library (STL).

With SPL, you can build your own templates to automate the generation of virtually any other format, for example: EJB's, WSDL files, SQL scripts, ASP or WML code.

XMLSpy's schema editor is well suited as a software modeling and prototyping tool, allowing XML applications to be rapidly prototyped at a high level in XML Schema and then automatically generated. Changes to an application's XML Schema content model can be immediately reconciled with a software implementation simply by re-running the code generator.

25.2 What's new ...

Version 2014

- Removal of compatibility mode option for code generation

Version 2011

- Contains bug fixes and enhancements

Version 2010 R3

- Support for Microsoft Visual Studio 2010
- Support for MSXML 6.0 in generated C++ code
- Support for 64-bit targets for C++ and C# projects

Version 2010 R2

- Code generation for C++ for the Linux platform

Version 2010

- Enumeration facets from XML schemas are now available as symbolic constants in the generated classes (using 2007r3 templates)

Version 2009 sp1

- Apache Xerces version 3.x support added (older versions starting from Xerces 2.6.x are still supported)

Version 2009

- No new updates

Version 2008 R2

- Support for generation of Visual Studio 2008 project files for C# and C++ has been added

Version 2008

The new 2007 R3-style SPL templates have been further enhanced:

- It is now possible to remove single elements
- Access to schema metadata (e.g. element names, facets, enumerations, occurrence, etc.) is provided

- Complex types derived by extension are now generated as derived classes

Version 2007 R3

Code Generator has been redesigned for version 2007 release 3 to simplify usage of the generated code, reduce code volume and increase performance.

- Handling of XML documents and nodes with explicit ownership, to avoid memory leaks and to enable multi-threading
- New syntax to avoid name collisions
- New data types for simpler usage and higher performance (native types where possible, new null handling, ...)
- Attributes are no longer generated as collections
- Simple element content is now also treated like a special attribute, for consistency
- New internal object model (important for customized SPL templates)
- Compatibility mode to generate code in the style of older releases

25.3 Generating Code from XML Schemas or DTDs

With XMLSpy code generator, you can generate C#, C++, or Java program code from XML schemas or DTDs. The generated schema wrapper libraries can then be integrated in your custom application in order to read, modify, or write XML documents programmatically.

Generating program code

1. Open the schema for which you want to generate source code.
2. Select the menu item **DTD/Schema | Generate Program Code**.
3. In the Choose Template pane of the dialog that pops up, set the code generation options (see Code Generator Options).
4. Click **OK**. The *Browse for Folder* dialog appears.
5. Select the target folder and click **OK**.
6. You are prompted to open the newly created project in Microsoft Visual Studio. Click **Yes**. If Java code is produced, you are prompted to open the corresponding output directory.

When XMLSpy generates code from an XML Schema or DTD, the following libraries are created:

C++ or C#	Java	Purpose
Altova	`com.altova`	Base library containing common runtime support, identical for every schema.
AltovaXML	`com.altova.xml`	Base library containing runtime support for XML, identical for every schema.
YourSchema	*`com.YourSchema`*	A library containing declarations generated from the input schema, named as the schema file or DTD. This library is a DOM (W3C Document Object Model) wrapper that allows you to read, modify and create XML documents easily and safely. All data is held inside the DOM, and there are methods for extracting data from the DOM, and to update and create data into the DOM. The generated C++ code supports either Microsoft MSXML or Apache Xerces 2.6 or higher, depending on code generation settings. The syntax for using the generated code is identical for both DOM implementations. The generated C# code uses the .NET standard System.XML library as the underlying DOM implementation. The generated Java code uses JAXP (Java API for XML Processing) as the underlying DOM interface.
YourSchemaTest	*`com.YourSchemaTest`*	The generated code also includes a test application skeleton named after your schema (for example, *YourSchemaTest*). This is a compilable application that calls an empty Example() method. You can add your test code into this method for easy and quick testing of your new generated library.

While prototyping an application from a frequently changing XML schema, you may need to

frequently generate code to the same directory, so that the schema changes are immediately reflected in the code. Note that the generated test application and the Altova libraries are overwritten every time when you generate code into the same target directory. Therefore do not add code to the generated test application. Instead, integrate the Altova libraries into your project (see Integrating Schema Wrapper Libraries).

Name generation and namespaces

XMLSpy generates classes corresponding to all declared elements or complex types which redefine any complex type in your XML Schema, preserving the class derivation as defined by extensions of complex types in your XML Schema. In the case of complex schemas which import schema components from multiple namespaces, XMLSpy preserves this information by generating the appropriate C# or C++ namespaces or Java packages.

Generally, the code generator tries to preserve the names for generated namespaces, classes and members from the original XML Schema. Characters that are not valid in identifiers in the target language are replaced by a "_". Names that would collide with other names or reserved words are made unique by appending a number. Name generation can be influenced by changing the default settings in the SPL (Spy Programming Language) template.

The namespaces from the XML Schema are converted to packages in Java or namespaces in C# or C++ code, using the namespace prefix from the schema as code namespace. The complete library is enclosed in a package or namespace derived from the schema file name, so you can use multiple generated libraries in one program without name conflicts.

Data Types

XML Schema has a more elaborate data type model than Java, C# or C++. Code Generator converts the built-in XML Schema types to language-specific primitive types, or to classes delivered with the Altova library. Complex types and derived types defined in the schema are converted to classes in the generated library. Enumeration facets from simple types are converted to symbolic constants.

The mapping of simple types can be configured in the SPL template, see SPL (Spy Programming Language).

If your XML instance files use schema types related to time and duration, these are converted to Altova native classes in the generated code. For information about the Altova library classes, see:

- Reference to Generated Classes (C++)
- Reference to Generated Classes (C#)
- Reference to Generated Classes (Java)

For information about type conversion and other details applicable to each language, see:

- About Schema Wrapper Libraries (C++)
- About Schema Wrapper Libraries (C#)
- About Schema Wrapper Libraries (Java)

Memory management

A DOM tree is comprised of nodes, which are always owned by a specific DOM document - even if the node is not currently part of the document's content. All generated classes are references to the DOM nodes they represent, not values. This means that assigning an instance of a generated class does not copy the value, it only creates an additional reference to the same data.

XML Schema support

The following XML Schema constructs are translated into code:

a) XML namespaces

b) Simple types:

- Built-in XML schema types
- Simple types derived by extension
- Simple types derived by restriction
- Facets
- Enumerations
- Patterns

c) Complex types:

- Built-in anyType node
- User-defined complex types
- Derived by extension: Mapped to derived classes
- Derived by restriction
- Complex content
- Simple content
- Mixed content

The following advanced XML Schema features are not supported (or not fully supported) in generated wrapper classes:

- Wildcards: `xs:any` and `xs:anyAttribute`
- Content models (sequence, choice, all). Top-level compositor is available in SPL (Spy Programming Language), but is not enforced by generated classes.
- Default and fixed values for attributes. These are available in SPL (Spy Programming Language), but are not set or enforced by generated classes.
- The attributes `xsi:type`, abstract types. When you need to write the xsi:type attribute, use the `SetXsiType()` method of the generated classes.
- Union types: not all combinations are supported.
- Substitution groups are partially supported (resolved like "choice").
- Attribute `nillable="true"` and `xsi:nil`
- Uniqueness constraints
- Identity constraints (`key` and `keyref`)

25.3.1 About Schema Wrapper Libraries (C++)

Character Types

The generated C++ code can be compiled with or without Unicode support. Depending on this setting, the types `string_type` and `tstring` will both be defined as `std::string` or `std::wstring`, consisting of narrow or wide characters. To use Unicode characters in your XML file that are not representable with the current 8-bit character set, Unicode support must be enabled. Pay special attention to the `_T()` macros. This macro ensures that string constants are stored correctly, whether you're compiling for Unicode or non-Unicode programs.

Data Types

The default mapping of XML Schema types to C++ data types is:

XML Schema	C++	Remarks
`xs:string`	`string_type`	string_type is defined as std::string or std:wstring
`xs:boolean`	`bool`	
`xs:decimal`	`double`	C++ does not have a decimal type, so double is used.
`xs:float, xs:double`	`double`	
`xs:integer`	`__int64`	xs:integer has unlimited range, mapped to _int64 for efficiency reasons.
`xs:nonNegativeInteger`	`unsigned __int64`	see above
`xs:int`	`int`	
`xs:unsignedInt`	`unsigned int`	
`xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth`	`altova::DateTime`	
`xs:duration`	`altova::Duration`	
`xs:hexBinary and xs:base64Binary`	`std::vector<unsigned char>`	Encoding and decoding of binary data is done automatically.
`xs:anySimpleType`	`string_type`	

All XML Schema types not contained in this list are derived types, and mapped to the same C++ type as their respective base type.

Generated Classes

For each type in the schema, a class is generated that contains a member for each attribute and element of the type. The members are named the same as the attributes or elements in the original schema (in case of possible collisions, a number is appended). For simple types, assignment and conversion operators are generated. For simple types with enumeration facets, the methods `GetEnumerationValue()` and `SetEnumerationValue(int)` can be used together with generated constants for each enumeration value. In addition, the method `StaticInfo()`

allows accessing schema information as one of the following types:

altova::meta::SimpleType
altova::meta::ComplexType

Classes generated from complex types include the method `SetXsiType()`, which enables you to set the `xsi:type` attribute of the type. This method is useful when you want to create XML instance elements of a derived type.

In addition to the classes for the types declared in the XML Schema, a document class (identified with "CDoc" below) is generated. It contains all possible root elements as members, and various other methods. For more information about the class, see `[YourSchema]::[CDoc]` .

Note: The actual class name depends on the name of the .xsd schema.

For each member attribute or element of a schema type, a new class is generated. For more information about such classes, see:

`[YourSchema]::MemberAttribute`
`[YourSchema]::MemberElement`

Note: The actual class names depend on the name of the schema attribute or element.

See also Example: Using the Schema Wrapper Libraries.

Error Handling

Errors are reported by exceptions. The following exception classes are defined in the namespace `altova`:

Class	Base Class	Description
`Error`	`std::logic_error`	Internal program logic error (independent of input data).
`Exception`	`std::runtime_error`	Base class for runtime errors.
`InvalidArgumentsException`	`Exception`	A method was called with invalid argument values.
`ConversionException`	`Exception`	Exception thrown when a type conversion fails.
`StringParseException`	`ConversionException`	A value in the lexical space cannot be converted to value space.
`ValueNotRepresentableException`	`ConversionException`	A value in the value space cannot be converted to lexical space.
`OutOfRangeException`	`ConversionException`	A source value cannot be represented in target domain.
`InvalidOperationException`	`Exception`	An operation was attempted that is not valid in the given context.
`DataSourceUnavailableEx`	`Exception`	A problem occurred while

ception		loading an XML instance.
DataTargetUnavailableException	Exception	A problem occurred while saving an XML instance.

All exception classes contain a message text and a pointer to a possible inner exception.

Method	Purpose
string_type message()	Returns a textual description of the exception.
std::exception inner()	Returns the exception that caused this exception, if available, or NULL.

Accessing schema information

The generated library allows accessing static schema information via the following classes. All methods are declared as const. The methods that return one of the metadata classes return a NULL object if the respective property does not exist.

altova::meta::Attribute
altova::meta::ComplexType
altova::meta::Element
altova::meta::SimpleType

25.3.2 About Schema Wrapper Libraries (C#)

The default mapping of XML Schema types to C# data types is as follows.

XML Schema	C#	Remarks
xs:string	string	
xs:boolean	bool	
xs:decimal	decimal	xs:decimal has unlimited range and precision, mapped to decimal for efficiency reasons.
xs:float, xs:double	double	
xs:long	long	
xs:unsignedLong	ulong	
xs:int	int	
xs:unsignedInt	uint	
xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth	Altova.Types.DateTime	
xs:duration	Altova.Types.Duration	
xs:hexBinary and xs:base64Binary	byte[]	Encoding and decoding of binary data is done automatically.
xs:anySimpleType	string	

All XML Schema types not contained in this list are derived types, and mapped to the same C# type as their respective base type.

Generated Classes

For each type in the schema, a class is generated that contains a member for each attribute and element of the type. The members are named the same as the attributes or elements in the original schema (in case of possible collisions, a number is appended). For simple types, assignment and conversion operators are generated. For simple types with enumeration facets, the methods `GetEnumerationValue()` and `SetEnumerationValue(int)` can be used together with generated constants for each enumeration value. In addition, the method `StaticInfo()` allows accessing schema information as one of the following types:

`Altova.Xml.Meta.SimpleType`
`Altova.Xml.Meta.ComplexType`

Classes generated from complex types include the method `SetXsiType()`, which enables you to set the `xsi:type` attribute of the type. This method is useful when you want to create XML instance elements of a derived type.

In addition to the classes for the types declared in the XML Schema, a document class (identified with "Doc" below) is generated. It contains all possible root elements as members, and various other methods. For more information about the class, see `[YourSchema].[Doc]`.

Note: The actual class name depends on the name of the .xsd schema.

For each member attribute or element of a schema type, a new class is generated. For more information about such classes, see:

`[YourSchemaType].MemberAttribute`
`[YourSchemaType].MemberElement`

Note: The actual class names depend on the name of the schema attribute or element.

Error Handling

Errors are reported by exceptions. The following exception classes are defined in the namespace Altova:

Class	Base Class	Description
`ConversionException`	`Exception`	Exception thrown when a type conversion fails
`StringParseException`	`ConversionException`	A value in the lexical space cannot be converted to value space.
`DataSourceUnavailableException`	`System.Exception`	A problem occurred while loading an XML instance.
`DataTargetUnavailableException`	`System.Exception`	A problem occurred while saving an XML instance.

In addition, the following .NET exceptions are commonly used:

Class	Description
System.Exception	Base class for runtime errors
System.ArgumentException	A method was called with invalid argument values, or a type conversion failed.
System.FormatException	A value in the lexical space cannot be converted to value space.
System.InvalidCastException	A value cannot be converted to another type.
System.OverflowException	A source value cannot be represented in target domain.

Accessing schema information

The generated library allows accessing static schema information via the following classes:

Altova.Xml.Meta.Attribute
Altova.Xml.Meta.ComplexType
Altova.Xml.Meta.Element
Altova.Xml.Meta.SimpleType

The properties that return one of the metadata classes return null if the respective property does not exist.

25.3.3 About Schema Wrapper Libraries (Java)

The default mapping of XML Schema types to Java data types is as follows:

XML Schema	Java	Remarks
xs:string	String	
xs:boolean	boolean	
xs:decimal	java.math.BigDecimal	
xs:float, xs:double	double	
xs:integer	java.math.BigInteger	
xs:long	long	
xs:unsignedLong	java.math.BigInteger	Java does not have unsigned types.
xs:int	int	
xs:unsignedInt	long	Java does not have unsigned types.
xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth	com.altova.types.DateTime	
xs:duration	com.altova.types.Duration	
xs:hexBinary and	byte[]	Encoding and decoding of

XML Schema	Java	Remarks
xs:base64Binary		binary data is done automatically.
xs:anySimpleType	string	

All XML Schema types not contained in this list are derived types, and mapped to the same Java type as their respective base type.

Generated Classes

For each type in the schema, a class is generated that contains a member for each attribute and element of the type. The members are named the same as the attributes or elements in the original schema (in case of possible collisions, a number is appended). For simple types, assignment and conversion operators are generated. For simple types with enumeration facets, the methods `GetEnumerationValue()` and `SetEnumerationValue(int)` can be used together with generated constants for each enumeration value. In addition, the method `StaticInfo()` allows accessing schema information as one of the following types:

`com.altova.xml.meta.SimpleType`
`com.altova.xml.meta.ComplexType`

Classes generated from complex types include the method `SetXsiType()`, which enables you to set the `xsi:type` attribute of the type. This method is useful when you want to create XML instance elements of a derived type.

In addition to the classes for the types declared in the XML Schema, a document class (identified with "Doc" below) is generated. It contains all possible root elements as members, and various other methods. For more information about the class, see `com.[YourSchema].[Doc]`.

Note: The actual class name depends on the name of the .xsd schema.

For each member attribute or element of a schema type, a new class is generated. For more information about such classes, see:

`com.[YourSchema].[YourSchemaType].MemberAttribute`
`com.[YourSchema].[YourSchemaType].MemberElement`

Note: The actual class names depend on the name of the schema attribute or element.

Error Handling

Errors are reported by exceptions. The following exception classes are defined in the namespace com.altova:

Class	Base Class	Description
SourceInstanceUnvailableException	Exception	A problem occurred while loading an XML instance.
TargetInstanceUnavailableException	Exception	A problem occurred while saving an XML instance.

In addition, the following Java exceptions are commonly used:

Class	Description
java.lang.Error	Internal program logic error (independent of input data)
java.lang.Exception	Base class for runtime errors
java.lang.IllegalArgumentsException	A method was called with invalid argument values, or a type conversion failed.
java.lang.ArithmeticException	Exception thrown when a numeric type conversion fails.

Accessing schema information

The generated library allows accessing static schema information via the following classes:

com.altova.xml.meta.Attribute
com.altova.xml.meta.ComplexType
com.altova.xml.meta.Element
com.altova.xml.meta.SimpleType

The properties that return one of the metadata classes return null if the respective property does not exist.

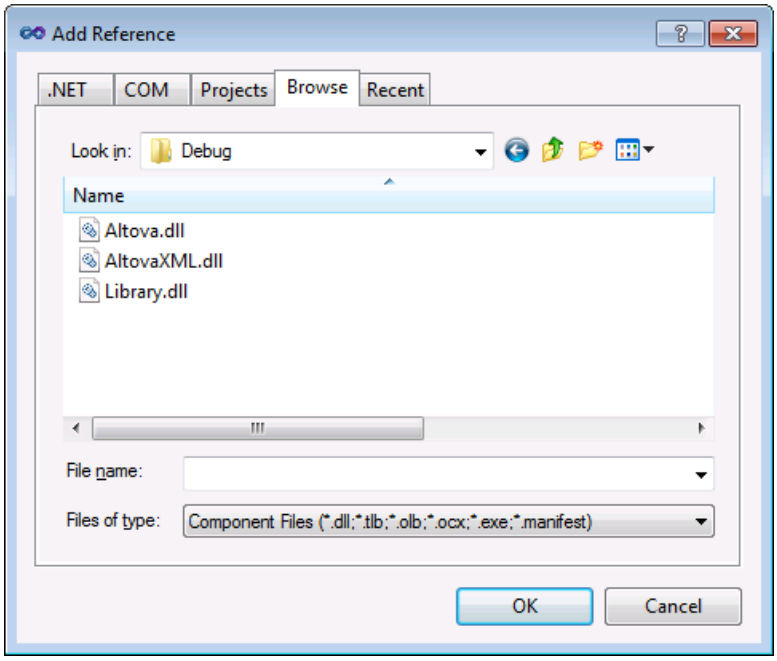
25.3.4 Integrating Schema Wrapper Libraries

To use the Altova libraries in your custom project, refer to the libraries from your project (or include them into your project), as shown below for each language.

C#

To integrate the Altova libraries into an existing C# project:

1. After XMLSpy generates code from a schema (for example, **YourSchema.xsd**), build the generated **YourSchema.sln** solution in Visual Studio. This solution is in a project folder with the same name as the schema.
2. Right-click your existing project in Visual Studio, and select **Add Reference**.
3. On the Browse tab, browse for the following libraries: **Altova.dll**, **AltovaXML.dll**, and **YourSchema.dll** located in the output directory of the generated projects (for example, **bin \Debug**).



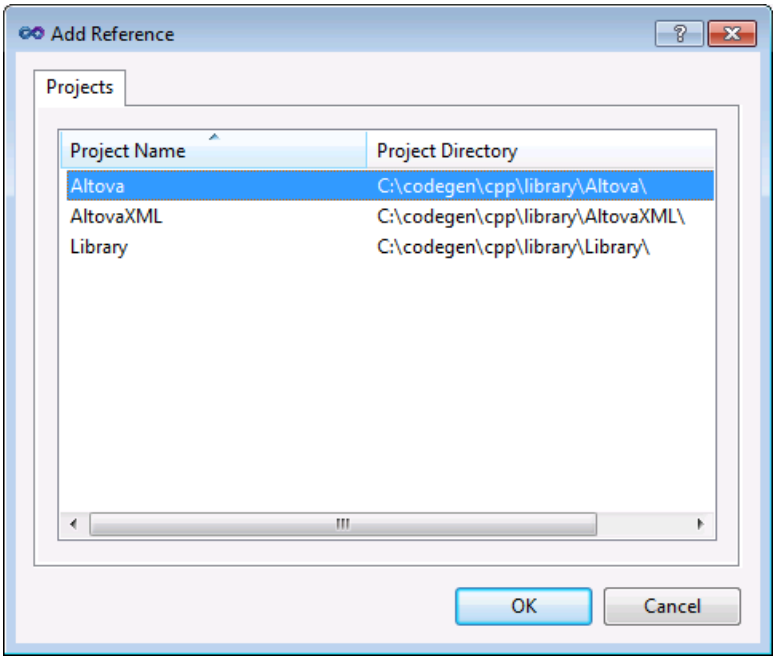
C++

The easiest way to integrate the libraries into an existing C++ project is to add the generated project files to your solution. For example, let's assume that you generated code from a schema called **Library.xsd** and selected **c:\codegen\cpp\library** as target directory. The generated libraries in this case are available at:

- c:\codegen\cpp\library\Altova.vcxproj
- c:\codegen\cpp\library\AltovaXML\AltovaXML.vcxproj
- c:\codegen\cpp\library\Library.vcxproj

First, open the generated **c:\codegen\cpp\library\Library.sln** solution and build it in Visual Studio.

Next, open your existing Visual Studio solution (in Visual Studio 2010, in this example), right-click it, select **Add | Existing Project**, and add the project files listed above, one by one. Be patient while Visual Studio parses the files. Next, right-click your project and select **Properties**. In the Property Pages dialog box, select **Common Properties | Framework and References**, and then click **Add New Reference**. Next, select and add each of the following projects: *Altova*, *AltovaXML*, and *Library*.



See also the MSDN documentation for using functionality from a custom library, as applicable to your version of Visual Studio, for example:

- If you chose to generate static libraries, see https://msdn.microsoft.com/en-us/library/ms235627(v=vs.100).aspx
- If you chose to generate dynamic libraries, see https://msdn.microsoft.com/en-us/library/ms235636(v=vs.100).aspx

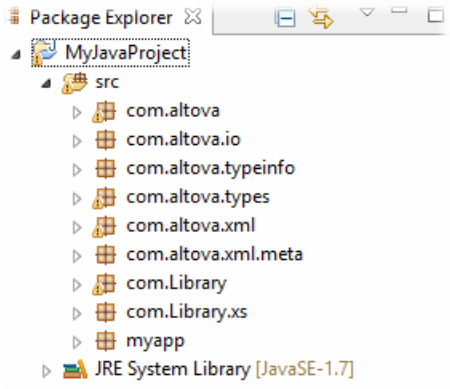
The option to generate static or dynamic libraries is available in code generation options (see Code generator options).

Java

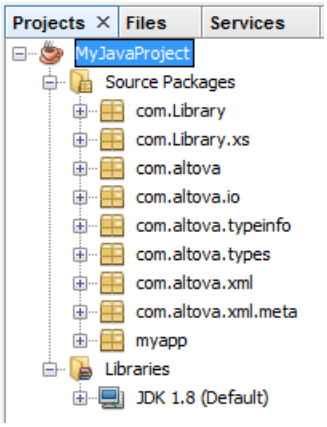
One of the ways to integrate the Altova packages into your Java project is to copy the **com** directory of the generated code to the directory which stores the source packages of your Java project (for example, **C:\Workspace\MyJavaProject\src**). For example, let's assume that you generated code in **c:\codegen\java\library**. The generated Altova classes in this case are available at **c:\codegen\java\library\com**.

After copying the libraries, refresh the project. To refresh the project in Eclipse, select it in the Package Explorer, and press **F5**. To refresh the project in NetBeans IDE 8.0, select the menu command **Source | Scan for External Changes**.

Once you perform the copy operation, the Altova packages are available in the Package Explorer (in case of Eclipse), or under "Source Packages" in the Projects pane (in case of NetBeans IDE).



Altova packages in Eclipse 4.4



Altova packages in NetBeans IDE 8.0.2

25.3.5 Example: Using the Schema Wrapper Libraries

This example illustrates how to use the generated schema wrapper libraries in order to write or read programmatically XML documents conformant to the schema. Before using the sample code, take some time to understand the structure of the included example schema. You will need this schema to generate the code libraries used in this example. Understanding the example schema will help you get started with the code generated from your schema and adapt it to your needs.

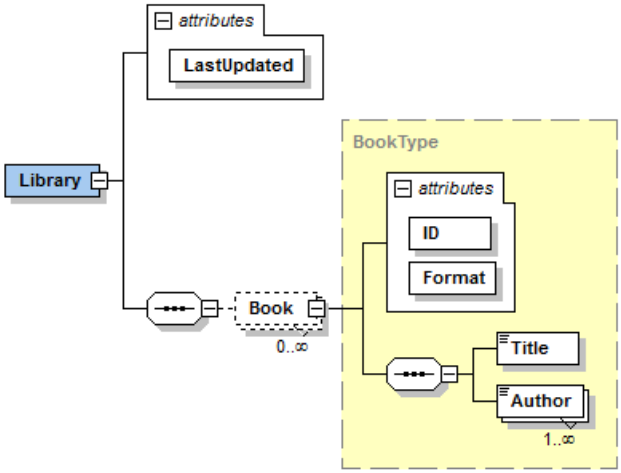
Example Schema

The schema used in this example describes a library of books. The complete definition of the schema is shown below. Save this code listing as **Library.xsd** if you want to get the same results as this example. You will need this schema to generate the code libraries used in this example.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.nanonull.com/LibrarySample" xmlns:xs="http://
www.w3.org/2001/XMLSchema" targetNamespace="http://www.nanonull.com/
LibrarySample" elementFormDefault="qualified"
```

```
attributeFormDefault="unqualified">
   <xs:element name="Library">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Book" type="BookType" minOccurs="0"
maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="LastUpdated" type="xs:dateTime"/>
      </xs:complexType>
   </xs:element>
   <xs:complexType name="BookType">
      <xs:sequence>
         <xs:element name="Title" type="xs:string"/>
         <xs:element name="Author" type="xs:string" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="ID" type="xs:integer" use="required"/>
      <xs:attribute name="Format" type="BookFormatType" use="required"/>
   </xs:complexType>
   <xs:complexType name="DictionaryType">
      <xs:complexContent>
         <xs:extension base="BookType">
            <xs:sequence>
               <xs:element name="FromLang" type="xs:string"/>
               <xs:element name="ToLang" type="xs:string"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:simpleType name="BookFormatType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="Hardcover"/>
         <xs:enumeration value="Paperback"/>
         <xs:enumeration value="Audiobook"/>
         <xs:enumeration value="E-book"/>
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

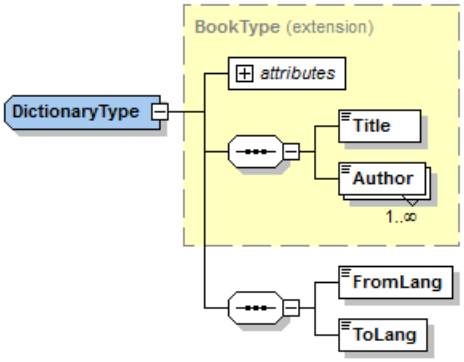
Library is a root element of a `complexType` which can be graphically represented as follows in the schema view of XMLSpy:



As shown above, the library has a **LastUpdated** attribute (defined as `xs:dateTime`), and stores a sequence of books. Each book is an `xs:complexType` and has two attributes: an **ID** (defined as `xs:integer`), and a **Format**. The format of any book can be hardcover, paperback, audiobook, or e-book. In the schema, **Format** is defined as `xs:simpleType` which uses an enumeration of the above-mentioned values.

Each book also has a **Title** element (defined as `xs:string`), as well as one or several **Author** elements (defined as `xs:string`).

The library may also contain books that are dictionaries. Dictionaries have the type `DictionaryType`, which is derived by extension from the `BookType`. In other words, a dictionary inherits all attributes and elements of a Book, plus two additional elements: **FromLang** and **ToLang**, as illustrated below.



The **FromLang** and **ToLang** elements store the source and destination language of the dictionary.

An XML instance file valid according to the schema above could therefore look as shown in the listing below (provided that it is in the same directory as the schema file):

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample" xsi:schemaLocation="http://
www.nanonull.com/LibrarySample Library.xsd" LastUpdated="2016-02-
03T17:10:08.4977404">
```

```
   <Book ID="1" Format="E-book">
      <Title>The XMLSpy Handbook</Title>
      <Author>Altova</Author>
   </Book>
   <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/
LibrarySample" xsi:type="n1:DictionaryType">
      <Title>English-German Dictionary</Title>
      <Author>John Doe</Author>
      <FromLang>English</FromLang>
      <ToLang>German</ToLang>
   </Book>
</Library>
```

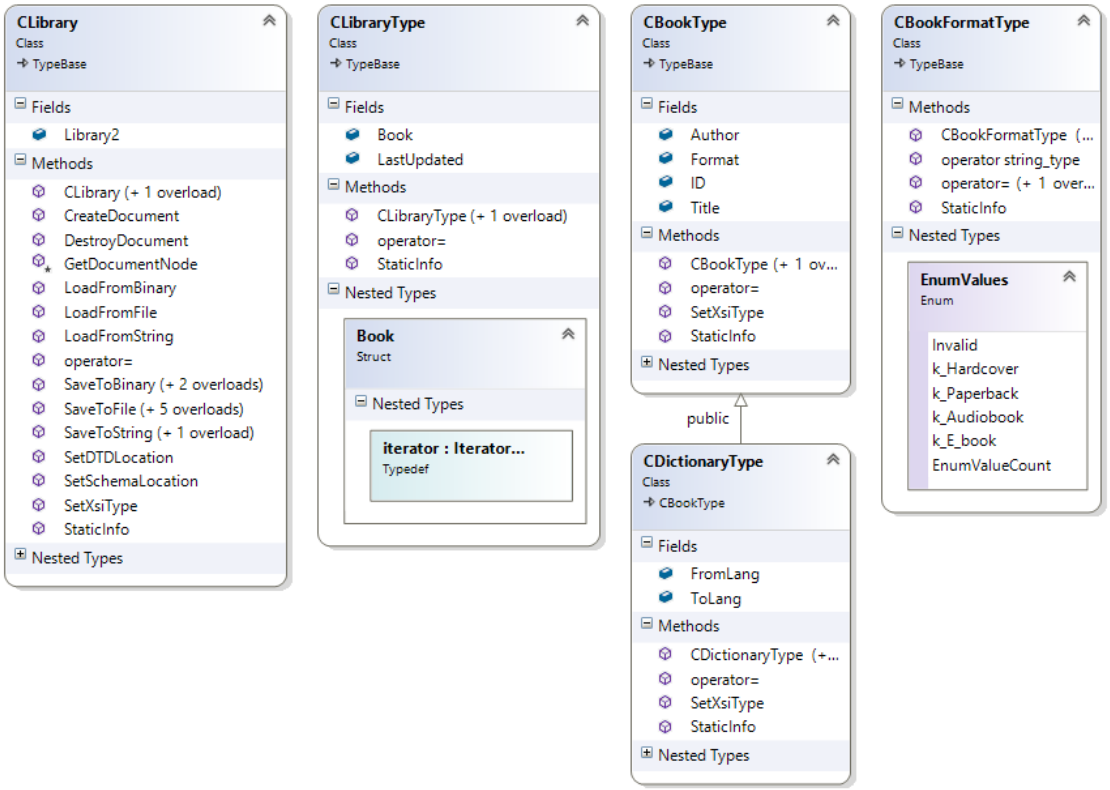
The next topics illustrate how to read from such a file programmatically, or write to such a file programmatically. To begin, generate the schema wrapper code from the schema above, using the steps described in Generating Code from XML Schemas or DTD.

Reading and Writing XML Documents (C++)

After you generate code from the Library schema (see Example Schema), a test C++ application is created, along with several supporting Altova libraries.

About the generated C++ libraries

The following diagram illustrates some of the most important classes of the generated code.



The central class of the generated code is the `CLibrary` class, which represents the XML

document. Such a class is generated for every schema and its name depends on the schema file name (**Library.xsd**, in this example). As shown in the diagram, this class provides methods for loading documents from files, binary streams, or strings (or saving documents to files, streams, strings). For a list of all members exposed by this class, see the class reference (`[YourSchema]::[CDoc]`).

The `Library2` field of the `CLibrary` class represents the actual root of the document. The number at the end is meant to avoid a naming conflict with the class name. **Library** is an element in the XML file, so in the C++ code it has a template class as type (`MemberElement`). The template class exposes methods and properties for interacting with the **Library** element. In general, each attribute and each element of a type in the schema is typed in the generated code with the `MemberAttribute` and `MemberElement` template classes, respectively. For more information, see `[YourSchema]::MemberAttribute` and `[YourSchema]::MemberElement` class reference.

The class `CLibraryType` is generated from the schema complex type with the same name, as mentioned in About Schema Wrapper Libraries (C++). Notice that the `CLibraryType` class contains a field `Book`, and a field `LastUpdated`. According to the logic already mentioned above, these correspond to the **Book** element and **LastUpdated** attribute in the schema, and enable you to manipulate programmatically (append, remove, etc) elements and attributes in the instance XML document.

Since the **DictionaryType** is a complex type derived from **BookType** in the schema, this relationship is also reflected in the generated classes. As illustrated in the diagram, the class `CDictionaryType` inherits the `CBookType` class.

If your XML schema defines simple types as enumerations, the enumerated values become available as `Enum` values in the generated code. In the schema used in this example, a book format can be hardcover, paperback, e-book, and so on. Therefore, in the generated code, these values would be available through an `Enum` that is a member of the `CBookFormatType` class.

Writing an XML document

1. Open the **LibraryTest.sln** solution in Visual Studio generated from the Library schema mentioned earlier in this example.

> While prototyping an application from a frequently changing XML schema, you may need to frequently generate code to the same directory, so that the schema changes are immediately reflected in the code. Note that the generated test application and the Altova libraries are overwritten every time when you generate code into the same target directory. Therefore do not add code to the generated test application. Instead, integrate the Altova libraries into your project (see Integrating Schema Wrapper Libraries).

2. In Solution Explorer, open the **LibraryTest.cpp** file, and edit the `Example()` method as shown below.

```
#include <ctime> // required to get current time
using namespace Library;// required to work with Altova libraries

void Example()
{
	// Create a new, empty XML document
	CLibrary libDoc = CLibrary::CreateDocument();
```

```
        // Create the root element <Library> and add it to the document
        CLibraryType lib = libDoc.Library2.append();

        // Get current time and set the "LastUpdated" attribute using Altova classes
        time_t t = time(NULL);
        struct tm * now = localtime( & t );
        altova:DateTime dt = altova:DateTime(now->tm_year + 1900,now->tm_mon + 1,now-
>tm_mday,now->tm_hour,now->tm_min,now->tm_sec);
        lib.LastUpdated = dt;

        // Create a new <Book> and add it to the library
        CBookType book = lib.Book.append();

        // Set the "ID" attribute of the book
        book.ID = 1;

        // Set the "Format" attribute of the <Book> using an enumeration constant
        book.FormatSetEnumerationValue(CBookFormatType::k_Paperback );

        // Add the <Title> and <Author> elements, and set values
        book.Title.append() = _T("The XML Spy Handbook");
        book.Author.append() = _T("Altova");

        // Append a dictionary (book of derived type) and populate its attributes and elements
        CDictionaryType dictionary = CDictionaryType(lib.Book.append().GetNode());
        dictionary.ID = 2;
        dictionary.FormatSetEnumerationValue(CBookFormatType::k_E_book);
        dictionary.Title.append() = _T("English-German Dictionary");
        dictionary.Author.append() = _T("John Doe");
        dictionary.FromLang.append() = _T("English");
        dictionary.ToLang.append() = _T("German");

        // Since dictionary a derived type, set the xsi:type attribute of the book element
        dictionary.SetXsiType();

        // Optionally, set the schema location
        libDoc.SetSchemaLocation(_T("Library.xsd"));

        // Save the XML document to a file with default encoding (UTF-8),
        // "true" causes the file to be pretty-printed.
        libDoc.SaveToFile(_T("GeneratedLibrary.xml"),true);

        // Destroy the document
        libDoc.DestroyDocument();
}
```

3. Press **F5** to start debugging. If the code was executed successfully, a **GeneratedLibrary.xml** file is created in the solution output directory.

Reading an XML document

1. Open the **LibraryTest.sln** solution in Visual Studio.
2. Save the code below as **Library1.xml** to a directory that can be read by the program code (for example, the same directory as **LibraryTest.sln**).

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample" xsi:schemaLocation="http://
www.nanonull.com/LibrarySample Library.xsd" LastUpdated="2016-02-
03T17:10:08.4977404">
   <Book ID="1" Format="E-book">
      <Title>The XMLSpy Handbook</Title>
      <Author>Altova</Author>
   </Book>
   <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/
LibrarySample" xsi:type="n1:DictionaryType">
      <Title>English-German Dictionary</Title>
      <Author>John Doe</Author>
      <FromLang>English</FromLang>
      <ToLang>German</ToLang>
   </Book>
</Library>
```

3. In Solution Explorer, open the **LibraryTest.cpp** file, and edit the `Example()` method as shown below.

```
using namespace Library;
void Example()
{
	// Load XML document
	CLibrary libDoc = CLibrary::LoadFromFile(_T("Library1.xml"));

	// Get the first (and only) root element <Library>
	CLibraryType lib = libDoc.Library2.first();

	// Check whether an element exists:
	if (!lib.Book.exists())
	{
		tcout << "This library is empty." << std::endl;
		return;
	}

	// iteration: for each <Book> ...
	for (Iterator<CBookType> itBook = lib.Book.all(); itBook; ++itBook)
	{
		// output values of ISBN attribute and (first and only) title element
		tcout << "ID:" << itBook->ID << std::endl;
		tcout << "Title:" << tstring(itBook->Title.first()) << std::endl;

		// read and compare an enumeration value
		if (itBook->Format.GetEnumerationValue() == CBookFormatType::k_Paperback)
			tcout << "This is a paperback book." << std::endl;

		// for each <Author> ...
		for (CBookType::Author::iterator itAuthor = itBook->Author.all(); itAuthor; +
+itAuthor)
			tcout << "Author:" << tstring(itAuthor) << std::endl;

		// alternative: use count and index
		for (unsigned int j = 0; j < itBook->Author.count(); ++j)
```

```
                    tcout << "Author:" << tstring(itBook->Author[j]) << std::endl;
        }

        // Destroy the document
        libDoc.DestroyDocument();
}
```

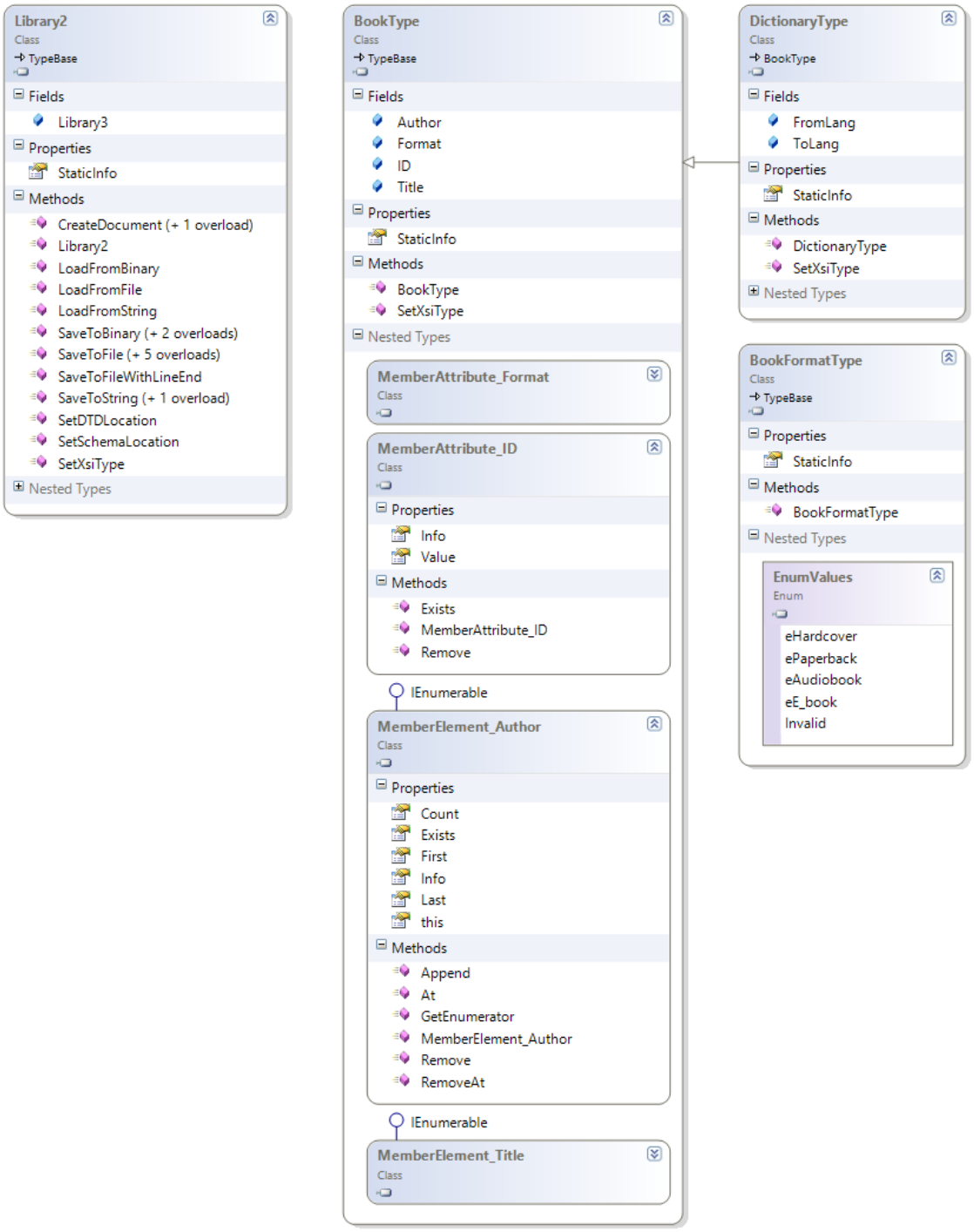
4. Press **F5** to start debugging.

Reading and Writing XML Documents (C#)

After you generate code from the Library schema (see Example Schema), a test C# application is created, along with several supporting Altova libraries.

About the generated C# libraries

The following diagram illustrates some of the most important classes of the generated code.



The central class of the generated code is the `Library2` class, which represents the XML document. Such a class is generated for every schema and its name depends on the schema file name (**Library.xsd**, in this example). Note that this class is called `Library2` to avoid a possible conflict with the namespace name. As shown in the diagram, this class provides methods for loading documents from files, binary streams, or strings (or saving documents to files, streams, strings). For a description of this class, see the class reference ([YourSchema].[Doc]).

The `Library3` member of the `Library2` class represents the actual root of the document. Again,

the number at the end is meant to avoid a naming conflict with the class name.

According to the code generation rules mentioned in About Schema Wrapper Libraries (C#), member classes are generated for each attribute and for each element of a type. In the generated code, the name of such member classes is prefixed with `MemberAttribute_` and `MemberElement_`, respectively. In the diagram above, examples of such classes are `MemberAttribute_ID` and `MemberElement_Author`, generated from the **Author** element and **ID** attribute of a book, respectively. Such classes enable you to manipulate programmatically the corresponding elements and attributes in the instance XML document (for example, append, remove, set value, etc). For more information, see `[YourSchemaType].MemberAttribute` and `[YourSchemaType].MemberElement` class reference.

Since the **DictionaryType** is a complex type derived from **BookType** in the schema, this relationship is also reflected in the generated classes. As illustrated in the diagram, the class `DictionaryType` inherits the `BookType` class.

If your XML schema defines simple types as enumerations, the enumerated values become available as `Enum` values in the generated code. In the schema used in this example, a book format can be hardcover, paperback, e-book, and so on. Therefore, in the generated code, these values would be available through an `Enum` that is a member of the `BookFormatType` class.

Writing an XML document

1. Open the **LibraryTest.sln** solution in Visual Studio generated from the Library schema mentioned earlier in this example.

> While prototyping an application from a frequently changing XML schema, you may need to frequently generate code to the same directory, so that the schema changes are immediately reflected in the code. Note that the generated test application and the Altova libraries are overwritten every time when you generate code into the same target directory. Therefore do not add code to the generated test application. Instead, integrate the Altova libraries into your project (see Integrating Schema Wrapper Libraries).

2. In Solution Explorer, open the **LibraryTest.cs** file, and edit the `Example()` method as shown below.

```
        protected static void Example()
        {
            // Create a new XML library
            Library2 doc = Library2.CreateDocument();
            // Append the root element
            LibraryType root = doc.Library3.Append();

            // Create the library generation date using Altova DateTime
class
            Altova.Types.DateTime dt = new
Altova.Types.DateTime(System.DateTime.Now);
            // Append the date to the root
            root.LastUpdated.Value = dt;

            // Add a new book
            BookType book = root.Book.Append();
            // Set the value of the ID attribute
            book.ID.Value = 1;
            // Set the format of the book (enumeration)
```

```
            book.Format.EnumerationValue =
BookFormatType.EnumValues.eHardcover;
            // Set the Title and Author elements
            book.Title.Append().Value = "The XMLSpy Handbook";
            book.Author.Append().Value = "Altova";

            // Append a dictionary (book of derived type) and populate its
attributes and elements
            DictionaryType dictionary = new
DictionaryType(root.Book.Append().Node);
            dictionary.ID.Value = 2;
            dictionary.Title.Append().Value = "English-German Dictionary";
            dictionary.Format.EnumerationValue =
BookFormatType.EnumValues.eE_book;
            dictionary.Author.Append().Value = "John Doe";
            dictionary.FromLang.Append().Value = "English";
            dictionary.ToLang.Append().Value = "German";
            // Since it's a derived type, make sure to set the xsi:type
attribute of the book element
            dictionary.SetXsiType();

            // Optionally, set the schema location (adjust the path if
            // your schema is not in the same folder as the generated
instance file)
            doc.SetSchemaLocation("Library.xsd");

            // Save the XML document with the "pretty print" option enabled
            doc.SaveToFile("GeneratedLibrary.xml", true);
        }
```

3. Press **F5** to start debugging. If the code was executed successfully, a **GeneratedLibrary.xml** file is created in the solution output directory (typically, **bin/Debug**).

Reading an XML document

1. Open the **LibraryTest.sln** solution in Visual Studio.
2. Save the code below as **Library.xml** to the output directory of the project (by default, **bin/Debug**). This is the file that will be read by the program code.

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample" xsi:schemaLocation="http://
www.nanonull.com/LibrarySample Library.xsd" LastUpdated="2016-02-
03T17:10:08.4977404">
   <Book ID="1" Format="E-book">
      <Title>The XMLSpy Handbook</Title>
      <Author>Altova</Author>
   </Book>
   <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/
LibrarySample" xsi:type="n1:DictionaryType">
      <Title>English-German Dictionary</Title>
      <Author>John Doe</Author>
      <FromLang>English</FromLang>
      <ToLang>German</ToLang>
   </Book>
</Library>
```

3. In Solution Explorer, open the **LibraryTest.cs** file, and edit the `Example()` method as shown below.

```
       protected static void Example()
        {
            // Load the XML file into a new Library instance
            Library2 doc = Library2.LoadFromFile("Library.xml");
            // Get the root element
            LibraryType root = doc.Library3.First;

            // Read the library generation date
            Altova.Types.DateTime dt = root.LastUpdated.Value;
            string dt_as_string = dt.ToString(DateTimeFormat.W3_dateTime);
            Console.WriteLine("The library generation date is: " +
dt_as_string);

            // Iteration: for each <Book>...
            foreach (BookType book in root.Book)
            {
                // Output values of ID attribute and (first and only) title
element
                Console.WriteLine("ID:    " + book.ID.Value);
                Console.WriteLine("Title: " + book.Title.First.Value);

                // Read and compare an enumeration value
                if (book.Format.EnumerationValue ==
BookFormatType.EnumValues.ePaperback)
                    Console.WriteLine("This is a paperback book.");

                // Iteration: for each <Author>
                foreach (xs.stringType author in book.Author)
                    Console.WriteLine("Author: " + author.Value);

                // Determine if this book is of derived type
                if (book.Node.Attributes.GetNamedItem("xsi:type") != null)
                {
                    // Find the value of the xsi:type attribute
                    string xsiTypeValue =
book.Node.Attributes.GetNamedItem("xsi:type").Value;
                    // Get the namespace URI and the lookup prefix of this
namespace
                    string namespaceUri = book.Node.NamespaceURI;
                    string prefix =
book.Node.GetPrefixOfNamespace(namespaceUri);

                    // if this book has DictionaryType
                    if (namespaceUri == "http://www.nanonull.com/
LibrarySample" && xsiTypeValue.Equals(prefix + ":DictionaryType"))
                    {
                        // output additional fields
                        DictionaryType dictionary = new
DictionaryType(book.Node);
                        Console.WriteLine("Language from: " +
dictionary.FromLang.First.Value);
                        Console.WriteLine("Language to: " +
dictionary.ToLang.First.Value);
                    }
                    else
                    {
```

```
                        throw new Exception("Unexpected book type");
                    }
                }
            }

            Console.ReadLine();
        }
```

4. Press **F5** to start debugging. If the code was executed successfully, **Library.xml** will be read by the program code, and its contents displayed as console output.

Reading and writing elements and attributes

Values of attributes and elements can be accessed using the `Value` property of the generated member element or attribute class, for example:

```
// Output values of ID attribute and (first and only) title element
Console.WriteLine("ID:    " + book.ID.Value);
Console.WriteLine("Title:" + book.Title.First.Value);
```

To get the value of the **Title** element in this particular example, we also used the `First()` method, since this is the first (and only) **Title** element of a book. For cases when you need to pick a specific element from a list by index, use the `At()` method.

The class generated for each member element of a type implements the standard `System.Collections.IEnumerable` interface. This makes it possible to loop through multiple elements of the same type. In this particular example, you can loop through all books of a Library object as follows:

```
// Iteration: for each <Book> ...
foreach (BookType book in root.Book)
{
    // your code here ...
}
```

To add a new element, use the `Append()` method. For example, the following code appends the root element to the document:

```
// Append the root element to the library
LibraryType root = doc.Library3.Append();
```

You can set the value of an attribute (like ID in this example) as follows:

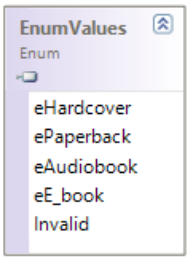
```
// Set the value of the ID attribute
book.ID.Value = 1;
```

For further information, see `[YourSchemaType].MemberAttribute` and `[YourSchemaType].MemberElement` class reference.

Reading and writing enumeration values

If your XML schema defines simple types as enumerations, the enumerated values become

available as `Enum` values in the generated code. In the schema used in this example, a book format can be hardcover, paperback, e-book, and so on. Therefore, in the generated code, these values would be available through an `Enum`:



To assign enumeration values to an object, use code such as the one below:

```
// Set the format of the book (enumeration)
book.FormatEnumerationValue = BookFormatType.EnumValues.eHardcover;
```

You can read such enumeration values from XML instance documents as follows:

```
// Read and compare an enumeration value
if (book.FormatEnumerationValue == BookFormatType.EnumValues.ePaperback)
Console.WriteLine("This is a paperback book.");
```

When an "if" condition is not enough, create a switch to determine each enumeration value and process it as required.

Working with xs:dateTime and xs:duration types

If the schema from which you generated code uses time and duration types such as `xs:dateTime`, or `xs:duration`, these are converted to Altova native classes in generated code. Therefore, to write a date or duration value to the XML document, do the following:

1. Construct an `Altova.Types.DateTime` or `Altova.Types.Duration` object (either from `System.DateTime`, or by using parts such as hours and minutes, see `Altova.Types.DateTime` and `Altova.Types.Duration` for more information).
2. Set the object as value of the required element or attribute, for example:

```
// Create the library generation date using Altova DateTime class
Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);
// Append the date to the root
root.LastUpdatedValue = dt;
```

To read a date or duration from an XML document, do the following:

1. Declare the element value (or attribute) as `Altova.Types.DateTime` or `Altova.Types.Duration` object.
2. Format the required element or attribute, for example:

```
// Read the library generation date
Altova.Types.DateTime dt = root.LastUpdatedValue;
string dt_as_string = dt.ToString(DateTimeFormat.W3_dateTime);
Console.WriteLine("The library generation date is:" + dt_as_string);
```

For more information, see Altova.Types.DateTime and Altova.Types.Duration class reference.

Working with derived types

If your XML schema defines derived types, you can preserve type derivation in XML documents that you create or load programmatically. Taking the schema used in this example, the following code listing illustrates how to create a new book of derived type `DictionaryType`:

```
// Append a dictionary (book of derived type) and populate its attributes and elements
DictionaryType dictionary = new DictionaryType(root.Book.Append().Node);
dictionary.ID.Value = 2;
dictionary.Title.Append().Value = "English-German Dictionary";
dictionary.Author.Append().Value = "John Doe";
dictionary.FromLanguage.Append().Value = "English";
dictionary.ToLanguage.Append().Value = "German";

// Since it's a derived type, make sure to set the xsi:type attribute of the book element
dictionary.SetXsiType();
```

Note that it is important to set the `xsi:type` attribute of the newly created book. This ensures that the book type will be interpreted correctly by the schema when the XML document is validated.

When you load data from an XML document, the following code listing shows how to identify a book of derived type `DictionaryType` in the loaded XML instance. First, the code finds the value of the `xsi:type` attribute of the book node. If the namespace URI of this node is `http://www.nanonull.com/LibrarySample`, and if the URI lookup prefix and type matches the value of the `xsi:type` attribute, then this is a dictionary:

```
      // Determine if this book is of derived type
      if (book.Node.Attributes.GetNamedItem("xsi:type") != null)
      {
          // Find the value of the xsi:type attribute
          string xsiTypeValue =
book.Node.Attributes.GetNamedItem("xsi:type").Value;
          // Get the namespace URI and the lookup prefix of this namespace
          string namespaceUri = book.Node.NamespaceURI;
          string prefix = book.Node.GetPrefixOfNamespace(namespaceUri);

          // if this book has DictionaryType
          if (namespaceUri == "http://www.nanonull.com/LibrarySample" &&
xsiTypeValue.Equals(prefix + ":DictionaryType"))
          {
             // output additional fields
             DictionaryType dictionary = new DictionaryType(book.Node);
             Console.WriteLine("Language from: " +
dictionary.FromLang.First.Value);
             Console.WriteLine("Language to: " +
dictionary.ToLang.First.Value);
          }
          else
          {
              throw new Exception("Unexpected book type");
          }
      }
```

Reading and Writing XML Documents (Java)

After you generate code from the Library schema (see Example Schema), a test Java project is created, along with several supporting Altova libraries.

About the generated Java libraries

The following diagram illustrates some of the most important classes of the generated code.

The central class of the generated code is the `Library2` class, which represents the XML document. Such a class is generated for every schema and its name depends on the schema file name (**Library.xsd**, in this example). Note that this class is called `Library2` to avoid a possible conflict with the namespace name. As shown in the diagram, this class provides methods for

loading documents from files, binary streams, or strings (or saving documents to files, streams, strings). For a description of this class, see the class reference (com.[YourSchema].[Doc]).

The Library3 member of the Library2 class represents the actual root of the document. Again, the number at the end is meant to avoid a naming conflict with the class name.

According to the code generation rules mentioned in About Generated Java Code, member classes are generated for each attribute and for each element of a type. In the generated code, the name of such member classes is prefixed with MemberAttribute_ and MemberElement_, respectively. In the diagram above, examples of such classes are MemberAttribute_ID and MemberElement_Author, generated from the **Author** element and **ID** attribute of a book, respectively. Such classes enable you to manipulate programmatically the corresponding elements and attributes in the instance XML document (for example, append, remove, set value, etc). For more information, see com.[YourSchema].[YourSchemaType].MemberAttribute and com.[YourSchema].[YourSchemaType].MemberElement class reference.

Since the **DictionaryType** is a complex type derived from **BookType** in the schema, this relationship is also reflected in the generated classes. As illustrated in the diagram, the class DictionaryType inherits the BookType class.

If your XML schema defines simple types as enumerations, the enumerated values become available as Enum values in the generated code. In the schema used in this example, a book format can be hardcover, paperback, e-book, and so on. Therefore, in the generated code, these values would be available through an Enum that is a member of the BookFormatType class.

Writing an XML document

1. On the **File** menu of Eclipse, click **Import**, select **Existing Projects into Workspace**, and click **Next**.
2. Next to **Select root directory**, click **Browse**, select the directory to which you generated the Java code, and then click **Finish**.
3. In the Eclipse Package Explorer, expand the **com.LibraryTest** package and open the **LibraryTest.java** file.

While prototyping an application from a frequently changing XML schema, you may need to frequently generate code to the same directory, so that the schema changes are immediately reflected in the code. Note that the generated test application and the Altova libraries are overwritten every time when you generate code into the same target directory. Therefore do not add code to the generated test application. Instead, integrate the Altova libraries into your project (see Integrating Schema Wrapper Libraries).

4. Edit the Example() method as shown below.

```
protected static void example() throws Exception {
      // create a new, empty XML document
      Library2 libDoc = Library2.createDocument();

      // create the root element <Library> and add it to the document
      LibraryType lib = libDoc.Library3.append();

      // set the "LastUpdated" attribute
      com.altova.types.DateTime dt = new
com.altova.types.DateTime(DateTime.now());
      lib.LastUpdated.setValue(dt);
```

```
      // create a new <Book> and populate its elements and attributes
      BookType book = lib.Book.append();
      book.ID.setValue(java.math.BigInteger.valueOf(1));
      book.Format.setEnumerationValue( BookFormatType.EPAPERBACK );
      book.Title.append().setValue("The XML Spy Handbook");
      book.Author.append().setValue("Altova");

      // create a dictionary (book of derived type)  and populate its
elements and attributes
      DictionaryType dict = new DictionaryType(lib.Book.append().getNode());
      dict.ID.setValue(java.math.BigInteger.valueOf(2));
      dict.Title.append().setValue("English-German Dictionary");
      dict.Format.setEnumerationValue(BookFormatType.EE_BOOK);
      dict.Author.append().setValue("John Doe");
      dict.FromLang.append().setValue("English");
      dict.ToLang.append().setValue("German");
      dict.setXsiType();

      // set the schema location (this is optional)
      libDoc.setSchemaLocation("Library.xsd");

      // save the XML document to a file with default encoding (UTF-8).
"true" causes the file to be pretty-printed.
      libDoc.saveToFile("Library1.xml", true);
   }
```

5. Build the Java project and run it. If the code is executed successfully, a **Library1.xml** file is created in the project directory.

Reading an XML document

1. On the **File** menu of Eclipse, click **Import**, select **Existing Projects into Workspace**, and click **Next**.
2. Next to **Select root directory**, click **Browse**, select the directory to which you generated the Java code, and then click **Finish**.
3. Save the code below as **Library1.xml** to a local directory (you will need to refer to the path of the **Library1.xml** file from the sample code below).

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample" xsi:schemaLocation="http://
www.nanonull.com/LibrarySample Library.xsd" LastUpdated="2016-02-
03T17:10:08.4977404">
   <Book ID="1" Format="E-book">
      <Title>The XMLSpy Handbook</Title>
      <Author>Altova</Author>
   </Book>
   <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/
LibrarySample" xsi:type="n1:DictionaryType">
      <Title>English-German Dictionary</Title>
      <Author>John Doe</Author>
      <FromLang>English</FromLang>
      <ToLang>German</ToLang>
   </Book>
</Library>
```

4. In the Eclipse Package Explorer, expand the **com.LibraryTest** package and open the **LibraryTest.java** file.
5. Edit the `Example()` method as shown below.

```
   protected static void example() throws Exception {
      // load XML document from a path, make sure to adjust the path as
necessary
      Library2 libDoc = Library2.loadFromFile("Library1.xml");

      // get the first (and only) root element <Library>
      LibraryType lib = libDoc.Library3.first();

      // check whether an element exists:
      if (!lib.Book.exists()) {
         System.out.println("This library is empty.");
         return;
      }

      // read a DateTime schema type
      com.altova.types.DateTime dt = lib.LastUpdated.getValue();
      System.out.println("The library was last updated on: " +
dt.toDateString());

      // iteration: for each <Book>...
      for (java.util.Iterator itBook = lib.Book.iterator();
itBook.hasNext();) {
         BookType book = (BookType) itBook.next();
         // output values of ID attribute and (first and only) title element
         System.out.println("ID: " + book.ID.getValue());
         System.out.println("Title: " + book.Title.first().getValue());

         // read and compare an enumeration value
         if (book.Format.getEnumerationValue() == BookFormatType.EPAPERBACK)
            System.out.println("This is a paperback book.");

         // for each <Author>...
         for (java.util.Iterator itAuthor = book.Author.iterator(); itAuthor
               .hasNext();)
            System.out.println("Author: " + ((com.Library.xs.stringType)
itAuthor.next()).getValue());

         // find the derived type of this book
         // by looking at the value of the xsi:type attribute, using DOM

         org.w3c.dom.Node bookNode = book.getNode();
         if (bookNode.getAttributes().getNamedItem("xsi:type") != null) {
            // Get the value of the xsi:type attribute
            String xsiTypeValue =
bookNode.getAttributes().getNamedItem("xsi:type").getNodeValue();

            // Get the namespace URI and lookup prefix of this namespace
            String namespaceUri = bookNode.getNamespaceURI();
            String lookupPrefix = bookNode.lookupPrefix(namespaceUri);

            // If xsi:type matches the namespace URI and type of the book
node
            if (namespaceUri == "http://www.nanonull.com/LibrarySample"
                  && (  xsiTypeValue.equals(lookupPrefix +
":DictionaryType" )))    {
               // ...then this is a book of derived type (dictionary)
```

```
                DictionaryType dictionary = new
DictionaryType(   book.getNode());
                // output the value of the "FromLang" and "ToLang" elements
                System.out.println("From language: " +
dictionary.FromLang.first().getValue());
                System.out.println("To language: " +
dictionary.ToLang.first().getValue());
             }
             else
             {
                // throw an error
                throw new java.lang.Error("This book has an unknown type.");
             }
          }
       }
    }
```

6. Build the Java project and run it. If the code is executed successfully, **Library1.xml** will be read by the program code, and its contents displayed in the Console view.

Reading and writing elements and attributes

Values of attributes and elements can be accessed using the `getValue()` method of the generated member element or attribute class, for example:

```
// output values of ID attribute and (first and only) title element
System.out.println("ID: " + book.ID.getValue());
System.out.println("Title: " + book.Title.first().getValue());
```

To get the value of the **Title** element in this particular example, we also used the `first()` method, since this is the first (and only) **Title** element of a book. For cases when you need to pick a specific element from a list by index, use the `at()` method.

To iterate through multiple elements, use either index-based iteration or `java.util.Iterator`. For example, you can iterate through the books of a library as follows:

```
// index-based iteration
for (int j = 0; j < lib.Book.count(); ++j ) {
   // your code here
}

// alternative iteration using java.util.Iterator
for (java.util.Iterator itBook = lib.Book.iterator(); itBook.hasNext();) {

      // your code here
   }
```

To add a new element, use the `append()` method. For example, the following code appends an empty root **Library** element to the document:

```
// create the root element <Library> and add it to the document
LibraryType lib = libDoc.Library3.append();
```

Once an element is appended, you can set the value of any of its elements or an attributes by using the setValue() method.

```
// set the value of the Title element
book.Title.append().setValue("The XML Spy Handbook");
// set the value of the ID attribute
book.ID.setValue(java.math.BigInteger.valueOf(1));
```

For further information, see `com.[YourSchema].[YourSchemaType].MemberAttribute` and `com.[YourSchema].[YourSchemaType].MemberElement` class reference.

Reading and writing enumeration values

If your XML schema defines simple types as enumerations, the enumerated values become available as `Enum` values in the generated code. In the schema used in this example, a book format can be hardcover, paperback, e-book, and so on. Therefore, in the generated code, these values would be available through an `Enum` (see the `BookFormatType` class diagram above). To assign enumeration values to an object, use code such as the one below:

```
// set an enumeration value
book.Format.setEnumerationValue( BookFormatType.EPAPERBACK );
```

You can read such enumeration values from XML instance documents as follows:

```
// read an enumeration value
if (book.Format.getEnumerationValue() == BookFormatType.EPAPERBACK)
        System.out.println("This is a paperback book."
```

When an "if" condition is not enough, create a switch to determine each enumeration value and process it as required.

Working with xs:dateTime and xs:duration types

If the schema from which you generated code uses time and duration types such as `xs:dateTime`, or `xs:duration`, these are converted to Altova native classes in generated code. Therefore, to write a date or duration value to the XML document, do the following:

1. Construct a `com.altova.types.DateTime` or `com.altova.types.Duration` object.
2. Set the object as value of the required element or attribute, for example:

```
// set the value of an attribute of DateTime type
com.altova.types.DateTime dt = new
com.altova.types.DateTime(DateTime.now());
lib.LastUpdated.setValue(dt);
```

To read a date or duration from an XML document:

1. Declare the element value (or attribute) as `com.altova.types.DateTime` or `com.altova.types.Duration` object.
2. Format the required element or attribute, for example:

```
// read a DateTime type
```

```
com.altova.types.DateTime dt = lib.LastUpdated.getValue();
   System.out.println("The library was last updated on: " +
dt.toDateString());
```

For more information, see com.altova.types.DateTime and com.altova.types.Duration class reference.

Working with derived types

If your XML schema defines derived types, you can preserve type derivation in XML documents that you create or load programmatically. Taking the schema used in this example, the following code listing illustrates how to create a new book of derived type `DictionaryType`:

```
// create a dictionary (book of derived type)  and populate its elements and
attributes
DictionaryType dict = new DictionaryType(lib.Book.append().getNode());
dict.ID.setValue(java.math.BigInteger.valueOf(2));
dict.Title.append().setValue("English-German Dictionary");
dict.Format.setEnumerationValue(BookFormatType.EE_BOOK);
dict.Author.append().setValue("John Doe");
dict.FromLang.append().setValue("English");
dict.ToLang.append().setValue("German");
dict.setXsiType();
```

Note that it is important to set the `xsi:type` attribute of the newly created book. This ensures that the book type will be interpreted correctly by the schema when the XML document is validated.

When you load data from an XML document, the following code listing shows how to identify a book of derived type `DictionaryType` in the loaded XML instance. First, the code finds the value of the `xsi:type` attribute of the book node. If the namespace URI of this node is `http://www.nanonull.com/LibrarySample`, and if the URI lookup prefix and type matches the value of the `xsi:type` attribute, then this is a dictionary:

```
         // find the derived type of this book
         // by looking at the value of the xsi:type attribute, using DOM

         org.w3c.dom.Node bookNode = book.getNode();
         if (bookNode.getAttributes().getNamedItem("xsi:type") != null) {
            // Get the value of the xsi:type attribute
            String xsiTypeValue =
bookNode.getAttributes().getNamedItem("xsi:type").getNodeValue();

            // Get the namespace URI and lookup prefix of the book node
            String namespaceUri = bookNode.getNamespaceURI();
            String lookupPrefix = bookNode.lookupPrefix(namespaceUri);

            // If xsi:type matches the namespace URI and type of the book
node
            if (namespaceUri == "http://www.nanonull.com/LibrarySample"
                  && (  xsiTypeValue.equals(lookupPrefix +
":DictionaryType" )))    {
               // ...then this is a book of derived type (dictionary)

               DictionaryType dictionary = new
DictionaryType(   book.getNode());
               // output the value of the "FromLang" and "ToLang" elements
```

```
                System.out.println("From language: " +
dictionary.FromLang.first().getValue());
                System.out.println("To language: " +
dictionary.ToLang.first().getValue());
             }
             else
             {
                // throw an error
                throw new java.lang.Error("This book has an unknown type.");
             }
          }
```

25.4 Reference to Generated Classes (C++)

This chapter includes a description of C++ classes generated with XMLSpy from a DTD or XML schema (see Generating Code from XML Schemas or DTDs). You can integrate these classes into your code to read, modify, and write XML documents.

Note: The generated code may include other supporting classes, which are not listed here and are subject to modification.

25.4.1 altova::DateTime

This class enables you to process XML attributes or elements that have date and time types, such as `xs:dateTime`.

Constructors

Name	Description
`DateTime()`	Initializes a new instance of the `DateTime` class to 12:00:00 midnight, January 1, 0001.
`DateTime(__int64 value, short timezone)`	Initializes a new instance of the `DateTime` class. The `value` parameter represents the number of ticks (100-nanosecond intervals) that have elapsed since 12:00:00 midnight, January 1, 0001.
`DateTime(int year, unsigned char month, unsigned char day, unsigned char hour, unsigned char minute, double second)`	Initializes a new instance of the `DateTime` class to the year, month, day, hour, minute, and second supplied as argument.
`DateTime(int year, unsigned char month, unsigned char day, unsigned char hour, unsigned char minute, double second, short timezone)`	Initializes a new instance of the `DateTime` class to the year, month, day, hour, minute, second and timezone supplied as argument. The timezone is expressed in minutes and can be positive or negative. For example, the timezone "UTC-01:00" is expressed as "-60".

Methods

Name	Description
`unsigned char Day() const`	Returns the day of month of the current `DateTime` object. The return values range from 1 through 31.
`int DayOfYear() const`	Returns the day of year of the current `DateTime` object. The return values range from 1 through 366.
`bool HasTimezone() const`	Returns Boolean **true** if the current `DateTime` object has a timezone defined; **false** otherwise.
`unsigned char Hour() const`	Returns the hour of the current `DateTime` object. The return values range from 0 through 23.
`static bool IsLeapYear(int`	Returns Boolean **true** if the year of the `DateTime` class

Name	Description
year)	is a leap year; **false** otherwise.
unsigned char Minute() const	Returns the minute of the current DateTime object. The return values range from 0 through 59.
unsigned char Month() const	Returns the month of the current DateTime object. The return values range from 1 through 12.
__int64 NormalizedValue() const	Returns the value of the DateTime object expressed as the Coordinated Universal Time (UTC).
double Second() const	Returns the second of the current DateTime object. The return values range from 0 through 59.
void SetTimezone(short tz)	Sets the timezone of the current DateTime object to the timezone value supplied as argument. The **tz** argument is expressed in minutes and can be positive or negative.
short Timezone() const	Returns the timezone, in minutes, of the current DateTime object. Before using this method, make sure that the object actually has a timezone, by calling the HasTimezone() method.
__int64 Value() const	Returns the value of the DateTime object, expressed in the number of ticks (100-nanosecond intervals) that have elapsed since 12:00:00 midnight, January 1, 0001.
int Weekday() const	Returns the day of week of the current DateTime object, as an integer. Values range from 0 through 6, where 0 is Monday (ISO-8601).
int Weeknumber() const	Returns the number of week in the year of the current DateTime object. The return values are according to ISO-8601.
int WeekOfMonth() const	Returns the number of week in the month of the current DateTime object. The return values are according to ISO-8601.
int Year() const	Returns the year of the current DateTime object.

Example

```
void Example()
{
	// initialize a new DateTime instance to 12:00:00 midnight, January 1st, 0001
	altova:DateTime dt1 = altova:DateTime();

	// initialize a new DateTime instance using the year, month, day, hour, minute, and
second
	altova:DateTime dt2 = altova:DateTime(2015,11,10,9,8,7);

	// initialize a new DateTime instance using the year, month, day, hour, minute, second,
and UTC +01:00 timezone
	 altova:DateTime dt = altova:DateTime(2015,11,22,13,53,7,60);
```

```
	// Get the value of this DateTime object
	std::cout << "The number of ticks of the DateTime object is:" << dtValue() <<
std::endl;

	// Get the year
	cout << "The year is:" << dtYear() << endl;
	// Get the month
	cout << "The month is:" << (int)dtMonth() << endl;
	// Get the day of the month
	cout << "The day of the month is:" << (int)dtDay() << endl;
	// Get the day of the year
	cout << "The day of the year is:" << dtDayOfYear() << endl;
	// Get the hour
	cout << "The hour is:" << (int)dtHour() << endl;
	// Get the minute
	cout << "The minute is:" << (int)dtMinute() << endl;
	// Get the second
	cout << "The second is:" << dtSecond() << endl;
	// Get the weekday
	cout << "The weekday is:" << dtWeekday() << endl;
	// Get the week number
	cout << "The week of year is:" << dtWeeknumber() << endl;
	// Get the week in month
	cout << "The week of month is:" << dtWeekOfMonth() << endl;

	// Check whether a DateTime instance has a timezone
	if (dtHasTimezone() == TRUE)
	{
		// output the value of the Timezone
		cout <<  "The timezone is:" << dtTimezone() << endl;
	}
	else
	{
		cout <<  "No timezone has been defined." << endl;
	}

	// Construct a DateTime object with a timezone UTC+01:00 (Vienna)
	altova:DateTime vienna_dt = DateTime(2015,11,23,14,30,59,+60);
	// Output the result in readable format
	cout << "The Vienna time:"
		<< (int)vienna_dtMonth()
		<< "-" << (int)vienna_dtDay()
		<< " " << (int)vienna_dtHour()
		<< ":" << (int)vienna_dtMinute()
		<< ":" << (int)vienna_dtSecond()
		<< endl;

	// Convert the value to UTC time
	DateTime utc_dt = DateTime(vienna_dtNormalizedValue());
	// Output the result in readable format
	cout << "The UTC time:  "
		<< (int)utc_dtMonth()
		<< "-" << (int)utc_dtDay()
		<< " " << (int)utc_dtHour()
		<< ":" << (int)utc_dtMinute()
```

```
                << ":" << (int)utc_dtSecond()
                << endl;

        // Check if a year is a leap year
        int year = 2016;
        if(altova:DateTime::IsLeapYear(year))
        {cout << year << " is a leap year" << endl;}
        else
        {cout << year << " is not a leap year" << endl;}
}
```

25.4.2 altova::Duration

This class enables you to process XML attributes or elements of type `xs:duration`.

Constructors

Name	Description
`Duration()`	Initializes a new instance of the `Duration` class to an empty value.
`Duration(const DayTimeDuration& dt)`	Initializes a new instance of the `Duration` class to a duration defined by the `dt` argument (see altova::DayTimeDuration).
`Duration(const YearMonthDuration& ym )`	Initializes a new instance of the `Duration` class to the duration defined by the `ym` argument (see altova::YearMonthDuration).
`Duration(const YearMonthDuration& ym, const DayTimeDuration& dt)`	Initializes a new instance of the `Duration` class to the duration defined by both the `dt` and the `ym` arguments (see altova::YearMonthDuration and altova::DayTimeDuration).

Methods

Name	Description
`int Days() const`	Returns the number of days in the current `Duration` instance.
`DayTimeDuration DayTime() const`	Returns the day and time duration in the current `Duration` instance, expressed as a `DayTimeDuration` object (see altova::DayTimeDuration).
`int Hours() const`	Returns the number of hours in the current `Duration` instance.
`bool IsNegative() const`	Returns Boolean **true** if the current `Duration` instance is negative.
`bool IsPositive() const`	Returns Boolean **true** if the current `Duration` instance is positive.

Name	Description
int Minutes() const	Returns the number of minutes in the current Duration instance.
int Months() const	Returns the number of months in the current Duration instance.
double Seconds() const	Returns the number of seconds in the current Duration instance.
YearMonthDuration YearMonth() const	Returns the year and month duration in the current Duration instance, expressed as a YearMonthDuration object (see altova::YearMonthDuration).
int Years() const	Returns the number of years in the current Duration instance.

Example

The following code listing illustrates creating a new Duration object, as well as reading values from it.

```
void ExampleDuration()
{
	// Create an empty Duration object
	altova:Duration empty_duration = altova:Duration();

	// Create a Duration object using an existing duration value
	altova:Duration duration1 = altova:Duration(empty_duration);

	// Create a YearMonth duration of six years and five months
	altova:YearMonthDuration yrduration = altova::YearMonthDuration(6,5);

	// Create a DayTime duration of four days, three hours, two minutes, and one second
	altova:DayTimeDuration dtduration = altova:DayTimeDuration(4,3,2,1);

	// Create a Duration object by combining the two previously created durations
	altova:Duration duration = altova:Duration(yrduration,dtduration);

	// Get the number of years in this Duration instance
	cout << "Years: " << duration.Years() << endl;

	// Get the number of months in this Duration instance
	cout << "Months:" << duration.Months() << endl;

	// Get the number of days in this Duration instance
	cout << "Days:  " << duration.Days() << endl;

	// Get the number of hours in this Duration instance
	cout << "Hours: " << duration.Hours() << endl;

	// Get the number of hours in this Duration instance
	cout << "Minutes:" << duration.Minutes() << endl;
```

```
        // Get the number of seconds in this Duration instance
        cout << "Seconds:" << durationSeconds() << endl;
}
```

25.4.3 altova::DayTimeDuration

This class enables you to process XML schema duration types that consist of a day and time part.

Constructors

Name	Description
`DayTimeDuration()`	Initializes a new instance of the `DayTimeDuration` class to an empty value.
`DayTimeDuration(int days, int hours, int minutes, double seconds)`	Initializes a new instance of the `DayTimeDuration` class to the number of days, hours, minutes, and seconds supplied as arguments.
`explicit DayTimeDuration(__int64 value)`	Initializes a new instance of the `DayTimeDuration` class to a duration that consists of as many ticks (100-nanosecond intervals) as supplied in the **value** argument.

Methods

Name	Description
`int Days() const`	Returns the number of days in the current `DayTimeDuration` instance.
`int Hours() const`	Returns the number of hours in the current `DayTimeDuration` instance.
`bool IsNegative() const`	Returns Boolean **true** if the current `DayTimeDuration` instance is negative.
`bool IsPositive() const`	Returns Boolean **true** if the current `DayTimeDuration` instance is positive.
`int Minutes() const`	Returns the number of minutes in the current `DayTimeDuration` instance.
`double Seconds() const`	Returns the number of seconds in the current `DayTimeDuration` instance.
`__int64 Value() const`	Returns the value (in ticks) of the current `DayTimeDuration` instance.

25.4.4 altova::YearMonthDuration

This class enables you to process XML schema duration types that consist of a year and month part.

Constructors

Name	Description
YearMonthDuration()	Initializes a new instance of the YearMonthDuration class to an empty value.
YearMonthDuration(int years, int months)	Initializes a new instance of the YearMonthDuration class to the number of years and months supplied in the **years** and **months** arguments.
explicit YearMonthDuration(int value)	Initializes a new instance of the YearMonthDuration class to a duration that consists of as many ticks (100-nanosecond intervals) as supplied in the **value** argument.

Methods

Name	Description
bool IsNegative() const	Returns Boolean **true** if the current YearMonthDuration instance is negative.
bool IsPositive() const	Returns Boolean **true** if the current YearMonthDuration instance is positive.
int Months() const	Returns the number of months in the current YearMonthDuration instance.
int Value() const	Returns the value (in ticks) of the current YearMonthDuration instance.
int Years()	Returns the number of years in the current YearMonthDuration instance.

25.4.5 altova::meta::Attribute

This class enables you to access schema information about classes generated from attributes. Note that this class is not meant to provide dynamic information about particular instances of an attribute in an XML document. Instead, it enables you to obtain programmatically information about a particular attribute defined in the XML schema.

Methods

Name	Description
SimpleType GetDataType()	Returns the type of the attribute content.
string_type GetLocalName()	Returns the local name of the attribute.
string_type GetNamespaceURI()	Returns the namespace URI of the attribute.
bool IsRequired()	Returns true if the attribute is required.

Operators

Name	Description
`bool operator()`	Returns true if this is not the NULL Attribute.
`bool operator!()`	Returns true if this is the NULL Attribute.

25.4.6 altova::meta::ComplexType

This class enables you to access schema information about classes generated from complex types. Note that this class is not meant to provide dynamic information about particular instances of a complex type in an XML document. Instead, it enables you to obtain programmatically information about a particular complex type defined in the XML schema.

Methods

Name	Description
`Attribute FindAttribute(const char_type* localName, const char_type* namespaceURI)`	Finds the attribute with the specified local name and namespace URI.
`Element FindElement(const char_type* localName, const char_type* namespaceURI)`	Finds the element with the specified local name and namespace URI.
`std::vector<Attribute> GetAttributes()`	Returns a list of all attributes.
`ComplexType GetBaseType()`	Returns the base type of this type.
`SimpleType GetContentType()`	Returns the simple type of the content.
`std::vector<Element> GetElements()`	Returns a list of all elements.
`string_type GetLocalName()`	Returns the local name of the type.
`string_type GetNamespaceURI()`	Returns the namespace URI of the type.

Operators

Name	Description
`bool operator()`	Returns true if this is not the NULL ComplexType.
`bool operator!()`	Returns true if this is the NULL ComplexType.

25.4.7 altova::meta::Element

This class enables you to access information about classes generated from schema elements. Note that this class is not meant to provide dynamic information about particular instances of an element in an XML document. Instead, it enables you to obtain programmatically information about a particular element defined in the XML schema.

Methods

Name	Description
`ComplexType GetDataType()`	Returns the type of the element. Note that this is always a complex type even if declared as simple in the original schema. Use GetContentType() of the returned object to get the simple content type.
`string_type GetLocalName()`	Returns the local name of the element.
`unsigned int GetMaxOccurs()`	Returns the maxOccurs value defined in the schema.
`unsigned int GetMinOccurs()`	Returns the minOccurs value defined in the schema.
`string_type GetNamespaceURI()`	Returns the namespace URI of the element.

Operators

Name	Description
`bool operator()`	Returns true if this is not the NULL Element.
`bool operator!()`	Returns true if this is the NULL Element.

25.4.8 altova::meta::SimpleType

This class enables you to access schema information about classes generated from simple types. Note that this class is not meant to provide dynamic information about particular instances of simple types in an XML document. Instead, it enables you to obtain programmatically information about a particular simple type defined in the XML schema.

Methods

Name	Description
`SimpleType GetBaseType()`	Returns the base type of this type.
`std::vector<string_type> GetEnumerations()`	Returns a list of all enumeration facets.
`unsigned int GetFractionDigits()`	Returns the value of this facet.

Name	Description
`unsigned int GetLength()`	Returns the value of this facet.
`string_type GetLocalName()`	Returns the local name of the type.
`string_type GetMaxExclusive()`	Returns the value of this facet.
`string_type GetMaxInclusive()`	Returns the value of this facet.
`unsigned int GetMaxLength()`	Returns the value of this facet.
`string_type GetMinExclusive()`	Returns the value of this facet.
`string_type GetMinInclusive()`	Returns the value of this facet.
`unsigned int GetMinLength()`	Returns the value of this facet.
`string_type GetNamespaceURI()`	Returns the namespace URI of the type.
`std::vector<string_type> GetPatterns()`	Returns a list of all pattern facets.
`unsigned int GetTotalDigits()`	Returns the value of this facet.
`WhitespaceType GetWhitespace()`	Returns the value of the whitespace facet, which is one of: • Whitespace_Unknown • Whitespace_Preserve • Whitespace_Replace • Whitespace_Collapse

Operators

Name	Description
`bool operator()`	Returns true if this is not the NULL SimpleType.
`bool operator!()`	Returns true if this is the NULL SimpleType.

25.4.9 [YourSchema]::[CDoc]

When code is generated from an XML Schema, the generated code provides a document class with the same name as the schema. This class contains all possible root elements as members, as well as the following methods. Note that, in the method names below, "CDoc" stands for the name of the generated document class itself.

Methods

Name	Description
`static CDoc CreateDocument()`	Creates a new, empty XML document. Must be released using DestroyDocument().
`void DestroyDocument()`	Destroys a document. All references to the document and its nodes are invalidated. This must be called when you finished working with a document.
`static CDoc LoadFromBinary(const std:vector<unsigned char>& xml)`	Loads an XML document from a byte array.
`static CDoc LoadFromFile(const string_type& fileName)`	Loads an XML document from a file.
`static CDoc LoadFromString(const string_type& xml)`	Loads an XML document from a string.
`std::vector<unsigned char> SaveToBinary(bool prettyPrint)`	Saves an XML document to a byte array. When set to true, the `prettyPrint` argument re-formats the XML document for better readability.
`std::vector<unsigned char> SaveToBinary(bool prettyPrint, const string_type & encoding)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding.
`std::vector<unsigned char> SaveToBinary(bool prettyPrint, const string_type & encoding, bool bBigEndian, bool bBOM)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding. Byte order and Unicode byte-order mark can be specified for Unicode encodings.
`void SaveToFile(const string_type & fileName, bool prettyPrint)`	Saves an XML document to a file, with optional "pretty-print" formatting.
`void SaveToFile(const string_type & fileName, bool omitXmlDecl)`	Saves an XML document to a file. If the `omitXmlDecl` argument is set to true, the XML declaration will not be written.
`void SaveToFile(const string_type & fileName, bool omitXmlDecl, const string_type & encoding)`	Saves an XML document to a file with the specified encoding. If the `omitXmlDecl` argument is set to true, the XML declaration will not be written.
`void SaveToFile(const string_type & fileName, bool prettyPrint, bool omitXmlDecl, const string_type & encoding, bool bBigEndian, bool bBOM)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding. Byte order and Unicode byte-order mark can be specified for Unicode encodings.
`void SaveToFile(const`	Saves an XML document to a file, with optional

Name	Description
`string_type & fileName, bool prettyPrint, bool omitXmlDecl, const string_type & encoding, bool bBigEndian, bool bBOM, const string_type & lineend)`	"pretty-print" formatting, with the specified encoding and the specified line end. Byte order and Unicode byte-order mark can be specified for Unicode encodings. This method is only available if you generated the code for the Xerces3 XML library (see Code Generator Options).
`void SaveToFile(const string_type& fileName, bool prettyPrint, bool omitXmlDecl, const string_type & encoding, const string_type & lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding and the specified line end. This method is only available if you generated the code for the Xerces3 XML library (see Code Generator Options).
`void SaveToFile(const string_type & fileName, bool prettyPrint, const string_type & encoding)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding.
`void SaveToFile(const string_type& fileName, bool prettyPrint, const string_type & encoding, bool bBigEndian, bool bBOM)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding. Byte order and Unicode byte-order mark can be specified for Unicode encodings.
`void SaveToFile(const string_type& fileName, bool prettyPrint, const string_type & encoding, bool bBigEndian, bool bBOM, const string_type & lineend)`	Saves an XML document to a file with the specified encoding and the specified line end. Byte order and Unicode byte-order mark can be specified for Unicode encodings. This method is only available if you generated the code for the Xerces3 XML library (see Code Generator Options).
`void SaveToFile(const string_type& fileName, bool prettyPrint, const string_type & encoding, const string_type & lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding and the specified line end. This method is only available if you generated the code for the Xerces3 XML library (see Code Generator Options).
`string_type SaveToString(bool prettyPrint)`	Saves an XML document to a string, with optional "pretty-print" formatting.
`string_type SaveToString(bool prettyPrint, bool omitXmlDecl)`	Saves an XML document to a string, with optional "pretty-print" formatting. If the `omitXmlDecl` argument is set to true, the XML declaration will not be written.
`void SetDTDLocation(const string_type & dtdLocation)`	Adds a DOCTYPE declaration with the specified system ID. A root element must already exist. This

Name	Description
	method is not supported for MSXML, since it is not possible to add a DOCTYPE declaration to a document in memory.
`void SetSchemaLocation(const string_type & schemaLocation)`	Adds an xsi:schemaLocation or xsi:noNamespaceSchemaLocation attribute to the root element. A root element must already exist.

25.4.10 [YourSchema]::MemberAttribute

When code is generated from an XML schema, a class such as this one is created for each member attribute of a type.

Methods

Name	Description
`bool exists()`	Returns true if the attribute exists.
`int GetEnumerationValue()`	Generated for enumeration types only. Returns one of the constants generated for the possible values, or "Invalid" if the value does not match any of the enumerated values in the schema.
`altova::meta::Attribute info()`	Returns an object for querying schema information (see `altova::meta::Attribute`).
`void remove()`	Removes the attribute from its parent element.
`void SetEnumerationValue(int)`	Generated for enumeration types only. Pass one of the constants generated for the possible values to this method to set the value.

25.4.11 [YourSchema]::MemberElement

When code is generated from an XML schema, a class such as this one is created for each member element of a type. In the descriptions below, "MemberType" stands for the name of the member element itself.

Methods

Name	Description
`Iterator<MemberType> all()`	Returns an object for iterating instances of the member element.
`MemberType append()`	Creates a new element and appends it to its parent.

Name	Description
`unsigned int count()`	Returns the count of elements.
`int GetEnumerationValue()`	Generated for enumeration types only. Returns one of the constants generated for the possible values, or **Invalid** if the value does not match any of the enumerated values in the schema.
`bool exists()`	Returns true if at least one element exists.
`MemberType first()`	Returns the first instance of the member element.
`MemberType operator[] (unsigned int index)`	Returns the member element specified by the index.
`altova::meta::Element info()`	Returns an object for querying schema information (see `altova::meta::Element`).
`MemberType last()`	Returns the last instance of the member element.
`void remove()`	Deletes all occurrences of the element from its parent.
`void remove(unsigned int index)`	Deletes the occurrence of the element specified by the index.
`void SetEnumerationValue(int)`	Generated for enumeration types only. Pass one of the constants generated for the possible values to this method to set the value.

25.5 Reference to Generated Classes (C#)

This chapter includes a description of C# classes generated with XMLSpy from a DTD or XML schema (see Generating Code from XML Schemas or DTDs). You can integrate these classes into your code to read, modify, and write XML documents.

Note: The generated code may include other supporting classes, which are not listed here and are subject to modification.

25.5.1 Altova.Types.DateTime

This class enables you to process XML attributes or elements that have date and time types, such as `xs:dateTime`.

Constructors

	Name	Description
	`DateTime(DateTime obj)`	Initializes a new instance of the `DateTime` class to the `DateTime` object supplied as argument.
	`DateTime(System.DateTime newvalue)`	Initializes a new instance of the `DateTime` class to the `System.DateTime` object supplied as argument.
	`DateTime(int year, int month, int day, int hour, int minute, double second, int offsetTZ)`	Initializes a new instance of the `DateTime` class to the year, month, day, hour, minute, second, and timezone offset supplied as arguments.
	`DateTime(int year, int month, int day, int hour, int minute, double second)`	Initializes a new instance of the `DateTime` class to the year, month, day, hour, minute, and second supplied as arguments.
	`DateTime(int year, int month, int day)`	Initializes a new instance of the `DateTime` class to the year, month and day supplied as arguments.

Properties

	Name	Description
	`bool HasTimezone`	Gets a Boolean value which indicates if the `DateTime` has a timezone.
	`static DateTime Now`	Gets a `DateTime` object that is set to the current date and time on this computer.
	`short TimezoneOffset`	Gets or sets the timezone offset, in minutes, of the `DateTime` object.
	`System.DateTime Value`	Gets or sets the value of the `DateTime` object as a `System.DateTime` value.

Methods

	Name	Description
	`int CompareTo(object obj)`	The `DateTime` class implements the `IComparable` interface. This method compares the current instance of `DateTime` to another object and returns an integer that indicates whether the current instance precedes, follows, or occurs in the same position in the sort order as the other object. See also https://msdn.microsoft.com/en-us/library/system.icomparable.compareto(v=vs.110).aspx
	`override bool Equals(object obj)`	Returns **true** if the specified object is equal to the current object; **false** otherwise.
	`System.DateTime GetDateTime(bool correctTZ)`	Returns a `System.DateTime` object from the current `Altova.Types.DateTime` instance. The `correctTZ` Boolean argument specifies whether the time of the returned object must be adjusted according to the timezone of the current `Altova.Types.DateTime` instance.
	`override int GetHashCode()`	Returns the hash code of the current instance.
	`int GetWeekOfMonth()`	Returns the number of the week in month as an integer.

	`static DateTime Parse(string s)`	Creates a `DateTime` object from the string supplied as argument. For example, the following sample string values would be converted successfully to a `DateTime` object:
		
		`2015-01-01T23:23:23`
		`2015-01-01`
		`2015-11`
		`23:23:23`
		
		An exception is raised if the string cannot be converted to a `DateTime` object.
		
		Note that this method is static and can only be called on the `Altova.Types.DateTime` class itself, not on an instance of the class.

	`static DateTime Parse(string s,DateTimeFormat format)`	Creates a `DateTime` object from a string, using the format supplied as argument. For the list of possible formats, see Altova.Types.DateTimeFormat. An exception is raised if the string cannot be converted to a `DateTime` object. Note that this method is static and can only be called on the `Altova.Types.DateTime` class itself, not on an instance of the class.
	`override string ToString()`	Converts the `DateTime` object to a string.
	`string`	Converts the `DateTime` object to a string, using the

	Name	Description
	ToString(DateTimeFormat format)	format supplied as argument. For the list of possible formats, see Altova.Types.DateTimeFormat.

Operators

Name	Description
!=	Determines if DateTime a is not equal to DateTime b.
<	Determines if DateTime a is less than DateTime b.
<=	Determines if DateTime a is less than or equal to DateTime b.
==	Determines if DateTime a is equal to DateTime b.
>	Determines if DateTime a is greater than DateTime b.
>=	Determines if DateTime a is greater than or equal to DateTime b.

Examples

Before using the following code listings in your program, ensure the Altova types are imported:

```
using Altova.Types;
```

The following code listing illustrates various ways to create DateTime objects:

```
protected static void DateTimeExample1()
{
        // Create a DateTime object from the current system time
        Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);
        Console.WriteLine("The current time is:" + dt.ToString());

        // Create an Altova DateTime object from parts (no timezone)
        Altova.Types.DateTime dt1 = new Altova.Types.DateTime(2015,10,12,10,50,33);
        Console.WriteLine("My custom time is :" + dt1.ToString());

        // Create an Altova DateTime object from parts (with UTC+60 minutes timezone)
        Altova.Types.DateTime dt2 = new Altova.Types.DateTime(2015,10,12,10,50,33,60);
        Console.WriteLine("My custom time with timezone is :" + dt2.ToString());

        // Create an Altova DateTime object by parsing a string
        Altova.Types.DateTime dt3 = Altova.Types.DateTime.Parse("2015-01-01T23:23:23");
        Console.WriteLine("Time created from string:" + dt3.ToString());

        // Create an Altova DateTime object by parsing a string formatted as schema date
        Altova.Types.DateTime dt4 = Altova.Types.DateTime.Parse("2015-01-01",
DateTimeFormat.W3_date);
        Console.WriteLine("Time created from string formatted as schema date:" +
dt4.ToString());
```

```
}
```

The following code listing illustrates various ways to format `DateTime` objects:

```
protected static void DateTimeExample2()
{
        // Create a DateTime object from the current system time
        Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);

        // Output the unformatted DateTime
        Console.WriteLine("Unformatted time:" + dt.ToString());

        // Output this DateTime formatted using various formats
        Console.WriteLine("S_DateTime:     " + dt.ToString(DateTimeFormat.S_DateTime));
        Console.WriteLine("S_Days:         " + dt.ToString(DateTimeFormat.S_Days));
        Console.WriteLine("S_Seconds:      " + dt.ToString(DateTimeFormat.S_Seconds));
        Console.WriteLine("W3_date:        " + dt.ToString(DateTimeFormat.W3_date));
        Console.WriteLine("W3_dateTime:    " + dt.ToString(DateTimeFormat.W3_dateTime));
        Console.WriteLine("W3_gDay:        " + dt.ToString(DateTimeFormat.W3_gDay));
        Console.WriteLine("W3_gMonth:      " + dt.ToString(DateTimeFormat.W3_gMonth));
        Console.WriteLine("W3_gMonthDay:   " +
dt.ToString(DateTimeFormat.W3_gMonthDay));
        Console.WriteLine("W3_gYear:       " + dt.ToString(DateTimeFormat.W3_gYear));
        Console.WriteLine("W3_gYearMonth:  " +
dt.ToString(DateTimeFormat.W3_gYearMonth));
        Console.WriteLine("W3_time:        " + dt.ToString(DateTimeFormat.W3_time));
}
```

25.5.2 Altova.Types.DateTimeFormat

The `DateTimeFormat` enum type has the following constant values:

Value	Description	Example
S_DateTime	Formats the value as standard dateTime, with a precision of a ten-millionth of a second, including timezone.	`2015-11-12 12:19:03.9019132 +01:00`
S_Days	Formats the value as number of days elapsed since the UNIX epoch.	`735913.631897345108796296263`
S_Seconds	Formats the value as number of seconds elapsed since the UNIX epoch, with a precision of a ten-millionth of a second.	`63582937678.0769062`
W3_date	Formats the value as schema date.	`2015-11-12`
W3_dateTime	Formats the value as schema dateTime.	`2015-11-12T15:12:14.5194251`
W3_gDay	Formats the value as schema gDay.	`---12` (assuming that the date is 12th of

Value	Description	Example
		the month)
W3_gMonth	Formats the value as schema gMonth.	`--11` (assuming that the month is November)
W3_gMonthDay	Formats the value as schema gMonthDay.	`--11-12` (assuming that the date is 12th of November)
W3_gYear	Formats the value as schema gYear.	`2015` (assuming that the year is 2015)
W3_gYearMonth	Formats the value as schema gYearMonth.	`2015-11` (assuming that the year is 2015 and the month is November)
W3_time	Formats the value as schema time, with a precision of a ten-millionth of a second.	`15:19:07.5582719`

25.5.3 Altova.Types.Duration

This class enables you to process XML attributes or elements of type `xs:duration`.

Constructors

	Name	Description
	`Duration(Duration obj)`	Initializes a new instance of the `Duration` class to the `Duration` object supplied as argument.
	`Duration(System.TimeSpan newvalue)`	Initializes a new instance of the `Duration` class to the `System.TimeSpan` object supplied as argument.
	`Duration(long ticks)`	Initializes a new instance of the `Duration` class to the number of ticks supplied as argument.
	`Duration(int newyears, int newmonths, int days, int hours, int minutes, int seconds, double partseconds, bool bnegative)`	Initializes a new instance of the `Duration` class to a duration built from parts supplied as arguments.

Properties

	Name	Description
	`int Months`	Gets or sets the number of months of the current instance of `Duration`.
	`System.TimeSpan Value`	Gets or sets the value (as `System.TimeSpan`) of the current instance of `Duration`.

	Name	Description
	`int Years`	Gets or sets the number of years of the current instance of `Duration`.

Methods

	Name	Description
	`override bool Equals(object other)`	Returns **true** if the specified object is equal to the current object; **false** otherwise.
	`override int GetHashCode()`	Returns the hash code of the current instance.
	`bool IsNegative()`	Returns **true** if the current instance of `Duration` represents a negative duration.
	`static Duration Parse(string s, ParseType pt)`	Returns an `Altova.Types.Duration` object parsed from the string supplied as argument, using the parse type supplied as argument. Valid parse type values: **DURATION** Parse duration assuming that year, month, day, as well as time duration parts exist. **YEARMONTH** Parse duration assuming that only year and month parts exist. **DAYTIME** Parse duration assuming that only the day and time parts exist. Note that this method is static and can only be called on the class itself, not on an instance of the class.
	`override string ToString()`	Converts the current `Duration` instance to string. For example, a time span of 3 hours, 4 minutes, and 5 seconds would be converted to "`PT3H4M5S`".
	`string ToYearMonthString()`	Converts the current `Duration` instance to string, using the "Year and Month" parse type.

Operators

Name	Description
`!=`	Determines if `Duration` a is not equal to `Duration` b.
`==`	Determines if `Duration` a is equal to `Duration` b.

Examples

Before using the following code listings in your program, ensure the Altova types are imported:

```
using Altova.Types;
```

The following code listing illustrates various ways to create `Duration` objects:

```
protected static void DurationExample1()
{
        // Create a new time span of 3 hours, 4 minutes, and 5 seconds
        System.TimeSpan ts = new TimeSpan(3,4,5);
        // Create a Duration from the time span
        Duration dr = new Duration(ts);
        // The output is: PT3H4M5S
        Console.WriteLine("Duration created from TimeSpan:" + dr.ToString());

        // Create a negative Altova.Types.Duration from 6 years, 5 months, 4 days, 3 hours,
    // 2 minutes, 1 second, and 33 of a second
        Duration dr1 = new Duration(6,5,4,3,2,1,33,true);
        // The output is: -P6Y5M4DT3H2M1.33S
        Console.WriteLine("Duration created from parts:" + dr1.ToString());

        // Create a Duration from a string using the DAYTIME parse type
        Duration dr2 = Altova.Types.Duration.Parse("-P4DT3H2M1S",
Duration.ParseType.DAYTIME);
        // The output is -P4DT3H2M1S
        Console.WriteLine("Duration created from string:" + dr2.ToString());

        // Create a duration from ticks
        Duration dr3 = new Duration(System.DateTime.UtcNow.Ticks);
        // Output the result
        Console.WriteLine("Duration created from ticks:" + dr3.ToString());
}
```

The following code listing illustrates getting values from `Duration` objects:

```
protected static void DurationExample2()
{
        // Create a negative Altova.Types.Duration from 6 years, 5 months, 4 days, 3 hours,
    // 2 minutes, 1 second, and 33 of a second
        Duration dr = new Duration(6,5,4,3,2,1,33,true);
        // The output is: -P6Y5M4DT3H2M1.33S
        Console.WriteLine("The complete duration is:" + dr.ToString());

        // Get only the year and month part as string
        string dr1 = dr.ToYearMonthString();
        Console.WriteLine("The YEARMONTH part is:" + dr1);

        // Get the number of years in duration
        Console.WriteLine("Years:" + dr.Years);

        // Get the number of months in duration
        Console.WriteLine("Months:" + dr.Months);
}
```

25.5.4 Altova.Xml.Meta.Attribute

This class enables you to access schema information about classes generated from attributes. Note that this class is not meant to provide dynamic information about particular instances of an attribute in an XML document. Instead, it enables you to obtain programmatically information

about a particular attribute defined in the XML schema.

Properties

	Name	Description
	`SimpleType DataType`	Returns the type of the attribute content.
	`string LocalName`	Returns the local name of the attribute.
	`string NamespaceURI`	Returns the namespace URI of the attribute.
	`XmlQualifiedName QualifiedName`	Returns the qualified name of the attribute.
	`bool Required()`	Returns true if the attribute is required.

25.5.5 Altova.Xml.Meta.ComplexType

This class enables you to access schema information about classes generated from complex types. Note that this class is not meant to provide dynamic information about particular instances of a complex type in an XML document. Instead, it enables you to obtain programmatically information about a particular complex type defined in the XML schema.

Properties

	Name	Description
	`Attribute[] Attributes`	Returns a list of all attributes.
	`ComplexType BaseType`	Returns the base type of this type or null if no base type exists.
	`SimpleType ContentType`	Returns the simple type of the content.
	`Element[] Elements`	Returns a list of all elements.
	`string LocalName`	Returns the local name of the type.
	`string NamespaceURI`	Returns the namespace URI of the type.
	`XmlQualifiedName QualifiedName`	Returns the qualified name of this type.

Methods

	Name	Description
	`ComplexType BaseType`	Returns the base type of this type.
	`bool Equals(obj)`	Checks if two info objects refer to the same type, based on qualified name comparison. Returns true if the type has the same qualified name.
	`Attribute FindAttribute(string`	Finds the attribute with the specified local name and namespace URI.

	Name	Description
	`localName, string namespaceURI)`	
	`Element FindElement(string localName, string namespaceURI)`	Finds the element with the specified local name and namespace URI.

25.5.6 Altova.Xml.Meta.Element

This class enables you to access information about classes generated from schema elements. Note that this class is not meant to provide dynamic information about particular instances of an element in an XML document. Instead, it enables you to obtain programmatically information about a particular element defined in the XML schema.

Properties

	Name	Description
	`ComplexType DataType`	Returns the type of the element. Note that this is always a complex type even if declared as simple in the original schema. Use the `ContentType` property of the returned object to get the simple content type.
	`string LocalName`	Returns the local name of the element.
	`int MaxOccurs`	Returns the `maxOccurs` value defined in the schema.
	`int MinOccurs`	Returns the `minOccurs` value defined in the schema.
	`string NamespaceURI`	Returns the namespace URI of the element.
	`XmlQualifiedName QualifiedName`	Returns the qualified name of the element.

25.5.7 Altova.Xml.Meta.SimpleType

This class enables you to access schema information about classes generated from simple types. Note that this class is not meant to provide dynamic information about particular instances of simple types in an XML document. Instead, it enables you to obtain programmatically information about a particular simple type defined in the XML schema.

Properties

	Name	Description
	`SimpleType BaseType`	Returns the base type of this type.
	`string[] Enumerations`	Returns a list of all enumeration facets.
	`int FractionDigits`	Returns the value of this facet.
	`int Length`	Returns the value of this facet.
	`string LocalName`	Returns the local name of the type.

	Name	Description
	`string MaxExclusive`	Returns the value of this facet.
	`string MaxInclusive`	Returns the value of this facet.
	`int MaxLength`	Returns the value of this facet.
	`string MinExclusive`	Returns the value of this facet.
	`string MinInclusive`	Returns the value of this facet.
	`int MinLength`	Returns the value of this facet.
	`string NamespaceURI`	Returns the namespace URI of the type.
	`string[] Patterns`	Returns the pattern facets, or null if no patterns are specified.
	`XmlQualifiedName QualifiedName`	Returns the qualified name of this type.
	`int TotalDigits`	Returns the value of this facet.
	`WhitespaceType Whitespace`	Returns the whitespace normalization facet.

25.5.8 [YourSchema].[Doc]

When code is generated from an XML Schema, the generated code provides a document class with the same name as the schema. This class contains all possible root elements as members, as well as the members listed below. Note that, in the method names below, "Doc" stands for the name of the generated document class itself.

Methods

	Name	Description
	`static Doc CreateDocument()`	Creates a new, empty XML document.
	`static Doc CreateDocument(string encoding)`	Creates a new, empty XML document, with encoding of type "encoding".
	`static Doc LoadFromBinary(byte[] binary)`	Loads an XML document from a byte array.
	`static Doc LoadFromFile(string filename)`	Loads an XML document from a file.
	`static Doc LoadFromString(string xmlstring)`	Loads an XML document from a string.
	`byte[] SaveToBinary(bool prettyPrint)`	Saves an XML document to a byte array, with optional "pretty-print" formatting.
	`byte[]`	Saves an XML document to a byte array, with

	Name	Description
	`SaveToBinary(bool prettyPrint, string encoding)`	optional "pretty-print" formatting, with the specified encoding.
	`byte[] SaveToBinary(bool prettyPrint, string encoding, bool bBigEndian, bool bBOM)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding, byte order, and BOM (Byte Order Mark).
	`void SaveToFile(string fileName, bool prettyPrint)`	Saves an XML document to a file, with optional "pretty-print" formatting.
	`void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl)`	Saves an XML document to a file, with optional "pretty-print" formatting. When `omitXmlDecl` is true, the XML declaration will not be written.
	`void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl, string encoding)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding. When `omitXmlDecl` is true, the XML declaration will not be written.
	`void SaveToFile(string fileName, bool prettyPrint, string encoding, string lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding, and line ending character(s).
	`void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl, string encoding, string lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding, and line ending character(s). When `omitXmlDecl` is true, the XML declaration will not be written.
	`void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl, string encoding, bool bBigEndian, bool bBOM, string lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding, byte order, BOM (Byte Order Mark), and line ending character(s). When `omitXmlDecl` is true, the XML declaration will not be written.
	`void SaveToFileWithLineEnd(string fileName, bool prettyPrint, bool omitXmlDecl, string lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, and line ending character(s). When `omitXmlDecl` is true, the XML declaration will not be written.
	`string SaveToString(bool prettyPrint)`	Saves an XML document to a file, with optional "pretty-print" formatting.
	`string SaveToString(bool prettyPrint, bool`	Saves an XML document to a file, with optional "pretty-print" formatting. When `omitXmlDecl` is true, the XML declaration will not be written.

	Name	Description
	omitXmlDecl)	
	void SetDTDLocation(string dtdLocation)	Adds a DOCTYPE declaration with the specified system ID. A root element must already exist.
	void SetSchemaLocation(string schemaLocation)	Adds an xsi:schemaLocation or xsi:noNamespaceSchemaLocation attribute to the root element. A root element must already exist.

25.5.9 [YourSchemaType].MemberAttribute

When code is generated from an XML schema, a class is created for each member attribute of a type. In the descriptions below, "AttributeType" stands for the type of the member attribute itself.

Methods

	Name	Description
	bool Exists()	Returns true if the attribute exists.
	void Remove()	Removes the attribute from its parent element.

Properties

	Name	Description
	int EnumerationValue	Generated for enumeration types only. Sets or gets the attribute value using one of the constants generated for the possible values. Returns **Invalid** if the value does not match any of the enumerated values in the schema.
	Altova.Xml.Meta.AttributeInfo	Returns an object for querying schema information (see Altova.Xml.Meta.Attribute).
	AttributeType Value	Sets or gets the attribute value.

25.5.10 [YourSchemaType].MemberElement

When code is generated from an XML schema, a class with the following members is created for each member element of a type. The class implements the standard System.Collections.IEnumerable interface, so it can be used with the foreach statement.

In the descriptions below, "MemberType" stands for the type of the member element itself.

Methods

	Name	Description
	`MemberType Append()`	Creates a new element and appends it to its parent.
	`MemberType At(int index)`	Returns the member element specified by the index.
	`System.Collections.IEnumerator GetEnumerator()`	Returns an object for iterating instances of the member element.
	`void Remove()`	Deletes all occurrences of the element from its parent.
	`void RemoveAt(int index)`	Deletes the occurrence of the element specified by the index.

Properties

	Name	Description
	`int Count`	Returns the count of elements.
	`int EnumerationValue`	Generated for enumeration types only. Sets or gets the element value using one of the constants generated for the possible values. Returns **Invalid** if the value does not match any of the enumerated values in the schema.
	`bool Exists`	Returns true if at least one element exists.
	`MemberType First`	Returns the first instance of the member element.
	`Altova.Xml.Meta.ElementInfo`	Returns an object for querying schema information (see `Altova.Xml.Meta.Element`).
	`MemberType Last`	Returns the last instance of the member element.
	`MemberType this[int index]`	Returns the member element specified by the index.
	`MemberType Value`	Sets or gets the element content (only generated if element can have mixed or simple content).

25.6 Reference to Generated Classes (Java)

This chapter includes a description of Java classes generated with XMLSpy from a DTD or XML schema (see Generating Code from XML Schemas or DTDs). You can integrate these classes into your code to read, modify, and write XML documents.

Note: The generated code may include other supporting classes, which are not listed here and are subject to modification.

25.6.1 com.altova.types.DateTime

This class enables you to process XML attributes or elements that have date and time types, such as `xs:dateTime`.

Constructors

	Name	Description
C	`public DateTime()`	Initializes a new instance of the `DateTime` class to an empty value.
C	`public DateTime(DateTime newvalue)`	Initializes a new instance of the `DateTime` class to the `DateTime` value supplied as argument.
C	`public DateTime(int newyear, int newmonth, int newday, int newhour, int newminute, int newsecond, double newpartsecond, int newoffsetTZ)`	Initializes a new instance of the `DateTime` class to the year, month, day, hour, minute, second, the fractional part of the second, and timezone supplied as arguments. The fractional part of the second `newpartsecond` must be between 0 and 1. The timezone offset `newoffsetTZ` can be either positive or negative and is expressed in minutes.
C	`public DateTime(int newyear, int newmonth, int newday, int newhour, int newminute, int newsecond, double newpartsecond)`	Initializes a new instance of the `DateTime` class to the year, month, day, hour, minute, second, and the fractional part of a second supplied as arguments.
C	`public DateTime(int newyear, int newmonth, int newday)`	Initializes a new instance of the `DateTime` class to the year, month, and day supplied as arguments.
C	`public DateTime(Calendar newvalue)`	Initializes a new instance of the `DateTime` class to the `java.util.Calendar` value supplied as argument.

Methods

	Name	Description
S	`static DateTime now()`	Returns the current time as a `DateTime` object.
S	`static DateTime parse(String s)`	Returns a `DateTime` object parsed from the string

	Name	Description

		value supplied as argument. For example, the following sample string values would be converted successfully to a `DateTime` object:
		
		`2015-11-24T12:54:47.969+01:00`
		`2015-11-24T12:54:47`
		`2015-11-24`

●	`int getDay()`	Returns the day of the current `DateTime` instance.
●	`int getHour()`	Returns the hour of the current `DateTime` instance.
●	`int getMillisecond()`	Returns the millisecond of the current `DateTime` instance, as an integer value.
●	`int getMinute()`	Returns the minute of the current `DateTime` instance.
●	`int getMonth()`	Returns the month of the current `DateTime` instance.
●	`double getPartSecond()`	Returns the fractional part of the second of the current `DateTime` instance, as a `double` value. The return value is greater than zero and smaller than one, for example: `0.313`
●	`int getSecond()`	Returns the second of the current `DateTime` instance.
●	`int getTimezoneOffset()`	Returns the timezone offset, in minutes, of the current `DateTime` instance. For example, the timezone "UTC-01:00" would be returned as: `-60`
●	`Calendar getValue()`	Returns the current `DateTime` instance as a `java.util.Calendar` value.
●	`int getWeekday()`	Returns the day in week of the current `DateTime` instance. Values range from 0 through 6, where 0 is Monday (ISO-8601).
●	`int getYear()`	Returns the year of the current `DateTime` instance.
●	`int hasTimezone()`	Returns information about the timezone of the current `DateTime` instance. Possible return values are: `CalendarBase.TZ_MISSING` — A timezone offset is not defined. `CalendarBase.TZ_UTC` — The timezone is UTC. `CalendarBase.TZ_OFFSET` — A timezone offset has been defined.
●	`void setDay( int nDay )`	Sets the day of the current `DateTime` instance to the value supplied as argument.
●	`void`	Sets the timezone information of the current

	Name	Description
	`setHasTimezone( int nHasTZ )`	`DateTime` instance to the value supplied as argument. This method can be used to strip the timezone information or set the timezone to UTC (Coordinated Universal Time). Valid values for the `nHasTZ` argument: `CalendarBase.TZ_MISSING` Set the timezone offset to undefined. `CalendarBase.TZ_UTC` Set the timezone to UTC. `CalendarBase.TZ_OFFSET` If the current object has a timezone offset, leave it unchanged.
●	`void setHour( int nHour )`	Sets the hour of the current `DateTime` instance to the value supplied as argument.
●	`void setMinute( int nMinute )`	Sets the minute of the current `DateTime` instance to the value supplied as argument.
●	`void setMonth( int nMonth )`	Sets the month of the current `DateTime` instance to the value supplied as argument.
●	`void setPartSecond( double nPartSecond )`	Sets the fractional part of the second of the current `DateTime` instance to the value supplied as argument.
●	`void setSecond( int nSecond )`	Sets the second of the current `DateTime` instance to the value supplied as argument.
●	`void setTimezoneOffset( int nOffsetTZ )`	Sets the timezone offset of the current `DateTime` instance to the value supplied as argument. The value `nOffsetTZ` must be an integer (positive or negative) and must be expressed in minutes.
●	`void setYear( int nYear )`	Sets the year of the current `DateTime` instance to the value supplied as argument.
●	`String toString()`	Returns the string representation of the current `DateTime` instance, for example: `2015-11-24T15:50:56.968+01:00`

Examples

Before using the following code listings in your program, ensure the Altova types are imported:

```
import com.altova.types.*;
```

The following code listing illustrates various ways to create `DateTime` objects:

```
protected static void DateTimeExample1()
{
        // Initialize a new instance of the DateTime class to the current time
```

```
        DateTime dt = new DateTime(DateTime.now());
        System.out.println("DateTime created from current date and time:" +
dt.toString());

        // Initialize a new instance of the DateTime class by supplying the parts
        DateTime dt1 = new DateTime(2015, 11, 23, 14, 30, 24, .459);
        System.out.println("DateTime from parts (no timezone):" + dt1.toString());

        // Initialize a new instance of the DateTime class by supplying the parts
        DateTime dt2 = new DateTime(2015, 11, 24, 14, 30, 24, .459, -60);
        System.out.println("DateTime from parts (with negative timezone):" +
dt2.toString());

        // Initialize a new instance of the DateTime class by parsing a string value
DateTime dt3 = DateTime.parse("2015-11-24T12:54:47.969+01:00");

        System.out.println("DateTime parsed from string:" + dt3.toString());

}
```

The following code listing illustrates getting values from `DateTime` objects:

```
protected static void DateTimeExample2()
   {
      // Initialize a new instance of the DateTime class to the current time
      DateTime dt = new DateTime(DateTime.now());

      // Output the formatted year, month, and day of this DateTime instance
      String str1 = String.format("Year: %d; Month: %d; Day: %d;",
dt.getYear(), dt.getMonth(), dt.getDay());
      System.out.println(str1);

      // Output the formatted hour, minute, and second of this DateTime
instance
      String str2 = String.format("Hour: %d; Minute: %d; Second: %d;",
dt.getHour(), dt.getMinute(), dt.getSecond());
      System.out.println(str2);

      // Return the timezone (in minutes) of this DateTime instance
      System.out.println("Timezone: " + dt.getTimezoneOffset());

      // Get the DateTime as a java.util.Calendar value
      java.util.Calendar dt_java = dt.getValue();
      System.out.println("" + dt_java.toString());

      // Return the day of week of this DateTime instance
      System.out.println("Weekday: " + dt.getWeekday());

      // Check whether the DateTime instance has a timezone defined
      switch(dt.hasTimezone())
      {
         case CalendarBase.TZ_MISSING:
            System.out.println("No timezone.");
            break;
         case CalendarBase.TZ_UTC:
            System.out.println("The timezone is UTC.");
            break;
         case CalendarBase.TZ_OFFSET:
            System.out.println("This object has a timezone.");
```

```
            break;
         default:
            System.out.println("Unable to determine whether a timezone is
defined.");
            break;
      }
   }
```

The following code listing illustrates changing the timezone offset of a `DateTime` object:

```
protected static void DateTimeExample3()
{
        // Create a new DateTime object with timezone -0100 UTC
        DateTime dt = new DateTime(2015, 11, 24, 14, 30, 24, .459, -60);
        // Output the value before the change
        System.out.println("Before:" + dt.toString());
        // Change the offset to +0100 UTC
        dt.setTimezoneOffset(60);
        // Output the value after the change
        System.out.println("After: " + dt.toString());
}
```

25.6.2 com.altova.types.Duration

This class enables you to process XML attributes or elements of type `xs:duration`.

Constructors

	Name	Description
C	`Duration(Duration newvalue)`	Initializes a new instance of the `Duration` class to the `Duration` object supplied as argument.
C	`Duration(int newyear, int newmonth, int newday, int newhour, int newminute, int newsecond, double newpartsecond, boolean newisnegative)`	Initializes a new instance of the `Duration` class to a duration built from parts supplied as arguments.

Methods

	Name	Description
S	`static Duration getFromDayTime( int newday, int newhour, int newminute, int newsecond, double newpartsecond )`	Returns a `Duration` object created from the number of days, hours, minutes, seconds, and fractional second parts supplied as argument.
S	`static Duration getFromYearMonth( int newyear, int newmonth )`	Returns a `Duration` object created from the number of years and months supplied as argument.

	Name	Description
S	`static Duration parse( String s )`	Returns a `Duration` object created from the string supplied as argument. For example, the string `-P1Y1M1DT1H1M1.333S` can be used to create a negative duration of one year, one month, one day, one hour, one minute, one second, and 0.333 fractional parts of a second. To create a negative duration, append the minus sign (-) to the string.
S	`static Duration parse( String s, ParseType pt )`	Returns a `Duration` object created from the string supplied as argument, using a specific parse format. The parse format can be any of the following: **ParseType.DAYTIME** Must be used when the string **s** consists of any of the following: days, hours, minutes, seconds, fractional second parts, for example `-P4DT4H4M4.774S`. **ParseType.DURATION** Must be used when the string **s** consists of any of the following: years, months, days, hours, minutes, seconds, fractional second parts, for example `P1Y1M1DT1H1M1.333S`. **ParseType.YEARMONTH** Must be used when the string **s** consists of any of the following: years, months. For example: `P3Y2M`.
	`int getDay()`	Returns the number of days in the current `Duration` instance.
	`long getDayTimeValue()`	Returns the day and time value (in milliseconds) of the current `Duration` instance. Years and months are ignored.
	`int getHour()`	Returns the number of hours in the current `Duration` instance.
	`int getMillisecond()`	Returns the number of milliseconds in the current `Duration` instance.
	`int getMinute()`	Returns the number of minutes in the current `Duration` instance.
	`int getMonth()`	Returns the number of months in the current `Duration` instance.
	`double getPartSecond()`	Returns the number of fractional second parts in the current `Duration` instance.
	`int getSecond()`	Returns the number of seconds in the current `Duration` instance.
	`int getYear()`	Returns the number of years in the current `Duration`

	Name	Description
		instance.
●	`int getYearMonthValue()`	Returns the year and month value (in months) of the current `Duration` instance. Days, hours, seconds, and milliseconds are ignored.
●	`boolean isNegative()`	Returns Boolean **true** if the current `Duration` instance is positive.
●	`void setDayTimeValue(long l)`	Sets the duration to the number of milliseconds supplied as argument, affecting only the day and time part of the duration.
●	`void setNegative( boolean isnegative )`	Converts the current `Duration` instance to a negative duration.
●	`void setYearMonthValue(int l)`	Sets the duration to the number of months supplied as argument. Only the years and months part of the duration is affected.
●	`String toString()`	Returns the string representation of the current `Duration` instance, for example: `-P4DT4H4M4.774S`
●	`String toYearMonthString()`	Returns the string representation of the YearMonth part of the current `Duration` instance, for example: `P1Y2M`

Examples

Before using the following code listings in your program, ensure the Altova types are imported:

```
import com.altova.types.*;
import com.altova.types.Duration.ParseType;
```

The following code listing illustrates various ways to create `Duration` objects:

```
protected static void ExampleDuration()
{
        // Create a negative duration of 1 year, 1 month, 1 day, 1 hour, 1 minute, 1 second,
        // and 0.333 fractional second parts
        Duration dr = new Duration(1, 1, 1, 1, 1, 1, .333, true);

        // Create a duration from an existing Duration object
        Duration dr1 = new Duration(dr);

        // Create a duration of 4 days, 4 hours, 4 minutes, 4 seconds, .774 fractional second parts
        Duration dr2 = Duration.getFromDayTime(4, 4, 4, 4, .774);

        // Create a duration of 3 years and 2 months
        Duration dr3 = Duration.getFromYearMonth(3, 2);

        // Create a duration from a string
```

```
        Duration dr4 = Duration.parse("-P4DT4H4M4.774S");

        // Create a duration from a string, using specific parse formats
        Duration dr5 = Duration.parse("-P1Y1M1DT1H1M1.333S",
ParseType.DURATION);
        Duration dr6 = Duration.parse("P3Y2M", ParseType.YEARMONTH);
        Duration dr7 = Duration.parse("-P4DT4H4M4.774S", ParseType.DAYTIME);
}
```

The following code listing illustrates getting and setting the value of `Duration` objects:

```
protected static void DurationExample2()
{
        // Create a duration of 1 year, 2 month, 3 days, 4 hours, 5 minutes, 6 seconds,
    // and 333 milliseconds
        Duration dr = new Duration(1, 2, 3, 4, 5, 6, .333, false);
        // Output the number of days in this duration
        System.out.println(dr.getDay());

        // Create a positive duration of one year and 333 milliseconds
        Duration dr1 = new Duration(1, 0, 0, 0, 0, 0, .333, false);
        // Output the day and time value in milliseconds
        System.out.println(dr1.getDayTimeValue());

        // Create a positive duration of 1 year, 1 month, 1 day, 1 hour, 1 minute, 1 second,
    // and 333 milliseconds
        Duration dr2 = new Duration(1, 1, 1, 1, 1, 1, .333, false);
        // Output the year and month value in months
        System.out.println(dr2.getYearMonthValue());

        // Create a positive duration of 1 year and 1 month
        Duration dr3 = new Duration(1, 1, 0, 0, 0, 0, 0, false);
        // Output the value
        System.out.println("The duration is now:" + dr3.toString());
        // Set the DayTime part of duration to 1000 milliseconds
        dr3.setDayTimeValue(1000);
        // Output the value
        System.out.println("The duration is now:" + dr3.toString());
        // Set the YearMonth part of duration to 1 month
        dr3.setYearMonthValue(1);
        // Output the value
        System.out.println("The duration is now:" + dr3.toString());
        // Output the year and month part of the duration
        System.out.println("The YearMonth part of the duration is:" +
dr3.toYearMonthString());
}
```

25.6.3 com.altova.xml.meta.Attribute

This class enables you to access schema information about classes generated from attributes. Note that this class is not meant to provide dynamic information about particular instances of an attribute in an XML document. Instead, it enables you to obtain programmatically information about a particular attribute defined in the XML schema.

Methods

	Name	Description
	`SimpleType getDataType()`	Returns the type of the attribute content.
	`String getLocalName()`	Returns the local name of the attribute.
	`String getNamespaceURI()`	Returns the namespace URI of the attribute.
	`boolean isRequired()`	Returns true if the attribute is required.

25.6.4 com.altova.xml.meta.ComplexType

This class enables you to access schema information about classes generated from complex types. Note that this class is not meant to provide dynamic information about particular instances of a complex type in an XML document. Instead, it enables you to obtain programmatically information about a particular complex type defined in the XML schema.

Methods

	Name	Description
	`Attribute findAttribute(String localName, String namespaceURI)`	Finds the attribute with the specified local name and namespace URI.
	`Element findElement(String localName, String namespaceURI)`	Finds the element with the specified local name and namespace URI.
	`Attribute[] GetAttributes()`	Returns a list of all attributes.
	`ComplexType getBaseType()`	Returns the base type of this type.
	`SimpleType getContentType()`	Returns the simple type of the content.
	`Element[] GetElements()`	Returns a list of all elements.
	`String getLocalName()`	Returns the local name of the type.
	`String getNamespaceURI()`	Returns the namespace URI of the type.

25.6.5 com.altova.xml.meta.Element

This class enables you to access information about classes generated from schema elements. Note that this class is not meant to provide dynamic information about particular instances of an element in an XML document. Instead, it enables you to obtain programmatically information about a particular element defined in the XML schema.

Methods

	Name	Description
	`ComplexType getDataType()`	Returns the type of the element. Note that this is always a complex type even if declared as simple in the original schema. Use `getContentType()` of the returned object to get the simple content type.
	`String getLocalName()`	Returns the local name of the element.
	`int getMaxOccurs()`	Returns the maxOccurs value defined in the schema.
	`int getMinOccurs()`	Returns the minOccurs value defined in the schema.
	`String getNamespaceURI()`	Returns the namespace URI of the element.

25.6.6 com.altova.xml.meta.SimpleType

This class enables you to access schema information about classes generated from simple types. Note that this class is not meant to provide dynamic information about particular instances of simple types in an XML document. Instead, it enables you to obtain programmatically information about a particular simple type defined in the XML schema.

Methods

	Name	Description
	`SimpleType getBaseType()`	Returns the base type of this type.
	`String[] getEnumerations()`	Returns an array of all enumeration facets.
	`int getFractionDigits()`	Returns the value of this facet.
	`int getLength()`	Returns the value of this facet.
	`String getLocalName()`	Returns the local name of the type.
	`String getMaxExclusive()`	Returns the value of this facet.
	`String getMaxInclusive()`	Returns the value of this facet.
	`int getMaxLength()`	Returns the value of this facet.
	`String getMinExclusive()`	Returns the value of this facet.
	`String getMinInclusive()`	Returns the value of this facet.
	`int getMinLength()`	Returns the value of this facet.
	`String getNamespaceURI()`	Returns the namespace URI of the type.
	`String[] getPatterns()`	Returns an array of all pattern facets.

	Name	Description
	`int getTotalDigits()`	Returns the value of this facet.
	`int getWhitespace()`	Returns the value of the whitespace facet, which is one of: `com.altova.typeinfo.WhitespaceType.Whitespace_Unknown` `com.altova.typeinfo.WhitespaceType.Whitespace_Preserve` `com.altova.typeinfo.WhitespaceType.Whitespace_Replace` `com.altova.typeinfo.WhitespaceType.Whitespace_Collapse`

25.6.7 com.[YourSchema].[Doc]

When code is generated from an XML Schema, the generated code provides a document class with the same name as the schema. This class contains all possible root elements as members, as well as the members listed below. Note that, in the method names below, "Doc" stands for the name of the generated document class itself.

Methods

	Name	Description
S	`static Doc createDocument()`	Creates a new, empty XML document.
S	`static Doc loadFromBinary(byte[] xml)`	Loads an XML document from a byte array.
S	`static Doc loadFromFile(String fileName)`	Loads an XML document from a file.
S	`static Doc loadFromString(String xml)`	Loads an XML document from a string.
	`byte[] saveToBinary(boolean prettyPrint)`	Saves an XML document to a byte array, with optional "pretty-print" formatting.
	`byte[] saveToBinary(boolean prettyPrint, String encoding)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding.
	`byte[] saveToBinary(boolean prettyPrint, String encoding, boolean bigEndian, boolean writeBOM)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding. Byte order and Unicode byte-order mark can be specified for Unicode encodings.
	`void saveToFile(String fileName, boolean prettyPrint)`	Saves an XML document to a file, with optional "pretty-print" formatting.

	Name	Description
	`void saveToFile(String fileName, boolean prettyPrint, boolean omitXmlDecl)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with UTF-8 encoding. When `omitXmlDecl` is true, the XML declaration will not be written.
	`void saveToFile(String fileName, boolean prettyPrint, boolean omitXmlDecl, String encoding)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding. When `omitXmlDecl` is true, the XML declaration will not be written.
	`void saveToFile(String fileName, boolean prettyPrint, boolean omitXmlDecl, String encoding, boolean bBigEndian, boolean bBOM)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding. When `omitXmlDecl` is true, the XML declaration will not be written. Byte order and Unicode byte-order mark can be specified for Unicode encodings.
	`void saveToFile(String fileName, boolean prettyPrint, String encoding)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding.
	`void saveToFile(String fileName, boolean prettyPrint, String encoding, boolean bBigEndian, boolean bBOM)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding. Byte order and Unicode byte-order mark can be specified for Unicode encodings.
	`String saveToString(boolean prettyPrint)`	Saves an XML document to a string, with optional "pretty-print" formatting.
	`String saveToString(boolean prettyPrint, boolean omitXmlDecl)`	Saves an XML document to a string, with optional "pretty-print" formatting. When `omitXmlDecl` is true, the XML declaration will not be written.
	`void setSchemaLocation(String schemaLocation)`	Adds an `xsi:schemaLocation` or `xsi:noNamespaceSchemaLocation` attribute to the root element. A root element must already exist.

25.6.8 com.[YourSchema].[YourSchemaType].MemberAttribute

When code is generated from an XML schema, a class is created for each member attribute of a type. In the descriptions below, "AttributeType" stands for the type of the member attribute itself.

Methods

	Name	Description
	`boolean exists()`	Returns true if the attribute exists.
	`int getEnumerationValue()`	Generated for enumeration types only. Returns

	Name	Description
		one of the constants generated for the possible values, or **Invalid** if the value does not match any of the enumerated values in the schema.
●	`com.altova.xml.meta.Attribute getInfo()`	Returns an object for querying schema information (see `com.altova.xml.meta.Attribute`).
●	`AttributeType getValue()`	Gets the attribute value.
●	`void remove()`	Removes the attribute from its parent element.
●	`void setEnumerationValue(int)`	Generated for enumeration types only. Pass one of the constants generated for the possible values to this method to set the value.
●	`void setValue(AttributeType value)`	Sets the attribute value.

25.6.9 com.[YourSchema].[YourSchemaType].MemberElement

When code is generated from an XML schema, a class with the following members is created for each member element of a type. In the descriptions below, "MemberType" stands for the type of the member element itself.

Methods

	Name	Description
●	`MemberType append()`	Creates a new element and appends it to its parent.
●	`MemberType at(int index)`	Returns the instance of the member element at the specified index.
●	`int count()`	Returns the count of elements.
●	`boolean exists()`	Returns true if at least one element exists.
●	`MemberType first()`	Returns the first instance of the member element.
●	`int getEnumerationValue()`	Generated for enumeration types only. Returns one of the constants generated for the possible values, or **Invalid** if the value does not match any of the enumerated values in the schema.
●	`com.altova.xml.meta.Element getInfo()`	Returns an object for querying schema information (see `com.altova.xml.meta.Element`).
●	`MemberType getValue()`	Gets the element content (only generated if element can have simple or mixed content).
●	`java.util.Iterator`	Returns an object for iterating instances of the

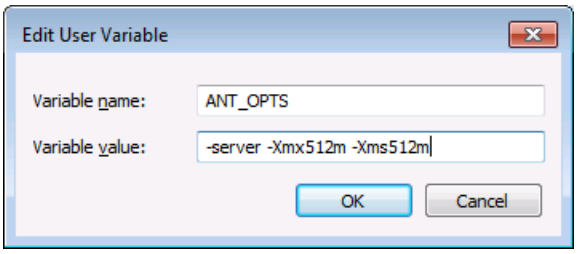
	Name	Description
	`iterator()`	member element.
●	`MemberType last()`	Returns the last instance of the member element.
●	`void remove()`	Deletes all occurrences of the element from its parent.
●	`void removeAt(int index)`	Deletes the occurrence of the element specified by the index.
●	`void setEnumerationValue(int index)`	Generated for enumeration types only. Pass one of the constants generated for the possible values to this method to set the value.
●	`void setValue(MemberType value)`	Sets the element content (only generated if element can have simple or mixed content).

25.7 Code Generation Tips

Resolving "Out of memory" exceptions during Java compilation

Complex schemas can produce a large amount of code, which might cause a java.lang.OutofMemory exception during compilation using Ant. To rectify this:

- Add the environment variable `ANT_OPTS`, which sets specific Ant options such as the memory to be allocated to the compiler, and add values as shown below.



- To make sure that the compiler and the generated code run in the same process as Ant, change the `fork` attribute, in **build.xml**, to `false`.

You may need to customize the values depending on the amount of memory in your machine and the size of the project you are working with. For more details, see your Java VM documentation.

When running the `ant jar` command, you may get an error message similar to "`[...] archive contains more than 65535 entities`". To prevent this, it is recommended that you use Ant 1.9 or later, and, in the **build.xml** file, add `zip64mode="as-needed"` to the `<jar>` element.

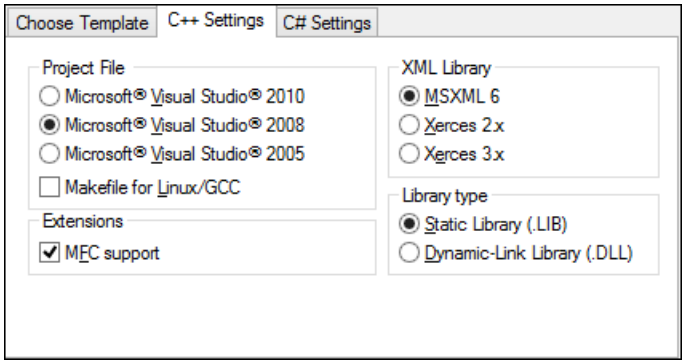
Reserving method names

When customizing code generation using the supplied SPL files, it might be necessary to reserve names to avoid collisions with other symbols. To do this:

1. Navigate to subdirectory corresponding to the programming language of the **spl** subdirectory of the program installation directory e.g. `C:\Program Files\Altova\XMLSpy2017\spl\java\`.
2. Open the **settings.spl** file and insert a new line into the reserve section e.g. reserve "myReservedWord".
3. Regenerate the program code.

25.8 Code Generator Options

The menu option **DTD/Schema | Generate Program Code** lets you specify the target programming language as well as specific code generation settings.



The available settings are as follows.

C++ Settings	Defines the specific compiler settings for the C++ environment, namely: • The Visual Studio edition (2005, 2008, 2010) • Whether a makefile for Linux with GCC compiler must be generated. • The XML library (MSXML, Xerces 2.x, Xerces 3.x) • Whether static or dynamic libraries must be generated • Whether code must be generated with or without MFC support The *Makefile for Linux/GCC* option adds makefiles to the generated code. C++ source files are generated so that they are portable using `#ifdef` constructs to support different compilers and operating systems. Note that only Xerces 3.x is supported on Linux. Checking *MFC support* has no effect on compilation with Linux/GCC.
C# Settings	Defines the specific compiler settings for the C# environment, namely, the Visual Studio edition (2005, 2008, 2010).

25.9 SPL (Spy Programming Language)

This section gives an overview of Spy Programming Language, the code generator's template language.

It is assumed that you have prior programming experience, and are familiar with operators, functions, variables and classes, as well as the basics of object-oriented programming - which is used heavily in SPL. You should also have detailed knowledge of XML Schema.

The templates used by XMLSpy are supplied in the ...\ XMLSpy\ spl folder. You can use these files as an aid to help you in developing your own templates.

How code generator works

Inputs to the code generator are the template files (.spl) and the object model provided by XMLSpy. The template files contain SPL instructions for creating files, reading information from the object model and performing calculations, interspersed with literal code fragments in the target programming language.

The template file is interpreted by the code generator and outputs **.cpp, .java, .cs** source code files, project files, or any other type of file depending on the template. The source code can then be compiled into an executable file that accesses XML data described by the schema file.

SPL files have access to a wide variety of information that is collated from the source schemas. Please note that an SPL file is not tied to a specific schema, but allows access to all schemas! Make sure you write your SPL files generically, avoid structures etc. which apply to specific schemas!

Example: Creating a new file in SPL:

```
[create "test.cpp"]
#include "stdafx.h"
[close]
```

This is a very basic SPL file. It creates a file named **test.cpp,** and places the include statement within it. The close command completes the template.

25.9.1 Basic SPL structure

An SPL file contains literal text to output, interspersed with code generator instructions.

Code generator instructions are enclosed in square brackets '**[**' and '**]**'.
Multiple statements can be included in a bracket pair. Additional statements have to be separated by a new line or a colon '**:**'.

Valid examples are:

```
[$x = 42
$x = $x + 1]
```

or

```
[$x = 42: $x = $x + 1]
```

Adding text to files

Text not enclosed by **[** and **]**, is written directly to the current output file. If there is no current output file, the text is ignored (see Using files how to create an output file).
To output literal square brackets, escape them with a backslash: **\[** and **\]**; to output a backslash

use \\.

Comments

Comments inside an instruction block always begin with a **'** character, and terminate on the next line, or at a block close character **]**.

25.9.2 Declarations

The following statements are evaluated while parsing the SPL template file. They are **not** affected by flow control statements like conditions, loops or subroutines, and are always evaluated exactly once.

These keywords, like all keywords in SPL, are not case sensitive.

Remember that all of these declarations must be inside a block delimited by square brackets.

map *mapname key* **to** *value* [, *key* **to** *value*]...

This statement adds information to a map. See below for specific uses.

map schemanativetype *schematype* **to** *typespec*

The specified built-in XML Schema type will be mapped to the specified native type or class, using the specified formatter. This setting applies only to code generation for version 2007r3 and higher. Typespec is a native type or class name, followed by a comma, followed by the formatter class instance.

Example:

```
map schemanativetype "double" to "double,Altova::DoubleFormatter"
```

map type *schematype* **to** *classname*

The specified built-in XML Schema type will be mapped to the specified class. This setting applies only to code generation for version 2007 or lower.

Example:

```
map type "float" to "CSchemaFloat"
```

default *setting* **is** *value*

This statement allows you to affect how class and member names are derived from the XML Schema.
Note that the setting names are case sensitive.

Example:

```
default "InvalidCharReplacement" is "_"
```

Setting name	Explanation
ValidFirstCharSet	Allowed characters for starting an identifier
ValidCharSet	Allowed characters for other characters in an identifier
InvalidCharReplacement	The character that will replace all characters in names that are not in the ValidCharSet

Setting name	Explanation
AnonTypePrefix	Prefix for names of anonymous types*
AnonTypeSuffix	Suffix for names of anonymous types*
ClassNamePrefix	Prefix for generated class names
ClassNameSuffix	Suffix for generated class names
EnumerationPrefix	Prefix for symbolic constants declared for enumeration values
EnumerationUpperCase	"on" to convert the enumeration constant names to upper case
FallbackName	If a name consists only of characters that are not in ValidCharSet, use this one

* Names of anonymous types are built from AnonTypePrefix + element name + AnonTypeSuffix

reserve *word*

Adds the specified word to the list of reserved words. This ensures that it will never be generated as a class or member name.

Example:
```
reserve "while"
```

include *filename*

Example:
```
include "Module.cpp"
```

includes the specified file as SPL source. This allows you to split your template into multiple files for easier editing and handling.

25.9.3 Variables

Any non-trivial SPL file will require variables. Some variables are predefined by the code generator, and new variables may be created simply by assigning values to them.

The **$** character is used when **declaring** or **using** a variable, a variable name is always prefixed by **$**.
Variable names are **case sensitive**.

Variables types:

- integer - also used as boolean, where 0 is false and everything else is true
- string
- object - provided by XMLSpy
- iterator - see foreach statement

Variable types are declared by first assignment:

```
[$x = 0]
```
x is now an integer.

```
[$x = "teststring"]
```

x is now treated as a string.

Strings

String constants are always enclosed in double quotes, like in the example above. **\n** and **\t** inside double quotes are interpreted as newline and tab, **\"** is a literal double quote, and **** is a backslash. String constants can also span multiple lines.

String concatenation uses the **&** character:

```
[$BasePath = $outputpath & "/" & $JavaPackageDir]
```

Objects

Objects represent the information contained in the XML schema. Objects have **properties**, which can be accessed using the **.** operator. It is not possible to create new objects in SPL (they are predefined by the code generator, derived from the input schema), but it is possible to assign objects to variables.

Example:

```
class [=$class.Name]
```

This example outputs the word "class", followed by a space and the value of the **Name** property of the **$class** object.

25.9.4 Predefined variables

After a Schema file is analyzed by the code generator, the objects in the table below exist in the Template Engine.

Name	Type	Description
$schematype	integer	1 for DTD, 2 for XML Schema
$TheLibrary	Library	The library derived from the XML Schema or DTD
$module	string	Name of the source Schema without extension
$outputpath	string	The output path specified by the user, or the default output path

For **C++** generation only:

Name	Type	Description
$domtype	integer	1 for MSXML, 2 for Xerces
$XercesVersion	integer	2 for Xerces 2.x, 3 for Xerces 3.x
$libtype	integer	1 for static LIB, 2 for DLL
$mfc	boolean	True if MFC support is enabled
$vc6project	boolean	True if Visual C++ project files are to be generated

Name	Type	Description
$VS2005Project	boolean	True if Visual C++ 2005 project files are to be generated
$VS2008Project	boolean	True if Visual C++ 2008 project files are to be generated
$VS2010Project	boolean	True if Visual C++ 2010 project files are to be generated

For **C#** generation only:

Name	Type	Description
$CreateVS2005Project	boolean	True if Visual Studio 2005 project files are to be generated
$CreateVS2008Project	boolean	True if Visual Studio 2008 project files are to be generated
$CreateVS2010Project	boolean	True if Visual Studio 2010 project files are to be generated

25.9.5 Creating output files

These statements are used to create output files from the code generation.

Remember that all of these statements must be inside a block delimited by square brackets.

create *filename*

creates a new file. The file has to be closed with the **close** statement. All following output is written to the specified file.

Example:

```
[create $outputpath & "/" & $JavaPackageDir & "/" & $application.Name &
".java"]
package [=$JavaPackageName];

public class [=$application.Name]Application {
...
}
[close]
```

close

closes the current output file.

=$*variable*

writes the value of the specified variable to the current output file.

Example:

```
[$x = 20+3]
The result of your calculation is [=$x] - so have a nice day!
```

- The file output will be:

```
The result of your calculation is 23 - so have a nice day!
```

write *string*

writes the string to the current output file.

Example:
```
[write "C" & $name]
```

This can also be written as:
```
C[=$name]
```

filecopy *source* **to** *target*

copies the source file to the target file, without any interpretation.

Example:
```
filecopy "java/mapforce/mapforce.png" to $outputpath & "/" & $JavaPackageDir &
"/mapforce.png"
```

25.9.6 Operators

Operators in SPL work like in most other programming languages.

List of SPL operators in descending precedence order:

.	Access object property
( )	Expression grouping
true	boolean constant "true"
false	boolean constant "false"
&	String concatenation
-	Sign for negative number
not	Logical negation
*	Multiply
/	Divide
%	Modulo
+	Add
-	Subtract
<=	Less than or equal
<	Less than
>=	Greater than or equal
>	Greater than
=	Equal
<>	Not equal
and	Logical conjunction (with short circuit evaluation)
or	Logical disjunction (with short circuit evaluation)
=	Assignment

25.9.7 Conditions

SPL allows you to use standard "if" statements. The syntax is as follows:

```
if condition
	statements
else
	statements
endif
```

or, without else:

```
if condition
	statements
endif
```

Please note that there are no round brackets enclosing the condition!
As in any other programming language, conditions are constructed with logical and comparison operators.

Example:

```
[if $namespace.ContainsPublicClasses and $namespace.Prefix <> ""]
	whatever you want ['inserts whatever you want, in the resulting file]
[endif]
```

Switch

SPL also contains a multiple choice statement.

Syntax:

```
switch $variable
	case X:
		statements
	case Y:
	case Z:
		statements
	default:
		statements
endswitch
```

The case labels must be constants or variables.

The switch statement in SPL does not fall through the cases (as in C), so there is no need for a "break" statement.

25.9.8 Collections and foreach

Collections and iterators

A collection contains multiple objects - like a ordinary array. Iterators solve the problem of storing and incrementing array indexes when accessing objects.

Syntax:

```
foreach iterator in collection
	statements
next
```

Example:

```
[foreach $class in $classes
	if not $class.IsInternal
```

```
            ]      class [=$class.Name];
[      endif
next]
```

Example 2:

```
[foreach $i in 1 To 3
      Write "// Step " & $i & "\n"
      ' Do some work
next]
```

The first line:
$classes is the **global object** of all generated types. It is a collection of single class objects.

Foreach steps through all the items in $classes, and executes the code following the instruction, up to the **next** statement, for each of them.

In each iteration, **$class** is assigned to the next class object. You simply work with the class object instead of using, classes[i]->Name(), as you would in C++.

All collection iterators have the following additional properties:

Index	The current index, starting with 0
IsFirst	true if the current object is the first of the collection (index is 0)
IsLast	true if the current object is the last of the collection
Current	The current object (this is implicit if not specified and can be left out)

Example:

```
[foreach $enum in $facet.Enumeration
      if not $enum.IsFirst
            ], [
      endif
      ]"[=$enum.Value]"[
next]
```

25.9.9 Subroutines

Code generator supports subroutines in the form of procedures or functions.

Features:

- By-value and by-reference passing of values
- Local/global parameters (local within subroutines)
- Local variables
- Recursive invocation (subroutines may call themselves)

Subroutine declaration

Subroutines

Syntax example:

```
Sub SimpleSub()
...lines of code
EndSub
```

- **Sub** is the keyword that denotes the procedure.
- **SimpleSub** is the name assigned to the subroutine.
- Round **parenthesis** can contain a parameter list.
- The code block of a subroutine starts immediately after the closing parameter parenthesis.
- **EndSub** denotes the end of the code block.

Please note:
Recursive or cascaded subroutine **declaration** is not permitted, i.e. a subroutine may not contain another subroutine.

Parameters

Parameters can also be passed by procedures using the following syntax:

- All parameters must be variables
- Variables must be prefixed by the **$** character
- Local variables are defined in a subroutine
- Global variables are declared explicitly, outside of subroutines
- Multiple parameters are separated by the comma character "**,**" within round parentheses
- Parameters can pass values

Parameters - passing values

Parameters can be passed in two ways, by value and by reference, using the keywords **ByVal** and **ByRef** respectively.

Syntax:

```
'define sub CompleteSub()
[Sub CompleteSub($param,ByVal$paramByValue,ByRef$paramByRef)
] …
```

- **ByVal** specifies that the parameter is passed by value. Note that most objects can only be passed by reference.
- **ByRef** specifies that the parameter is passed by reference. This is the default if neither ByVal nor ByRef is specified.

Function return values

To return a value from a subroutine, use the **return** statement. Such a function can be called from within an expression.

Example:

```
'define a function
[Sub MakeQualifiedName(ByVal$namespacePrefix,ByVal$localName)
if $namespacePrefix = ""
  return $localName
else
  return $namespacePrefix & ":" & $localName
endif
EndSub
]
```

Subroutine invocation

Use **call** to invoke a subroutine, followed by the procedure name and parameters, if any.

```
    CallSimpleSub()
```

or,

```
CallCompleteSub("FirstParameter",$ParamByValue,$ParamByRef)
```

Function invocation

To invoke a function (any subroutine that contains a **return** statement), simply use its name inside an expression. Do not use the **call** statement to call functions.
Example:

```
$QName = MakeQualifiedName($namespace,"entry")
```

Subroutine example

Highlighted example showing subroutine declaration and invocation.

A sample SPL file:

```
[create $outputpath & $module & "output.txt"

' define sub SimpleSub()
sub SimpleSub()
]SimpleSub() called
[endsub

' execute sub SimpleSub()
call SimpleSub()

$ParamByValue    = "Original Value"
]ParamByValue    = [=$ParamByValue]
[$ParamByRef     = "Original Value"
]ParamByRef      = [=$ParamByRef]

' define sub CompleteSub()
[sub CompleteSub( $param, byval $paramByValue, byref $paramByRef )
]CompleteSub called.
    param = [=$param]
    paramByValue = [=$paramByValue]
    paramByRef = [=$paramByRef]
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]   Set values inside sub
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]CompleteSub finished.
[endsub

' run sub CompleteSub()
call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
]
ParamByValue=[=$ParamByValue]
ParamByRef=[=$ParamByRef]
[
close
]
```

The same sample code:

```
[create $outputpath & $module & "output.txt"

'define sub SimpleSub()
Sub SimpleSub()
]SimpleSub() called
[endsub

'execute sub SimpleSub()
CallSimpleSub()

$ParamByValue      = "OriginalValue"
]ParamByValue      = [=$ParamByValue]
[$ParamByRef= "OriginalValue"
]ParamByRef  = [=$ParamByRef]
```

```
'define sub CompleteSub()
[Sub CompleteSub($param,ByVal$paramByValue,ByRef$paramByRef)
]CompleteSub called.
        param = [=$param]
        paramByValue = [=$paramByValue]
        paramByRef= [=$paramByRef]
[$ParamByRef= "LocalVariable"
$paramByValue = "new value"
$paramByRef= "new value"
]       Set values inside Sub
[$ParamByRef= "LocalVariable"
$paramByValue = "new value"
$paramByRef= "new value"
]CompleteSub finished.
[endsub

'run sub CompleteSub()
CallCompleteSub("FirstParameter",$ParamByValue,$ParamByRef)
]
ParamByValue=[=$ParamByValue]
ParamByRef=[=$ParamByRef]
[
Close
]
```

25.9.10 Built in Types

The section describes the properties of the built-in types used in the predefined variables which describe the parsed schema.

Library

This object represents the whole library generated from the XML Schema or DTD.

Property	Type	Description
SchemaNamespaces	Namespace collection	Namespaces in this library
SchemaFilename	string	Name of the XSD or DTD file this library is derived from
SchemaType	integer	1 for DTD, 2 for XML Schema
Guid	string	A globally unique ID
CodeName	string	Generated library name (derived from schema file name)

Namespace

One namespace object per XML Schema namespace is generated. Schema components that are not in any namespace are contained in a special namespace object with an empty NamespaceURI.
Note that for DTD, namespaces are also derived from attributes whose names begin with "xmlns".

Property	Type	Description
CodeName	string	Name for generated code (derived from prefix)
LocalName	string	Namespace prefix

Property	Type	Description
NamespaceURI	string	Namespace URI
Types	Type collection	All types contained in this namespace
Library	Library	Library containing this namespace

Type

This object represents a complex or simple type. It is used to generate a class in the target language.
There is one additional type per library that represents the document, which has all possible root elements as members.
Anonymous types have an empty LocalName.

Property	Type	Description
CodeName	string	Name for generated code (derived from local name or parent declaration)
LocalName	string	Original name in the schema
Namespace	Namespace	Namespace containing this type
Attributes	Member collection	Attributes contained in this type*
Elements	Member collection	Child elements contained in this type
IsSimpleType	boolean	True for simple types, false for complex types
IsDerived	boolean	True if this type is derived from another type, which is also represented by a Type object
IsDerivedByExtension	boolean	True if this type is derived by extension
IsDerivedByRestriction	boolean	True if this type is derived by restriction
IsDerivedByUnion	boolean	True if this type is derived by union
IsDerivedByList	boolean	True if this type is derived by list
BaseType	Type	The base type of this type (if IsDerived is true)
IsDocumentRootType	boolean	True if this type represents the document itself
Library	Library	Library containing this type
IsFinal	boolean	True if declared as final in the schema
IsMixed	boolean	True if this type can have mixed content
IsAbstract	boolean	True if this type is declared as abstract

Property	Type	Description
IsGlobal	boolean	True if this type is declared globally in the schema
IsAnonymous	boolean	True if this type is declared locally in an element

For simple types only:

Property	Type	Description
IsNativeBound	boolean	True if native type binding exists
NativeBinding	NativeBinding	Native binding for this type
Facets	Facets	Facets of this type
Whitespace	string	Shortcut to the Whitespace facet

* Complex types with text content (these are types with mixed content and complexType with simpleContent) have an additional unnamed attribute member that represents the text content.

Member

This object represents an attribute or element in the XML Schema. It is used to create class members of types.

Property	Type	Description
CodeName	string	Name for generated code (derived from local name or parent declaration)
LocalName	string	Original name in the schema. Empty for the special member representing text content of complex types.
NamespaceURI	string	The namespace URI of this Element/ Attribute within XML instance documents/ streams.
DeclaringType	Type	Type originally declaring the member (equal to ContainingType for non-inherited members)
ContainingType	Type	Type where this is a member of
DataType	Type	Data type of this member's content
Library	Library	Library containing this member's DataType
IsAttribute	boolean	True for attributes, false for elements
IsOptional	boolean	True if minOccurs = 0 or optional attribute
IsRequired	boolean	True if minOccurs > 0 or required attribute

Property	Type	Description
IsFixed	boolean	True for fixed attributes, value is in Default property
IsDefault	boolean	True for attributes with default value, value is in Default property
IsNillable	boolean	True for nillable elements
IsUseQualified	boolean	True if NamespaceURI is not empty
MinOccurs	integer	minOccurs, as in schema. 1 for required attributes
MaxOccurs	integer	maxOccurs, as in schema. 0 for prohibited attributes, - 1 for unbounded
Default	string	Default value

NativeBinding

This object represents the binding of a simple type to a native type in the target programming language, as specified by the "schemanativetype" map.

Property	Type	Description
ValueType	string	Native type
ValueHandler	string	Formatter class instance

Facets

This object represents all facets of a simple type.
Inherited facets are merged with the explicitly declared facets. If a Length facet is in effect, MinLength and MaxLength are set to the same value.

Property	Type	Description
DeclaringType	Type	Type facets are declared on
Whitespace	string	"preserve", "collapse" or "replace"
MinLength	integer	Facet value
MaxLength	integer	Facet value
MinInclusive	integer	Facet value
MinExclusive	integer	Facet value
MaxInclusive	integer	Facet value
MaxExclusive	integer	Facet value
TotalDigits	integer	Facet value
FractionDigits	integer	Facet value
List	Facet collection	All facets as list

Facet

This object represents a single facet with its computed value effective for a specific type.

Property	Type	Description
LocalName	string	Facet name
NamespaceURI	string	Facet namespace
FacetType	string	one of "normalization", "lexicalspace", "valuespace-length", "valuespace-enum" or "valuespace-range"
DeclaringType	Type	Type this facet is declared on
FacetCheckerName	string	Name of facet checker (from schemafacet map)
FacetValue	string or integer	Actual value of this facet

25.10 Error Codes

Operating System Error Codes

201 File not found: '%s'
202 Cannot create file '%s'
203 Cannot open file '%s'
204 Cannot copy file '%s' to '%s'

Syntax Error Codes

401 Keyword expected
402 '%s' expected
403 No output file specified
404 Unexpected end of file
405 Keyword not allowed

Runtime Error Codes

501 Unknown variable '%s'
502 Redefinition of variable '%s'
503 Variable '%s' is not a container
504 Unknown property '%s'
505 Cannot convert from %s to %s
507 Unknown function
508 Function already defined
509 Invalid parameter
510 Division by zero
511 Unknown method
512 Incorrect number of parameters
513 Stack overflow

26 Menu Commands

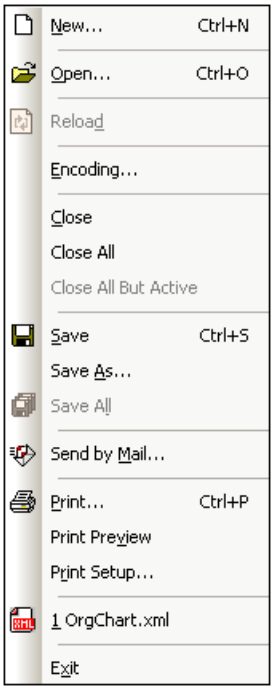
The **User Reference** section contains a complete description of all XMLSpy menu commands and explains their use. We have tried to be comprehensive. If, however, you have questions which are not covered in the User Reference or other parts of this documentation, please look up the FAQs and Discussion Forums on the Altova website. If you cannot find a suitable answer at these locations, please do not hesitate to contact the Altova Support Center.

Standard Windows commands, such as (**Open**, **Save**, **Cut**, **Copy** and **Paste**) are in the File and Edit menus. These menus additionally contain XML- and Internet-related commands.

(C) 2017 Altova GmbH

26.1 File Menu

The **File** menu contains commands for file operations, ordered as in most Windows applications. In addition to the standard New, Open, Save, Print, Print Setup, and Exit commands, XMLSpy also offers XML-specific and application-specific commands.



26.1.1 New

This section:

- Icon and shortcut
- Description
- Templates for new documents
- Assigning a DTD or XML Schema to a new XML document
- Specifying the root element of a new XML document
- Assigning an SPS to a new XML document
- Creaing new XBRL taxonomies with the XBRL Taxonomy Wizard

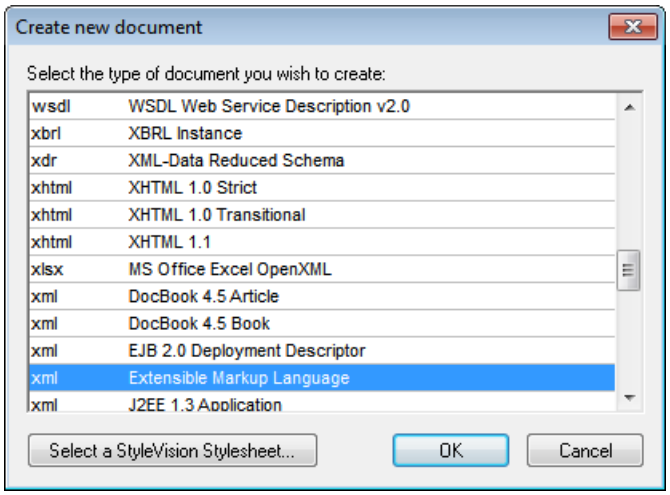
Icon and shortcut

Icon:	
Shortcut:	**Ctrl+N**

Description

The **New** command is used to create a new document. Clicking **New** opens the Create New

Document dialog (*screenshot below*), in which you can select the type of document you wish to create. If the document type you wish to create is not listed in the dialog, select XML and change the file extension when you save the file. Note that you can add new document types to this dialog list using the Tools | Options | File types tab.

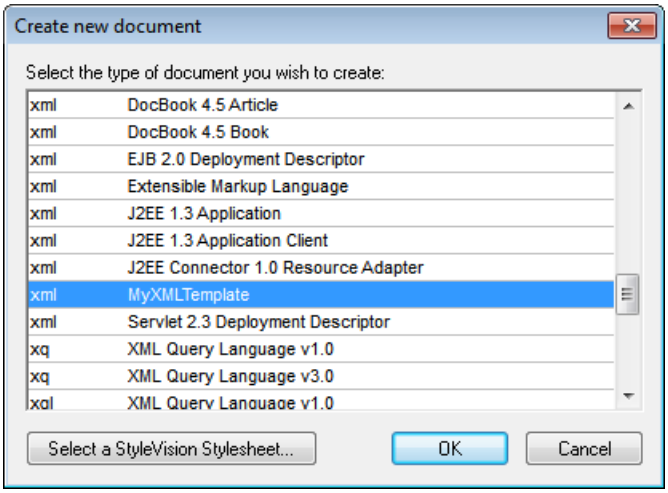


Templates for new documents

The document-type list of the Create New Document dialog can also contain entries for user-defined document templates of any document type. These templates can be opened directly from the Create New Document dialog and edited. To create your own document template so that it appears in the list of document types in the Create New Document dialog, you first create the template document and then save it to the folder designated to contain document templates.

Create a document template as follows:

1. Open the `XMLSpy\Template` folder of the application folder using Windows Explorer or your preferred navigation tool, and select a rudimentary template file from among the files named **`new xxx`** (where `.xxx` is a file extension, such as `.xml` or `.xslt`).
2. Open the file in XMLSpy, and modify the file as required. This file will be the template file.
3. After you have finished, select **File | Save as** to save the file back to the `\Template` folder with a suitable name, say `MyXMLTemplate.xml`. You now have a document template called `MyXMLTemplate`, which will appear in the list of document types in the Create New Document dialog.

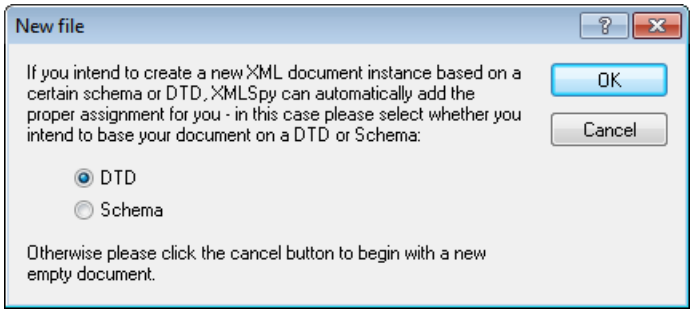


4. To open the template, select **File | New**, and then the template (`my-xml` in this case).

To delete a document template from the list of document types, delete (or move) the template file from the template folder.

Assigning a DTD or XML Schema to a new XML document
When you create a new document of a certain type via the Create New Document dialog, the document is automatically opened with the correct DTD or XML Schema association. For example, an XHTML file will be opened with the DTD `http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd` associated with it. And an XML Schema (`.xsd`) file is associated with the `http://www.w3.org/2001/XMLSchema` schema document.

If you are creating a new document for which the schema is not known (for example, an XML file), then you are prompted to associate a schema (DTD or XML Schema) to the new document (*screenshot below*).

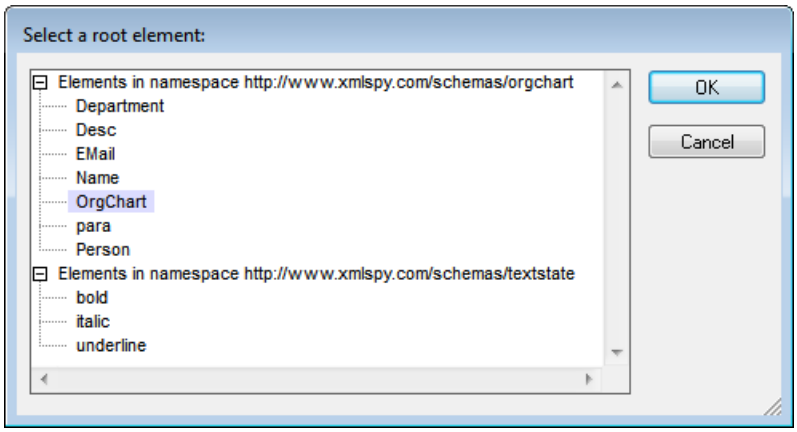


If, in the dialog, you select DTD or XML Schema and click **OK**, you can browse for the schema you want. Clicking **Cancel** creates a new file that is not associated with any schema.

Specifying the root element of a new XML document

If an XML Schema is selected as the associated schema of an XML document and if this schema has more than one global element, each of these is a potential root element. In this case, the Select a Root Element dialog (*screenshot below*) pops up, in which you can select which global

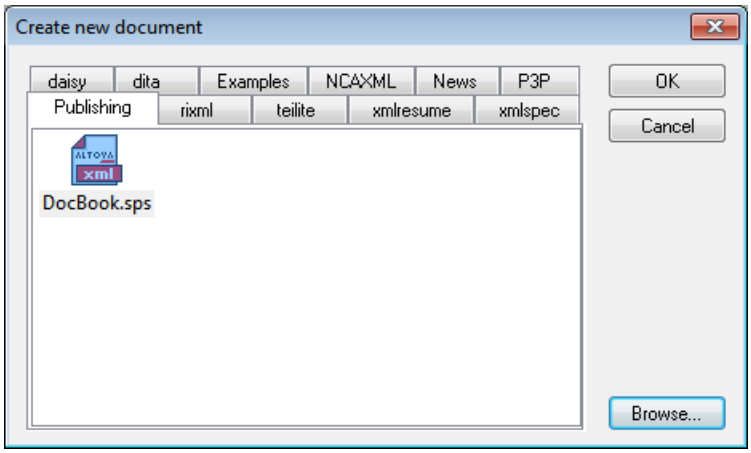
element is to be the root element of the XML document. In the screenshot below, the `OrgChart` global element is selected.



Clicking **OK** now will create a new XML document with this element (`OrgChart`) as its root element.

Assigning an SPS to a new XML document

When a new XML document is created, you can associate a StyleVision Power Stylesheet (`.sps` file) to view the document in Authentic View. In the Create New Document dialog (*see first screenshot in this section*), when you click the **Select StyleVision Stylesheet**, the Create New Document dialog (*shown below*) appears.



You can browse for the required SPS in the folder tabs, or you can click the **Browse** button to navigate for and select the SPS.

Creating new XBRL taxonomies with the XBRL Taxonomy Wizard

In the Create a New Document dialog, if XBRL Taxonomy Schema (`.xsd`) is selected, then a wizard guides you through the steps for creating a new XBRL taxonomy. Ths wizard is described in the XBRL section of the documentation.

26.1.2 Open

Icon and shortcut

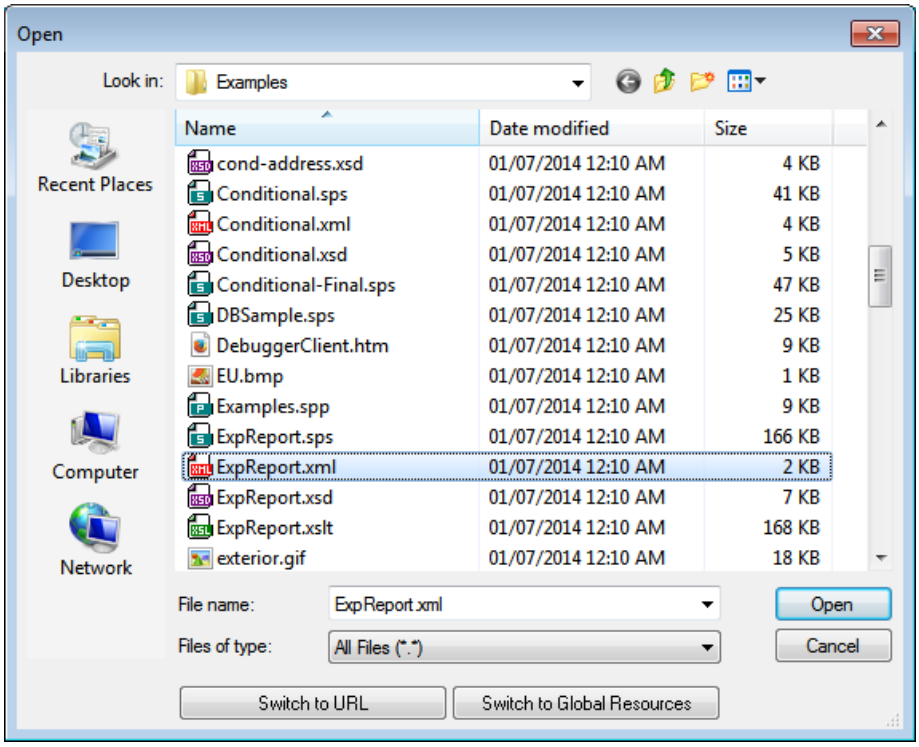
Icon:	
Shortcut:	**Ctrl+O**

Description

The **Open** command pops up the familiar Windows Open dialog, and allows you to open any XML-related document or text document. In the Open dialog, you can select more than one file to open. Use the Files of Type combo box to restrict the kind of files displayed in the dialog box. (The list of available file types can be configured in the File Types tab of the Options dialog (**Tools | Options**).) When an XML file is opened, it is checked for well-formedness. If the file is not well-formed, you will get a file-not-well-formed error. Fix the error and select the menu command **XML | Check Well-Formedness (F7)** to recheck. If you have opted for automatic validation upon opening and the file is invalid, you will get an error message. Fix the error and select the menu command **XML | Validate XML (F8)** to revalidate.

Selecting and saving files via URLs and Global Resources

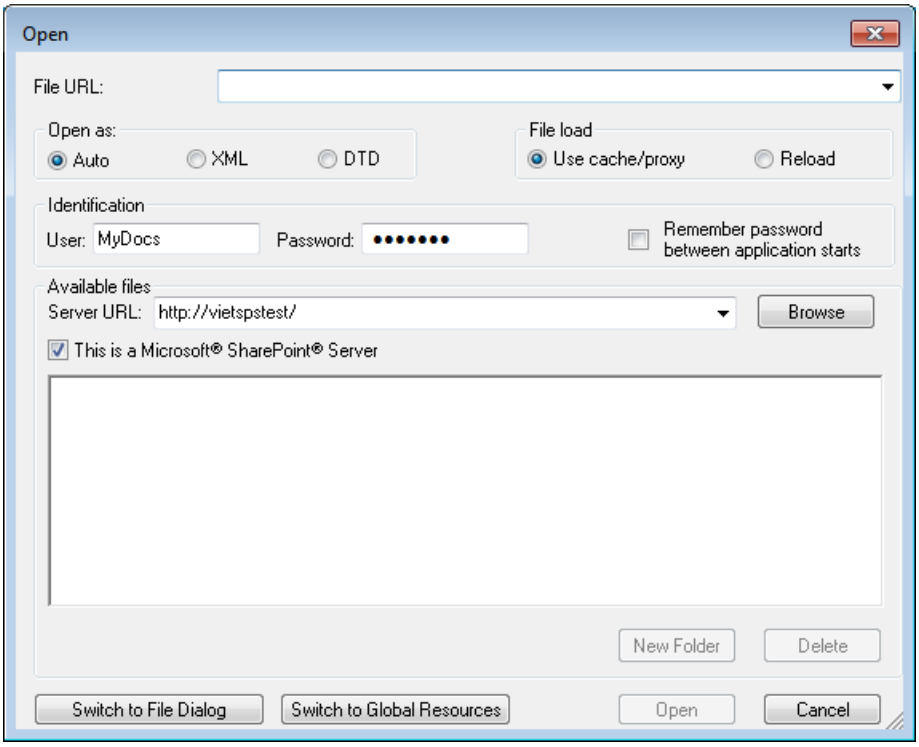
In several File Open and File Save dialogs, you can choose to select the required file or save a file via a URL or a global resource (*see screenshot below*). Click **Switch to URL** or **Switch to Global Resource** to go to one of these selection processes.



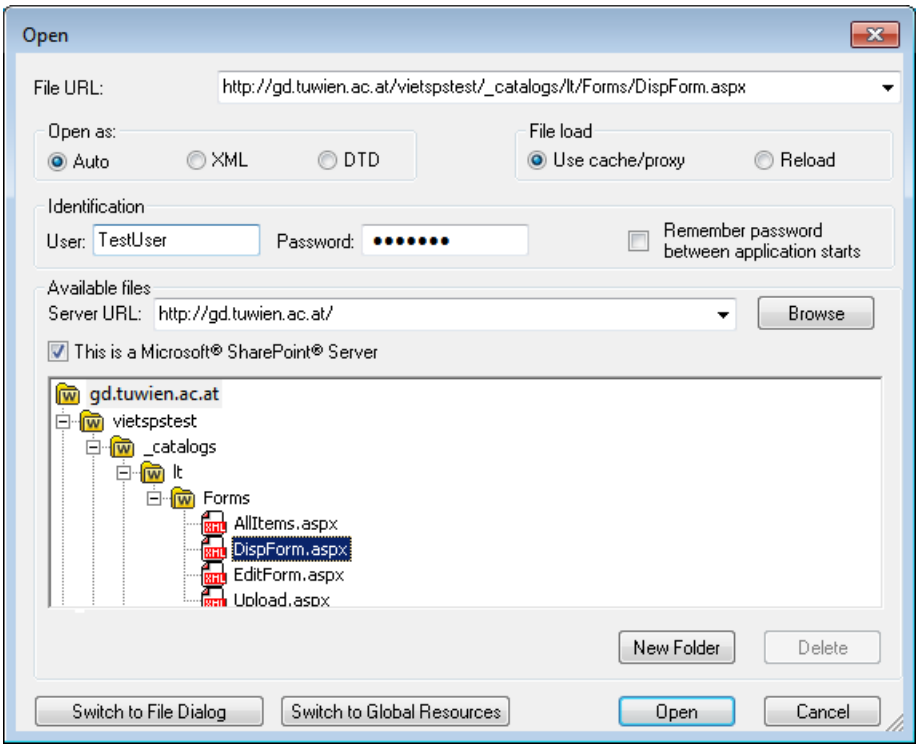
Selecting files via URLs

To select a file via a URL (either for opening or saving), do the following:

1. Click the **Switch to URL** command. This switches to the URL mode of the Open or Save dialog (*the screenshot below shows the Open dialog*).



2. Enter the URL you want to access in the *Server URL* field (*screenshot above*). If the server is a Microsoft® SharePoint® Server, check the *Microsoft® SharePoint® Server* check box. See the Microsoft® SharePoint® Server Notes below for further information about working with files on this type of server.
3. If the server is password protected, enter your User-ID and password in the *User* and *Password* fields.
4. Click **Browse** to view and navigate the directory structure of the server.
5. In the folder tree, browse for the file you want to load and click it.



The file URL appears in the File URL field (*see screenshot above*). The **Open** or **Save** button only becomes active at this point.

6. Click **Open** to load the file or **Save** to save it.

Note the following:

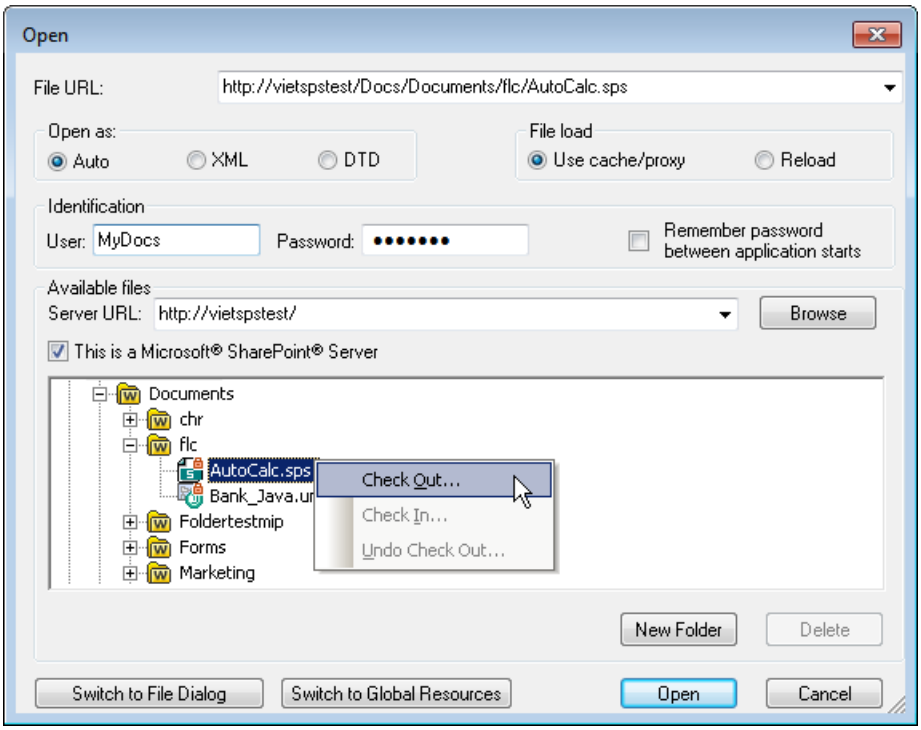
- The Browse function is only available on servers which support WebDAV and on Microsoft SharePoint Servers. The supported protocols are FTP, HTTP, and HTTPS.
- To give you more control over the loading process when opening a file, you can choose to load the file through the local cache or a proxy server (which considerably speeds up the process if the file has been loaded before). Alternatively, you may want to reload the file if you are working, say, with an electronic publishing or database system; select the **Reload** option in this case

.

Microsoft® SharePoint® Server Notes

Note the following points about files on Microsoft® SharePoint® Servers:

- In the directory structure that appears in the Available Files pane (*screenshot below*), file icons have symbols that indicate the check-in/check-out status of files.



Right-clicking a file pops up a context menu containing commands available for that file (*screenshot above*).

- The various file icons are shown below:

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

- After you check out a file, you can edit it in your Altova application and save it using **File | Save (Ctrl+S)**.
- You can check-in the edited file via the context menu in the Open URL dialog (*see screenshot above*), or via the context menu that pops up when you right-click the file tab in the Main Window of your application (*screenshot below*).

- When a file is checked out by another user, it is not available for check out.
- When a file is checked out locally by you, you can undo the check-out with the Undo Check-Out command in the context menu. This has the effect of returning the file unchanged to the server.
- If you check out a file in one Altova application, you cannot check it out in another Altova application. The file is considered to be already checked out to you. The available commands at this point in any Altova application supporting Microsoft® SharePoint® Server will be: **Check In** and **Undo Check Out**.

Opening and saving files via Global Resources

To open or save a file via a global resources, click **Switch to Global Resource**. This pops up a dialog in which you can select the global resource. These dialogs are described in the section, Using Global Resources. For a general description of Global Resources, see the Global Resources section in this documentation.

26.1.3 Reload

Icon

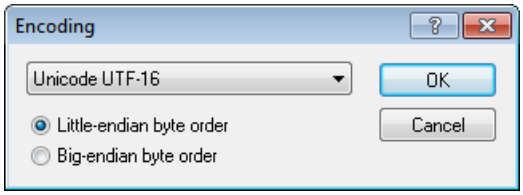
Icon:	

Description

Reloads any open documents that have modified outside XMLSpy. If one or more documents is modified outside XMLSpy, a prompt appears asking whether you wish to reload the modified document/s. If you choose to reload, then any changes you may have made to the file since the last time it was saved will be lost.

26.1.4 Encoding

The **Encoding** command lets you: (i) view the current encoding of the active document (XML or non-XML), and (ii) select a different encoding with which the active document will be saved the next time.



In XML documents, if you select a different encoding than the one currently in use, the encoding attribute in the XML declaration will be modified accordingly. For two-byte and four-byte character encodings (UTF-16, UCS-2, and UCS-4) you can also specify the byte-order to be used for the file. Another way to change the encoding of an XML document is to directly edit the encoding attribute of the document's XML declaration. Default encodings for existing and new XML and non-XML documents can be set in the Encoding tab of the Options dialog.

Note: When saving a document, XMLSpy automatically checks the encoding specification and enables you to select the required encoding via the Encoding dialog. If your document contains characters that cannot be represented in the selected encoding and you attempt to save the file, you will get a warning message to this effect.

26.1.5 Close, Close All, Close All But Active

Close

The **Close** command closes the active document window. If the file was modified (indicated by an asterisk * after the file name in the title bar), you will be asked if you wish to save the file first.

Close All

The **Close All** command closes all open document windows. If any document has been modified (indicated by an asterisk * after the file name in the title bar), you will be asked if you wish to save the file first.

Close All But Active

The **Close All But Active** command closes all open document windows except the active document window. If any document has been modified (indicated by an asterisk * after the file name in the title bar), you will be asked if you wish to save the file first.

26.1.6 Save, Save As, Save All

Icons and shortcuts

Command	*Icon*	*Shortcut*
Save		**Ctrl+S**
Save All		

Save

The **Save** command **(Ctrl+S)** saves the contents of the active document to the file from which it has been opened. When saving a document, the file is automatically checked for well-formedness. The file will also be validated automatically if this option has been set in the File tab of the Options dialog (**Tools | Options**). The XML declaration is also checked for the encoding specification, and this encoding is applied to the document when the file is saved.

Save As

The **Save As** command pops up the familiar Windows Save As dialog box, in which you enter the name and location of the file you wish to save the active file as. The same checks and validations occur as for the **Save** command.

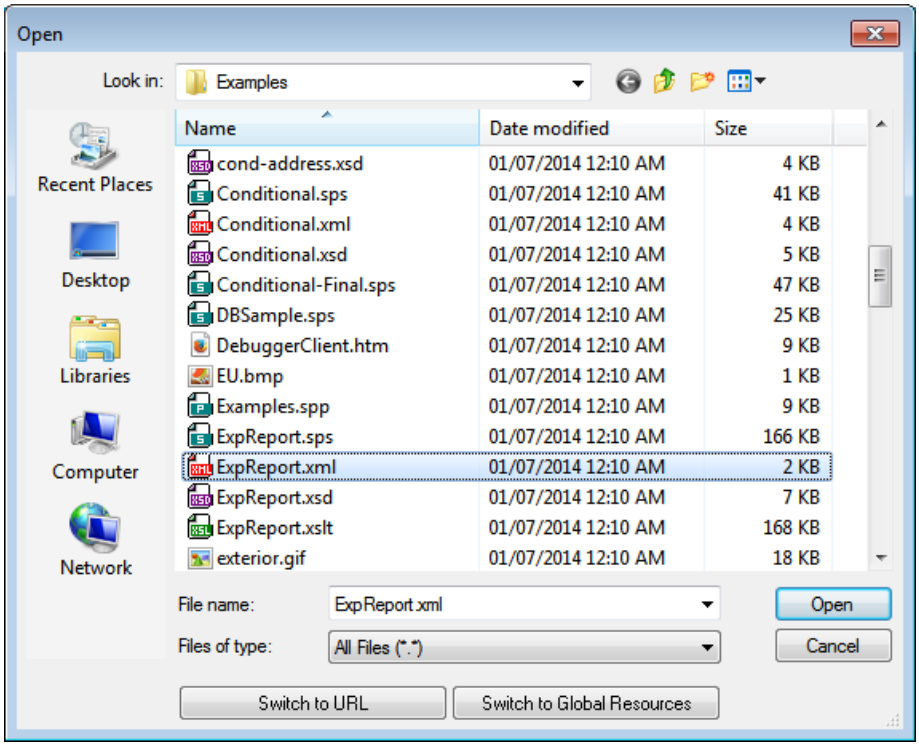
Save All

The **Save All** command saves all modifications that have been made to any open documents. The command is useful if you edit multiple documents simultaneously. If a document has not been saved before (for example, after being newly created), the Save As dialog box is presented for

that document.

Selecting and saving files via URLs and Global Resources

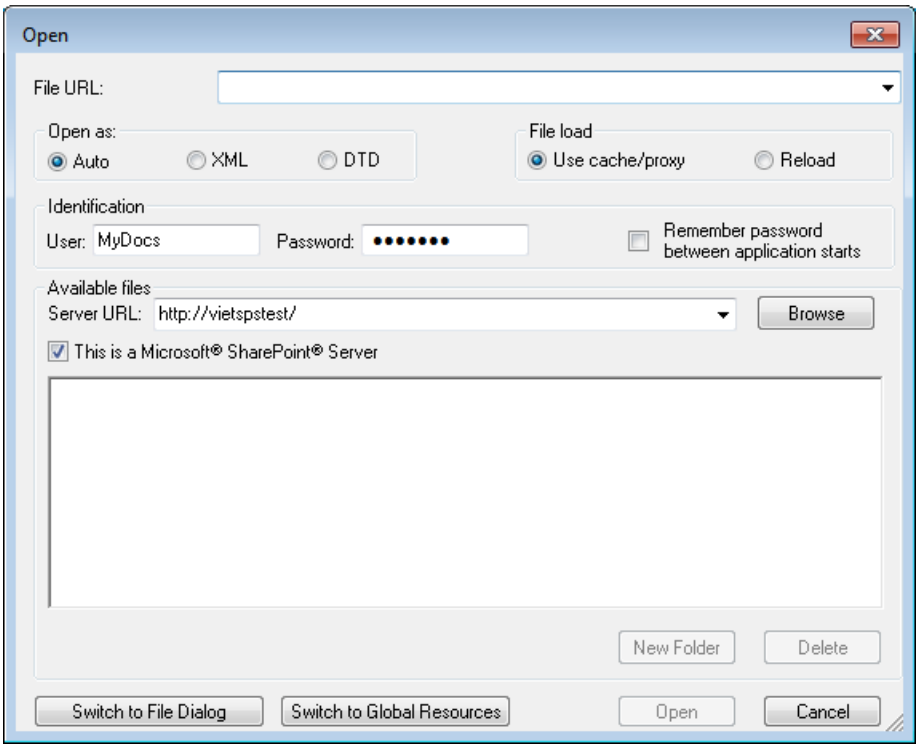
In several File Open and File Save dialogs, you can choose to select the required file or save a file via a URL or a global resource (*see screenshot below*). Click **Switch to URL** or **Switch to Global Resource** to go to one of these selection processes.



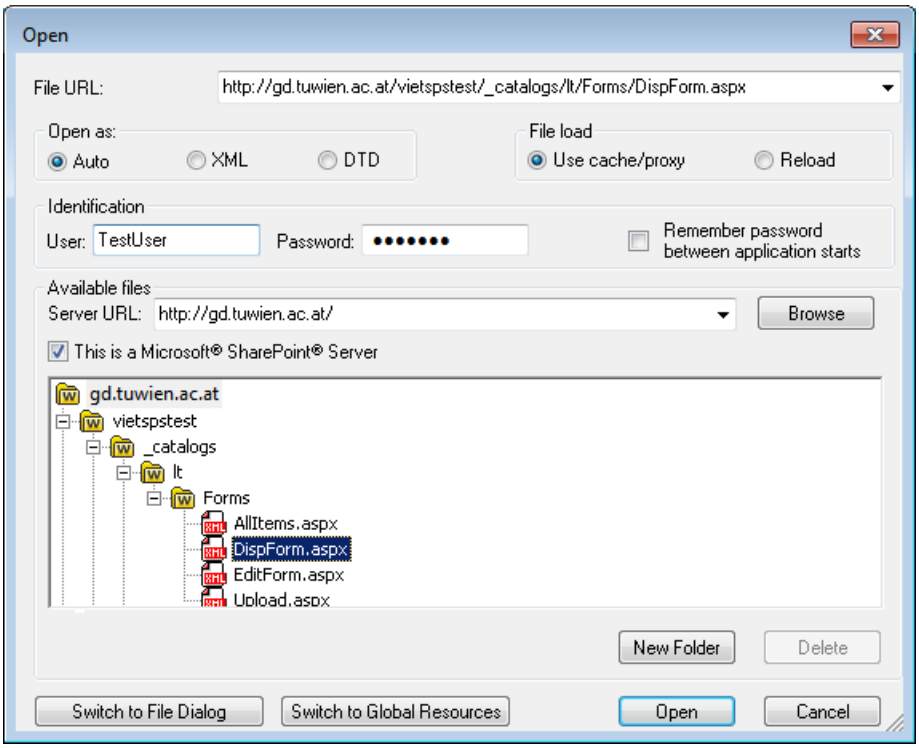
Selecting files via URLs

To select a file via a URL (either for opening or saving), do the following:

1. Click the **Switch to URL** command. This switches to the URL mode of the Open or Save dialog (*the screenshot below shows the Open dialog*).



2. Enter the URL you want to access in the *Server URL* field (*screenshot above*). If the server is a Microsoft® SharePoint® Server, check the *Microsoft® SharePoint® Server* check box. See the Microsoft® SharePoint® Server Notes below for further information about working with files on this type of server.
3. If the server is password protected, enter your User-ID and password in the *User* and *Password* fields.
4. Click **Browse** to view and navigate the directory structure of the server.
5. In the folder tree, browse for the file you want to load and click it.



The file URL appears in the File URL field (*see screenshot above*). The **Open** or **Save** button only becomes active at this point.

6. Click **Open** to load the file or **Save** to save it.

Note the following:

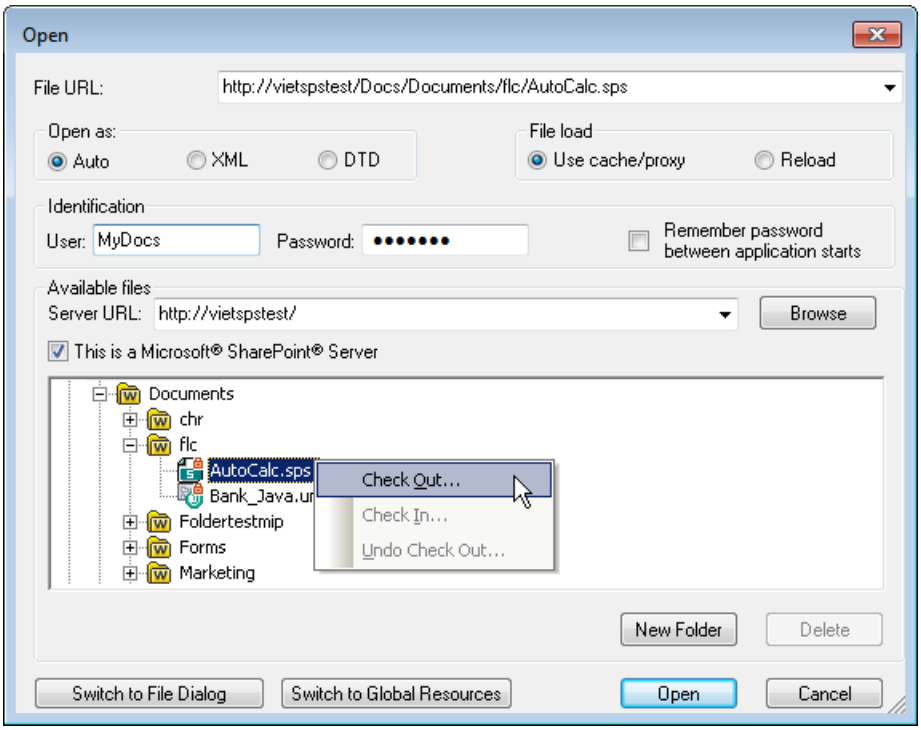
- The Browse function is only available on servers which support WebDAV and on Microsoft SharePoint Servers. The supported protocols are FTP, HTTP, and HTTPS.
- To give you more control over the loading process when opening a file, you can choose to load the file through the local cache or a proxy server (which considerably speeds up the process if the file has been loaded before). Alternatively, you may want to reload the file if you are working, say, with an electronic publishing or database system; select the **Reload** option in this case

.

Microsoft® SharePoint® Server Notes

Note the following points about files on Microsoft® SharePoint® Servers:

- In the directory structure that appears in the Available Files pane (*screenshot below*), file icons have symbols that indicate the check-in/check-out status of files.



Right-clicking a file pops up a context menu containing commands available for that file (*screenshot above*).

- The various file icons are shown below:

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

- After you check out a file, you can edit it in your Altova application and save it using **File | Save (Ctrl+S)**.
- You can check-in the edited file via the context menu in the Open URL dialog (*see screenshot above*), or via the context menu that pops up when you right-click the file tab in the Main Window of your application (*screenshot below*).

- When a file is checked out by another user, it is not available for check out.
- When a file is checked out locally by you, you can undo the check-out with the Undo Check-Out command in the context menu. This has the effect of returning the file unchanged to the server.
- If you check out a file in one Altova application, you cannot check it out in another Altova application. The file is considered to be already checked out to you. The available commands at this point in any Altova application supporting Microsoft® SharePoint® Server will be: **Check In** and **Undo Check Out**.

Opening and saving files via Global Resources

To open or save a file via a global resources, click **Switch to Global Resource**. This pops up a dialog in which you can select the global resource. These dialogs are described in the section, Using Global Resources. For a general description of Global Resources, see the Global Resources section in this documentation.

26.1.7 Send by Mail

Icon

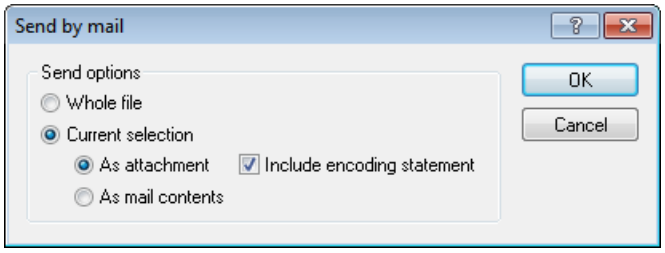
Icon:	

Description

The **Send by Mail** command lets you send XML document/s or selections from an XML document by e-mail. Depending on what kind it is, a document or selection can be sent as an attachment, content, or as a link. See the table below for details.

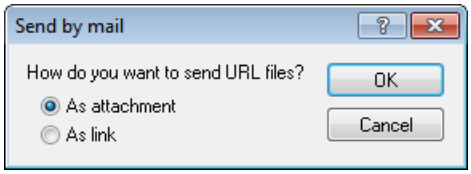
What can be sent	How it can be sent
Active XML document	As e-mail attachment
Selection in active XML document	As e-mail attachment or e-mail content
One or more files in Project window	As e-mail attachment
One or more URLs in Project window	As e-mail attachment or link

When the **Send by Mail** command is invoked on a selection in the active XML document, the Send by Mail dialog (*screenshot below*) pops up and offers the sending options shown in the screenshot. If the **Send by Mail** command is invoked with no text selected in the active file, then the *Whole File* radio button (*refer to screenshot above*) is the only option that is enabled; the other options are disabled.



Since files sent from the Project window are always sent as e-mail attachments only, the Send by Email dialog is skipped and an e-mail is opened that has the selected file/s as attachments. URLs in the project window can be sent as an attachment or as a link (*see screenshot below*). Select

how the URL is to be sent and click **OK**.



26.1.8 Print

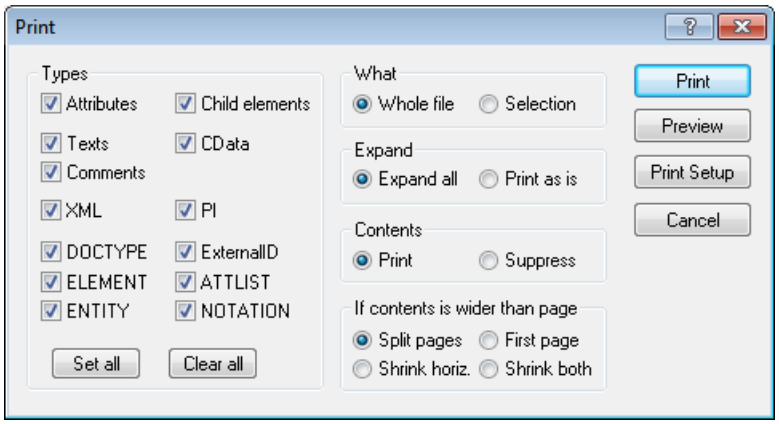
Icon and shortcut

Icon:	
Shortcut:	**Ctrl+P**

Description

The **Print** command opens the Print dialog box, in which you can select printing options for printing the currently active document.

Clicking the **Print** command in Grid View opens a Print options dialog (*screenshot below*), which enables you to set printing options for the XML document. Clicking **Print** in this dialog takes you to the Print dialog for printer options.



The available options for Grid View printing are described below:

- In the Types pane, you select the items you wish to have appear in the output.
- For the What option, you specify whether the current selection or the entire file is to be printed.
- The Expand option allows you to print the document as is, or with all descendant elements expanded fully.
- The Contents option enables you to choose between printing contents of all nodes or printing node names only.
- In the If Contents Are Wider Than Page pane, you select what to do if contents are wider than the page. The Split Pages option prints the entire document at normal size, splitting contents over pages both horizontally and vertically. The pages could then be glued together to form a poster. The First Page option prints only the first, left-hand page of the print area. The area that overflows horizontally is not printed. This option is useful if most

of the important information in your Grid View of the document is contained on the left side. The Shrink Horizontally option reduces the size of the output (proportionally) until it fits horizontally on the page; the document may run on for several pages. The Shrink Both option shrinks the document in both directions until it fits exactly on one sheet.

- The Print button prints the document with the selected options.
- The Preview button opens a print preview window that lets you view the final output before committing it to paper.
- The Print Setup button opens the Print Setup dialog box and allows you to adjust the paper format, orientation, and other printer options for this print job only. Also see the Print Setup command.

Note: You can change column widths in Grid View to optimize the print output.

Program logo
If you have a purchased license, you can turn off the program logo, copyright notice, and registration details when printing a document from XMLSpy. This option is available in the View tab of the Options dialog.

SDL and XBRL designs

Graphical views of WSDL and XBRL documents, as seen on screen, can also be printed using the **Print** command.

26.1.9 Print Preview, Print Setup

Print Preview

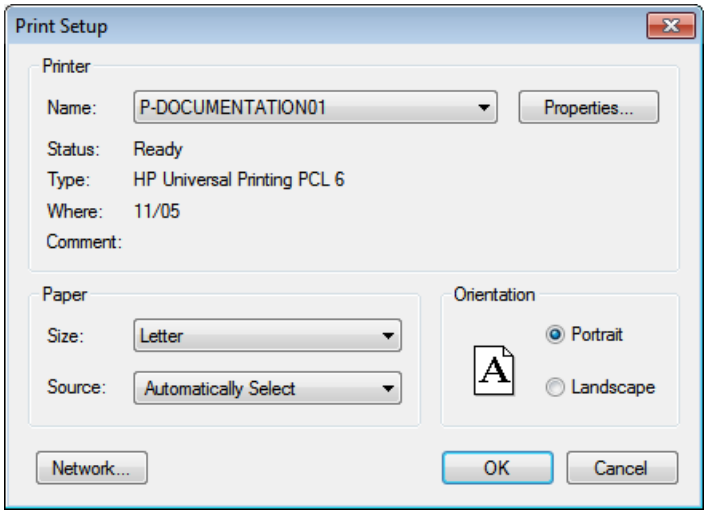
The **Print Preview** command clicked in Text View, Authentic View, and Browser View opens a print preview of the currently active document. From Grid View, Schema View, WSDL View, and XBRL View, it opens the Print dialog box, in which you can select print options and then click the **Preview** button to get the print preview.

In Print Preview mode, the Print Preview toolbar at top left of the preview window provides print- and preview-related options. The preview can be magnified or miniaturized using the the **Zoom In** and **Zoom Out** buttons. When the page magnification is such that an entire page length fits in a preview window, then the **One Page / Two Page** button toggles the preview to one or two pages at a time. The **Next Page** and **Previous Page** buttons can be used to navigate among the pages. The toolbar also contains buttons to print all pages and to close the preview window.

Note: To enable background colors and images in Print Preview, do the following: (i) In the **Tools** menu of Internet Explorer, click **Internet Options**, and then click the Advanced tab; (ii) In the Settings box, under Printing, select the *Print background colors and images* check box, and (iii) Then click **OK**.

Print Setup

The **Print Setup** command, displays the printer-specific Print Setup dialog box, in which you specify such printer settings as paper format and page orientation. These settings are applied to all subsequent print jobs.



26.1.10 Recent Files, Exit

Recent Files

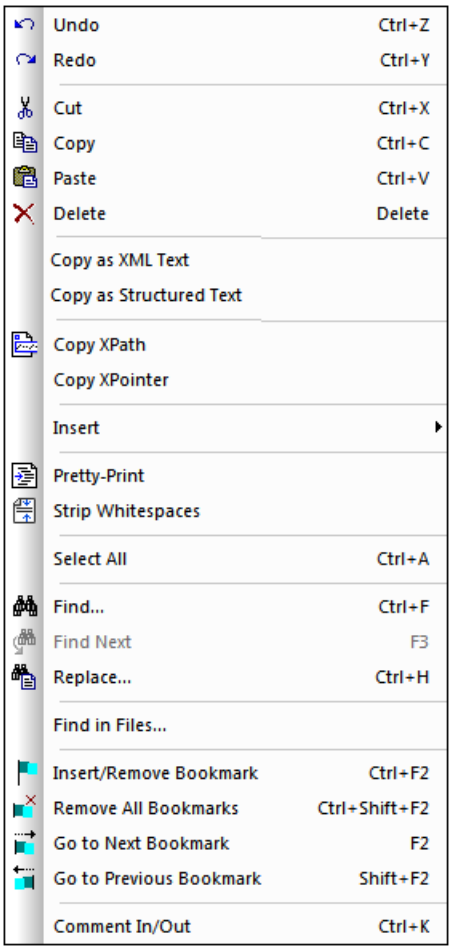
At the bottom of the **File** menu is a list of the nine most recently used files, with the most recently opened file shown at the top of the list. You can open any of these files by clicking its name. To open a file in the list using the keyboard, press **Alt+F** to open the **File** menu, and then press the number of the file you want to open.

Exit

Quits XMLSpy. If you have any open files with unsaved changes, you are prompted to save these changes. XMLSpy also saves modifications to program settings and information about the most recently used files.

26.2 Edit Menu

The **Edit** menu contains commands for editing documents in XMLSpy. These include the familiar Undo, Redo, Cut, Copy, Paste, Delete, Select All, Find, Find Next and Replace commands.



XMLSpy also offers special commands to:

- copy a selection to the clipboard as XML- Text,
- copy as structured text
- copy an XPath selector to the selected item to the clipboard.
- insert and remove bookmarks, and to navigate to bookmarks.

26.2.1 Undo, Redo

Icons and shortcuts

Command	*Icon*	*Shortcut*
Undo		**Ctrl+Z**
Redo		**Ctrl+Y**

Undo

The **Undo** command contains support for unlimited levels of Undo. Every action can be undone and it is possible to undo one command after another. The Undo history is retained after using the **Save** command, enabling you go back to the state the document was in before you saved your changes. You can step backwards and forwards through this history using the **Undo** and **Redo** commands (*see **Redo** command below*).

Redo

The **Redo** command allows you to redo previously undone commands, thereby giving you a complete history of work completed. You can step backwards and forwards through this history using the **Undo** and **Redo** commands.

26.2.2 Cut, Copy, Paste, Delete

Icons and shortcuts

Command	*Icon*	*Shortcut*
Cut		**Ctrl+X** or **Shift+Del**
Copy		**Ctrl+C**
Paste		**Ctrl+V**
Delete		**Del**

Cut

The **Cut** command copies the selected text or items to the clipboard and deletes them from their present location.

Copy

The **Copy** command copies the selected text or items to the clipboard. This can be used to duplicate data within XMLSpy or to move data to another application.

Note: When copying from Grid View, the selection is copied using one of two methods: **Copy as XML-Text** and **Copy as Structured Text**. The former copies the selection as XML text; the latter copies the selection as a table. Which of these two methods is used when the **Copy** command is invoked is specified in the Editing tab of the Tools | Options dialog.

Paste

The **Paste** command inserts the contents of the clipboard at the current cursor position.

Delete

The **Delete** command deletes the currently selected text or items without placing them in the clipboard.

26.2.3 Copy as XML Text

The **Copy as XML Text** command copies XML data from Grid View as XML text. Highlight any XML data in Standard Grid View or in Table View (which is another view available in Grid View), and select the **Copy As XML Text** command. The data is copied to the clipboard as XML text (*as in the listing below)*. You can then paste this text in another document.

```
<row>
   <para align="left">
      <bold>Check the FAQ</bold>
   </para>
   <para>
      <link mode="internal">
         <link_section>support</link_section>
         <link_subsection>faq30</link_subsection>
         <link_text>XMLSPY 4.0 FAQ</link_text>
      </link>
      <link mode="internal">
         <link_section>support</link_section>
         <link_subsection>faq25</link_subsection>
         <link_text>XMLSPY 3.5 FAQ</link_text>
      </link>
   </para>
</row>
```

The **Copy as XML Text** command automatically formats text using the currently active settings for saving a file. These settings can be modified in the *Save File* section of the File tab of the Options dialog (**Tools | Options**).

The same effect as that of the **Copy as XML Text** command can be obtained by switching to Text View and copying an XML text fragment with **Ctrl+C** (**Edit | Copy**). From Grid View, you can also use the **Copy as Structured Text** command to copy XML data in its tabular Grid View representation.

26.2.4 Copy as Structured Text

The **Copy as Structured Text** command copies elements to the clipboard as they appear on screen. This command is especially useful for copying table-like data from the Table View of Grid View into another application that holds data in tabular form (such as a spreadsheet application).

- Copying individual rows from Table View
- Copying a whole table from Table View
- Copying from standard Grid View

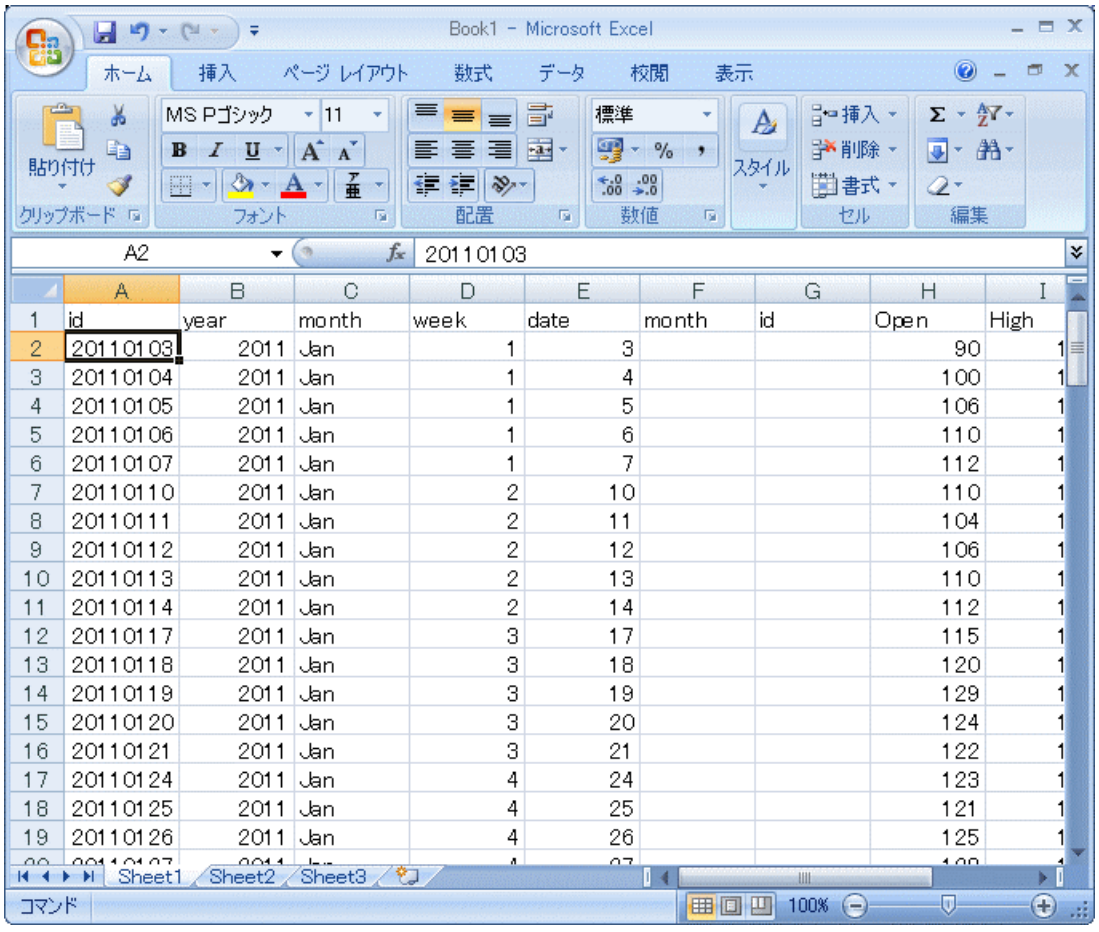
Copying individual rows from Table View

In the screenshot below two rows in the **Table View** of Grid View are copied as structured text.

item (6)

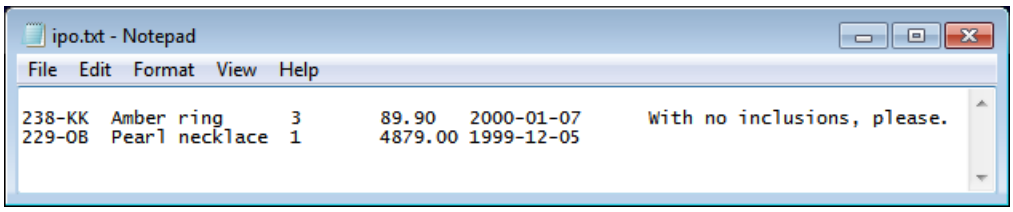
	partNum	productName	quantity	price	shipDate	ipo:comment
1	833-AA	Lapis necklace	2	99.95	1999-12-05	Need this for the holidays!
2	748-OT	Diamond heart	1	248.90	2000-02-14	Valentine's day packaging.
3	783-KL	Uncut diamond	7	79.90	2000-01-07	
4	238-KK	Amber ring	3	89.90	2000-01-07	With no inclusions, please.
5	229-OB	Pearl necklace	1	4879.00	1999-12-05	
6	128-UL	Jade earring	5	179.90	2000-02-14	

The next two screenshots below show the data that was copied as structured text in the Grid View screenshot shown above now pasted in a Microsoft Excel document and in a Notepad document.



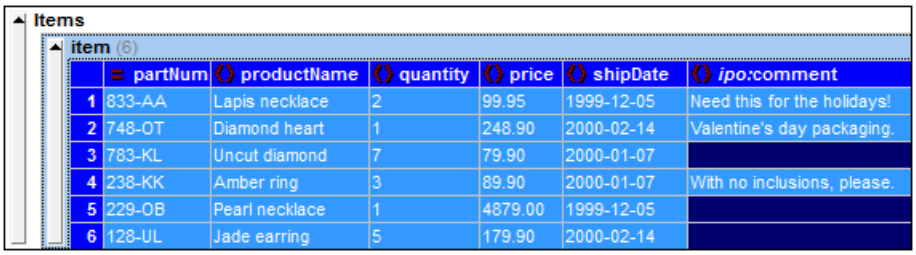
	A	B	C	D	E	F	G	H	I
1	id	year	month	week	date	month	id	Open	High
2	20110103	2011	Jan	1	3			90	1
3	20110104	2011	Jan	1	4			100	1
4	20110105	2011	Jan	1	5			106	1
5	20110106	2011	Jan	1	6			110	1
6	20110107	2011	Jan	1	7			112	1
7	20110110	2011	Jan	2	10			110	1
8	20110111	2011	Jan	2	11			104	1
9	20110112	2011	Jan	2	12			106	1
10	20110113	2011	Jan	2	13			110	1
11	20110114	2011	Jan	2	14			112	1
12	20110117	2011	Jan	3	17			115	1
13	20110118	2011	Jan	3	18			120	1
14	20110119	2011	Jan	3	19			129	1
15	20110120	2011	Jan	3	20			124	1
16	20110121	2011	Jan	3	21			122	1
17	20110124	2011	Jan	4	24			123	1
18	20110125	2011	Jan	4	25			121	1
19	20110126	2011	Jan	4	26			125	1

Notice that while Excel (*screenshot above*) automatically formats each piece of cell text on the basis of its lexical form of text, Notepad (*screenshot below*) pastes all cell text as strings.



Copying a whole table from Table View

In the screenshot below, a whole table is selected (by clicking the `item (6)` node entry).



Items

item (6)

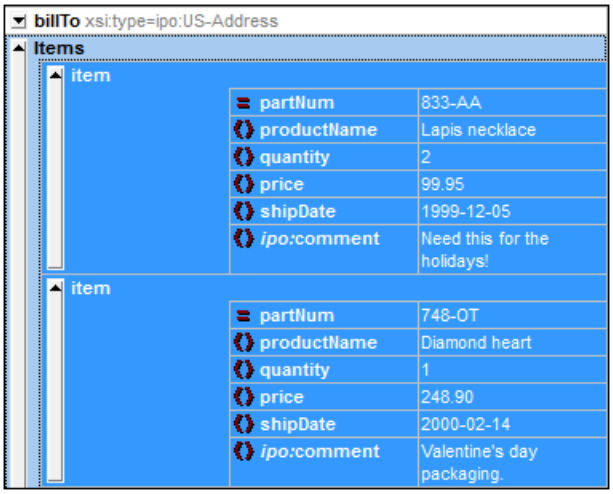
	partNum	productName	quantity	price	shipDate	ipo:comment
1	833-AA	Lapis necklace	2	99.95	1999-12-05	Need this for the holidays!
2	748-OT	Diamond heart	1	248.90	2000-02-14	Valentine's day packaging.
3	783-KL	Uncut diamond	7	79.90	2000-01-07	
4	238-KK	Amber ring	3	89.90	2000-01-07	With no inclusions, please.
5	229-OB	Pearl necklace	1	4879.00	1999-12-05	
6	128-UL	Jade earring	5	179.90	2000-02-14	

The result is as for individual rows above, except that the table's column titles are now also copied. The screenshot below shows the copied data pasted into a Microsoft Excel worksheet. Note that, in the XML document, the column headers are element or attribute names.

	A	B	C	D	E	F
1	partNum	productName	quantity	price	shipDate	ipo:comment
2	833-AA	Lapis necklace	2	99.95	12/05/1999	Need this for the holidays!
3	748-OT	Diamond heart	1	248.9	02/14/2000	Valentine's day packaging.
4	783-KL	Uncut diamond	7	79.9	01/07/2000	
5	238-KK	Amber ring	3	89.9	01/07/2000	With no inclusions, please.
6	229-OB	Pearl necklace	1	4879	12/05/1999	
7	128-UL	Jade earring	5	179.9	02/14/2000	

Copying from Standard Grid View

The examples shown above copied data from the Table View of Grid View (that is, with Table View toggled on in Grid View). In Standard Grid View (Table View toggled off), the **Copy as Structured Text** also copies the selected data as displayed in Grid View. However, the structure is not a table. For example, in the Standard Grid View screenshot below, the `items` node, which has two `item` children, is selected.



billTo xsi:type=ipo:US-Address

Items

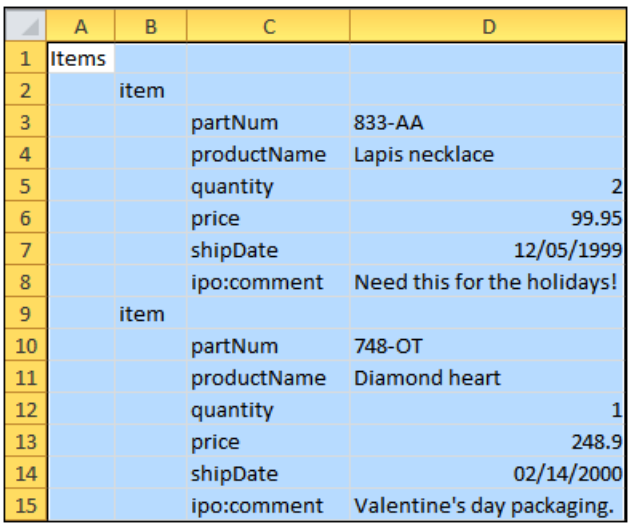
item

partNum	833-AA
productName	Lapis necklace
quantity	2
price	99.95
shipDate	1999-12-05
ipo:comment	Need this for the holidays!

item

partNum	748-OT
productName	Diamond heart
quantity	1
price	248.90
shipDate	2000-02-14
ipo:comment	Valentine's day packaging.

When the copied text is pasted in Microsoft Excel, the worksheet will look like the screenshot below.



	A	B	C	D
1	Items			
2		item		
3			partNum	833-AA
4			productName	Lapis necklace
5			quantity	2
6			price	99.95
7			shipDate	12/05/1999
8			ipo:comment	Need this for the holidays!
9		item		
10			partNum	748-OT
11			productName	Diamond heart
12			quantity	1
13			price	248.9
14			shipDate	02/14/2000
15			ipo:comment	Valentine's day packaging.

In addition to copying the XML data as structured text with the Copy as Structured Text command, data can also be copied from Grid View as XML text.

26.2.5 Copy XPath

The **Copy XPath** command is available in Text View and Grid View, and creates an XPath expression that locates the currently selected node/s in the document, and copies the XPath expression to the clipboard. This enables you to paste the XPath expression into a document (for example, in an XSLT document). All expressions start from the document root. For example, if an element called `LastName` of the third `Person` element of the second `Company` element is selected, the XPath expression tha is copied would be: `/Companies/Company[2]/Person[3]/LastName`

Note: In Grid View the **Copy XPath** command can also be accessed via the context menu.

26.2.6 Copy XPointer

The **Copy XPointer** command is available in Text View and Grid View. It creates an `element()` scheme XPointer for the currently selected node/s and copies it to the clipboard. This enables you to paste the XPointer into a document (for example, in the `xpointer` attribute of an XInclude element in an XML document).

The `element()` scheme of XPointer returns results in the form `element(/1/3)`. This XPointer selects the third child of the document element (aka root element). You should note the following points:

- Attributes cannot be represented using the `element()` scheme. If an attribute is selected in XMLSpy, the following happens: In Grid View, the **Copy XPointer** command is disabled; in Text View, the XPointer of the parent element of that attribute is generated.
- XPointers for multiple elements cannot be generated. If multiple elements are selected, the situation is resolved as follows: In Grid View, the **Copy XPointer** command is disabled. In Text View, the XPointer of the parent element of the selection is generated.

Note: In Grid View the **Copy XPointer** command can also be accessed via the context menu.

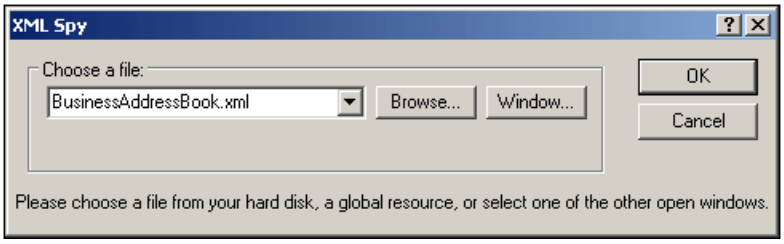
26.2.7 Insert

Mousing over or selecting the **Insert** command rolls out a submenu with three commands (described below):

- Insert File Path
- Insert XInclude
- Insert Encoded External File

Insert File Path

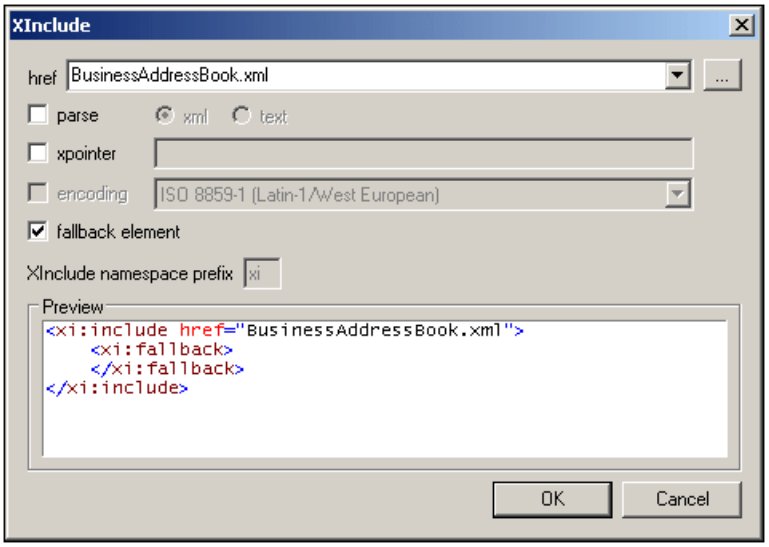
The **File Path** command is enabled in the Text View and Grid View of documents of any file type. Using it, you can insert the path to a file at the cursor selection point. Clicking the command pops up a dialog (*screenshot below*) in which you select the required file.



The required file can be selected in one of the following ways: (i) by browsing for the file, URL, or global resource (use the **Browse** button); (ii) by selecting the window in which the file is open (the **Window** button). When done, click **OK**. The path to the selected file will be inserted in the active document at the cursor selection point.

Insert XInclude

The **XInclude** command is available in Text View and Grid View, and enables you to insert a new XInclude element at the cursor selection point in Text View, or before the selected item in both Text View and Grid View. If in Grid View the current selection is an attribute, the XInclude element is inserted after the attribute and before the first child element of the attribute's parent element. Selecting this command pops up the XInclude dialog (*screenshot below*).



The XML file to be included is entered in the `href` text box (alternatively, you can browse for the file by clicking the **Browse (...)** button to the right of the text box). The filename will be entered in the XML document as the value of the `href` attribute. The `parse`, `xpointer`, and `encoding` attributes of the XInclude element (`xi:include`), and the `fallback` child element of `xi:include` can also be inserted via the dialog. Do this by first checking the appropriate check box and then selecting/entering the required values. In the case of the `fallback` element, checking its check box only inserts the empty element. The content of the `fallback` element must be added subsequently in one of the editing views.

The `parse` attribute determines whether the included document is to be parsed as XML or text. (XML is the default value and therefore need not be specified.) The `xpointer` attribute identifies a specific fragment of the document located with the `href` attribute; it is this fragment that will be included. The `encoding` attribute specifies the encoding of the included document so that XMLSpy can transcode this document (or the part of it to be included) into the encoding of the including document. The contents of the `fallback` child element replace the `xi:include` element if the document to be included cannot be located.

Here is an example of an XML document that uses XInclude to include two XML documents:

```
<?xml version="1.0" encoding="UTF-16"?>
<AddressBook xsi:schemaLocation="http://www.altova.com/sv/myaddresses
AddressBook.xsd"
       xmlns="http://www.altova.com/stylevision/tutorials/myaddresses"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="BusinessAddressBook.xml"/>
       <xi:include href="PersonalAddressBook.xml"/>
</AddressBook>
```

When this XML document is parsed, it will replace the two XInclude elements with the files specified in the respective `href` attributes.

xml:base

When the XML validator of XMLSpy reads an XML document and encounters the `include` element in the XInclude namespace (hereafter `xi:include`), it replaces this element (`xi:include`) with the XML document named in the `href` attribute of the `xi:include` element. The document element (root element) of the included XML document (or the element identified by

an XPointer) will be included with an attribute of `xml:base` in order to preserve the base URIs of the included element. If the resulting XML document (containing the included XML document/s or tree fragment/s) must be valid according to a schema, then the document element of the included document (or the top-level element of the tree fragment) must be created with a content model that allows an attribute of `xml:base`. If, according to the schema, the `xml:base` attribute is not allowed on this element, then the resulting document will be invalid. How to define an `xml:base` attribute in an element's content model using XMLSpy's Schema View is described in the xml:Prefixed Attributes section of the Schema View section of the documentation.

XPointers

XMLSpy supports XPointers in XInclude. The relevant W3C recommendations are the XPointer Framework and XPointer element() Scheme recommendations. The use of an XPointer in an XInclude element enables a specific part of the XML document to be included, instead of the entire XML document. XPointers are used within an XInclude element as follows:

```
<xi:include href="PersonalAddressBook.xml" xpointer="element(usa)"/>
<xi:include href="BusinessAddressBook.xml" xpointer="element(/1/1)"/>
<xi:include href="BobsAddressBook.xml" xpointer="element(usa/3/1)"/>
<xi:include href="PatsAddressBook.xml" xpointer="element(usa)
element(/1/1)"/>
```

In the element() scheme of XPointer, an NCName or a child sequence directed by integers may be used.

- In the first `xi:include` element listed above, the `xpointer` attribute uses the element scheme with an NCName of `usa`. According to the XPointer Framework, this NCName identifies the element that has an ID of `usa`.
- In the second `xi:include` listed above, the `xpointer` attribute with a value of `element(/1/1)` identifies, in the first step, the first child element of the document root (which, if the document is well-formed, will be its document (or root) element). In the second step, the first child element of the element located in the previous step is located; in our example, this would be the first child element of the document element.
- The `xpointer` attribute of the third `xi:include` listed above uses a combination of NCName and child sequence. This XPointer locates the first child element of the third child element of the element having an ID of `usa`.
- If you are not sure whether your first XPointer will work, you can back it up with a second one as shown in the fourth `xi:include` listed above: `xpointer="element(usa) element(/1/1)"`. Here, if there is no element with an ID of `usa`, the back-up XPointer specifies that the first child element of the document element is to be selected. Additional backups are also allowed. Individual XPointers may not be separated, or they may be separated by whitespace: for example, `xpointer="element(usa) element(addresses/1) element(/1/1)"`.

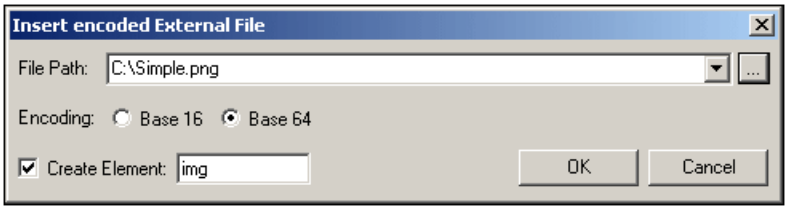
Note: The namespace binding context is not used in the element() scheme because the element() scheme does not support qualified names.

Insert Encoded External File

The **Encoded External File** command is available in Text View and Grid View. It enables an external file to be included as encoded Base-16 or Base-64 text at any location in the XML document. This feature enables external files to be embedded in the XML document.

Clicking the **Insert | Encoded External File** command pops up the Insert Encoded External File

dialog (*screenshot below*).



You can browse for or enter the name of the external file to be encoded and embedded. Either a Base-16 or Base-64 encoding must be specified. If you wish to enclose the encoded text in an element, then check the Create Element check box and specify the name of the desired element in the Create Element text box. If the Create Element check box is not checked, then the encoded text will be inserted directly at the cursor location.

On clicking **OK**, the encoded text of the selected file is inserted at the cursor location, with an enclosing element if this has been specified.

```
<img ext="png" encoding="xs:base64Binary">
  iVBORw0KGgoAAAANSUhEUgAAABAAAAAQAQMAAAA1Pw0iAAAAB3BMVEUAAAD/
  //+l2Z/dAAAAM0lEQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83NDDeNGe4U
  g9C9zwz3gVLMDA/A6P9/AFGGFyjOXZtQAAAAAElFTkSuQmCC
</img>
```

The listing above shows the encoded text of a PNG image file. An `img` element was created around the encoded text.

26.2.8 Pretty-Print

Icon

Icon:	

Description

The **Pretty-Print** command re-formats your XML document in Text View. Two formatting options are available: which one is used depends upon whether the *Use Indentation* check box in the View tab of the Options dialog (**Tools | Options**) is currently checked or not:

- *Use Indentation* checked: The document is re-formatted to give a structured display, indenting each deeper level in the hierarchy by an additional amount of the specified indentation space (this space being specified in the Text View Settings dialog (**View | Text View Settings**)). Indentation enables a clearer view of the document structure.
- *Use Indentation* unchecked: The document is re-formatted so that each new line is left-aligned.

Procedure for using Pretty-Print

To set up a structured, indented view of the XML document via Pretty-Print, do the following:

1. In the View tab of the Options dialog (**Tools | Options**), check the *Use Indentation* check box.
2. In the Text View Settings dialog (**View | Text View Settings**), set the tab size you want

for the indentation of the pretty-printed text.
3. In the File tab of the Options dialog (**Tools | Options**), enter the elements for which no output formatting (indentation) is wanted.
4. Click the **Pretty-Print** command (this command).

To re-format the document so that all lines are left-aligned, uncheck the *Use Indentation* check box.

Note the following points

- The XML document must be well-formed for this command to work.
- Pretty-printing adds spaces or tabs to the document when the document is saved.
- If pretty-printing has been switched on (**Tools | Options | View | Use Indentation**) and if you change from Text View to Grid View and back to Text View, then the document will be pretty-printed automatically. There is no need to select the Pretty-Print command.
- To remove all whitespace (new lines and indentation) created with the Pretty-Print command, use the **Strip Whitespaces** command.

26.2.9 Strip Whitespaces

Icon

Icon:	

Description

The **Strip Whitespaces** command strips all whitespace from the document. This can help reduce file size. This command can be useful if you wish to remove whitespace generated by the **Pretty-Print** command.

26.2.10 Select All

The **Select All** command (**Ctrl+A**) selects the contents of the entire document.

26.2.11 Find, Find Next

Icons and shortcuts

Command	*Icon*	*Shortcut*
Find		**Ctrl+F**
Find Next		**F3**

Find

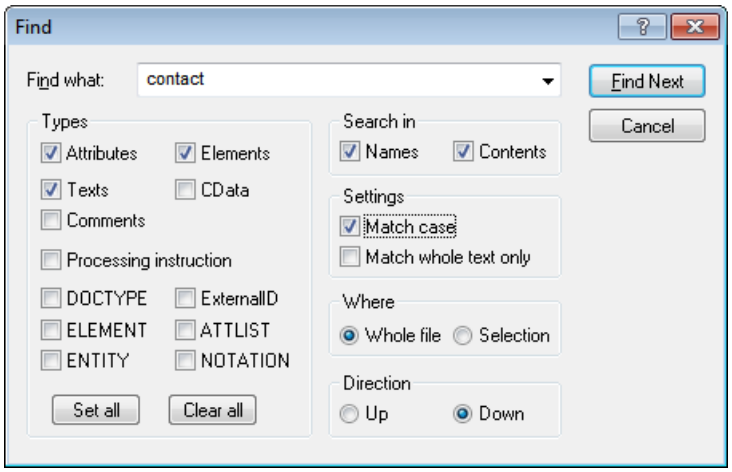
The **Find** command displays the Find dialog, in which you can specify the string you want to find and other options for the search. Depending on the current view, the Find dialog displays

different options. To find text, enter the text in the Find field or use the combo box to select from one of the last 10 search criteria, and then specify the options for the search.

The **Find** and **Find Next** commands can also be used to find file and folder names when a project is selected in the Project window.

Grid View

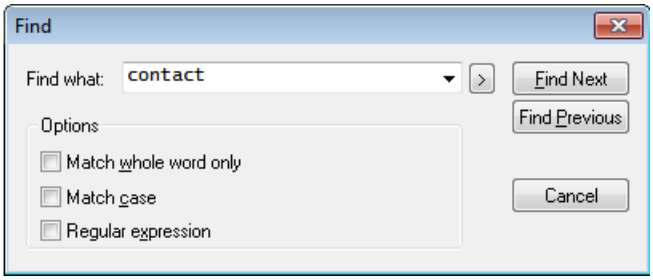
In Grid View, the following dialog box appears. Select the options you require or select a radio button. The options available are described below.



- The *Types* pane allows you to select what XML document nodes or components you wish to include in the search. This enables the search to skip particular node types and thus go faster. The **Set All** button selects all node types; the **Clear All** button deselects all node types.
- The *Search In* pane allows you to define whether the names of a node, the contents of a node, or both should be searched for the input text string.
- The *Settings* pane enables you to define whether the search should be case- sensitive and/ or match the entire input string.
- The *Where* pane allows you to define the scope of the search (the whole file or the selected text).
- The *Direction* option specifies the search direction.

Text View

For a description of the Find function of editable Text Views, see the section Text View. The Find dialog of non-editable Text Views (such as in the XPath/XQuery window) is shown in the screenshot below.



The available options are as follows:

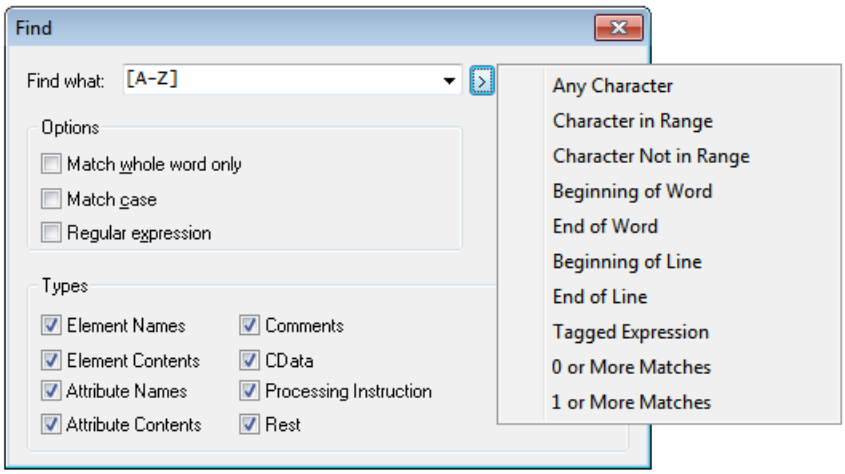
- *Match whole word only*: Only the exact words in the text will be matched. For example, for the input string `fit`, with *Match whole word only* checked, only the word `fit` will match the find string; the `fit` in `fitness`, for example, will not.
- *Match case*: Case- sensitive search (`Address` is not the same as `address`).
- *Regular expression*: Searches for text specified by the regular expression you enter in the text box. See Regular expressions below for a description of regular expressions.

Note the following points:

- The Find dialog is "modeless", which means that it can remain open while you continue to use Text View. Pressing **Enter** while the dialog box is open, closes the dialog box. If text is marked prior to opening the dialog box, then the marked text is automatically inserted into the Find What text box.
- Once the Find dialog is closed, you can repeat the current search by pressing **F3** for a forwards search, or **Shift+F3** for a backwards search.
- The **Unfold** button to the right of the *Find What* input box (the button marked **>**), opens a secondary window which you can use to enter regular expressions in the *Find What* input box.

Regular expressions

You can use regular expressions to further refine your search criteria. A pop-up list is available to help you build regular expressions. To access this list, click the **Unfold** button to the right of the *Find What* input box (the button marked **>**).



Clicking on the required expression description inserts the corresponding expression syntax in the input field. Given below is a list of regular expression syntax characters.

`.`	Matches any character. This is a placeholder for a single character.
`(`	Marks the start of a region for tagging a match.
`)`	Marks the end of a tagged region.
`\n`	Where `n` is `1` through `9` refers to the first through ninth tagged region when replacing. For example, if the search string was `Fred([1-9])XXX` and the replace string was `Sam\1YYY`, when applied to `Fred2XXX` this

	would generate `Sam2YYY`.
`\<`	Matches the start of a word.
`\>`	Matches the end of a word.
`\x`	Allows you to use a character `x`, which would otherwise have a special meaning. For example, `\[` would be interpreted as `[` and not as the start of a character set.
`[...]`	Indicates a *set of characters*. For example, `[abc]` means any of the characters `a`, `b` or `c`. You can also use ranges: for example `[a-z]` for any lower case character.
`[^...]`	The complement of the characters in the set. For example, `[^A-Za-z]` means any character except an alphabetic character.
`^`	Matches the start of a line (unless used inside a set, *see above*).
`$`	Matches the end of a line. Example: `A+$` to find one or more `A`'s at end of line.
`*`	Matches 0 or more times. For example, `Sa*m` matches `Sm`, `Sam`, `Saam`, `Saaam` and so on.
`+`	Matches 1 or more times. For example, `Sa+m` matches `Sam`, `Saam`, `Saaam` and so on.

Note: Regular expressions are not supported in the Replace field.

Find Next

The **Find Next** command repeats the last Find command. It searches for the next occurrence of the input text.

26.2.12 Replace

Icons and shortcuts

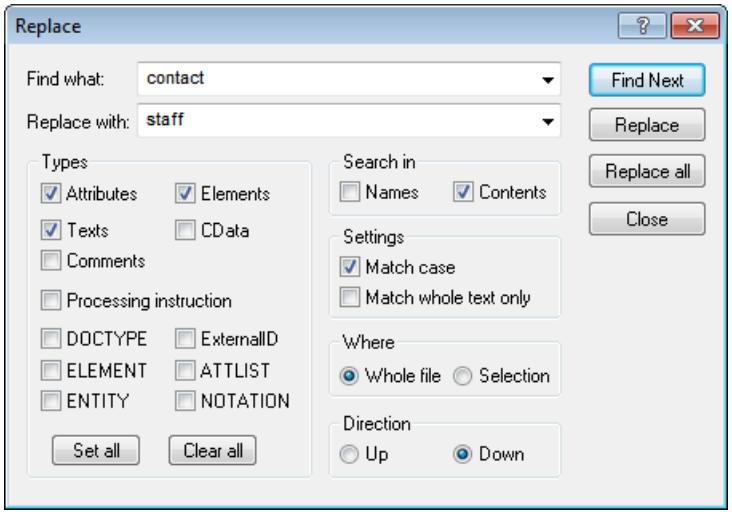
Command	*Icon*	*Shortcut*
Replace		**Ctrl+H**

Description

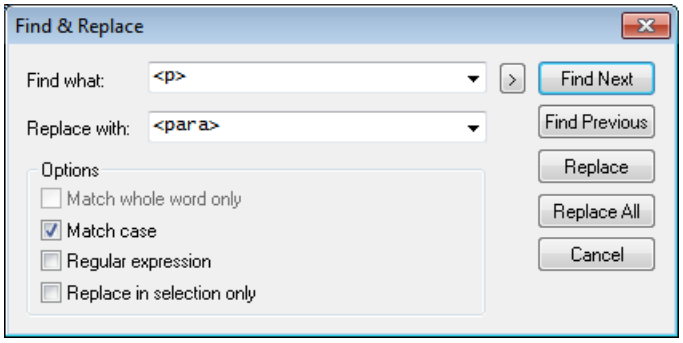
The **Replace** command enables you to find and replace one text string with another. It features the same options as the **Find** command. Depending on the view you are using, the Replace dialog displays different find options. You can replace each item individually, or you can use the **Replace All** button to perform a global find-and-replace operation.

Grid View

In Grid View, selecting the **Replace** command opens the Replace dialog (*screenshot below*). The screenshot shows the various Find options, which are described in the description of the **Find** command.



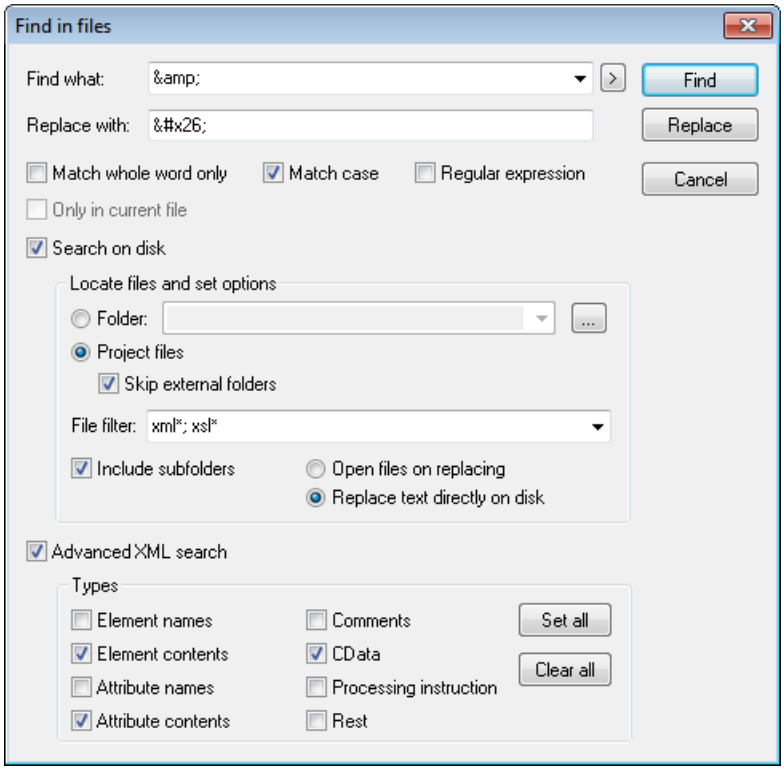
Text View
For a description of the Replace function of editable Text Views, see the section Text View. The Replace function of non-editable Text Views (such as in the XPath/XQuery window) is described below. In Text View, selecting the **Replace** command opens the Find & Replace dialog (*screenshot below*). The Find options are the same as for the **Find** command. The *Replace in selection only* option carries out the find-and-replace operation only within the text selection.



Note: When using the **Replace All** command, each replacement is recorded as a single operation, so **Replace All** can be undone step-by-step.

26.2.13 Find in Files

The **Find in Files** command is a powerful way to find and replace text quickly among a large number of files. Clicking the command pops up the Find in Files dialog (*screenshot below*). The **Find in Files** command is different from the **Find** command in that it searches all the specified locations for the Find string at once and executes replace actions immediately. A report is then displayed in the Find in Files output window. In the case of the **Find** command, however, the user enters the search string and goes through the (single) active document one found item at a time.



Find criteria

There are two broad find criteria: (i) what to find, and (ii) where to look? For a description of how to set the text that is to be searched (what to find), see the description of the **Find** command. If the text entered in the Find What text box is a regular expression, then the Regular Expression check box must be checked. An entry helper for regular expressions can be accessed by clicking the **Unfold** button to the right of the *Find What* input box (the button marked **>**) button. The use of regular expressions for searching is explained in the section about the **Find** command.

To specify what node types and parts of an XML document should be searched, check the Advanced XML Search check box and then select the required node types.

You can specify what files should be searched by checking either the *Only in Current File* check box or the *Search on Disk* check box. If you choose to search on disk, you can select a folder or a project to search (after checking the *Search On Disk* check box). When a project folder is selected, external folders added to the project can be skipped. The files to be searched can be filtered by file extension and a star (`xml*` or `xsl*`, for example). The separator between two file extensions can be a comma or a semi-colon (`xml*;xsl*`, for example). The star character can also be used as a wildcard.

The instances of the Find string at all the search locations are listed in the Find in Files output bar. Clicking on one of the listed items opens that file in Text View and highlights the item.

Replace

The most important thing to note is that clicking the **Replace** button replaces all the instances of the Find string with the Replace string. If *Open Files On Replacing* was checked in the Find in Files dialog, then the file will be opened in Text View; otherwise the replacement is done silently. All the replaced strings are listed in the Find in Files output bar. Clicking on one of the listed items opens that file in Text View and highlights the item.

Note: Regular expressions are not supported in the Replace field.

26.2.14 Bookmark Commands

Icons and shortcuts

Command	*Icon*	*Shortcut*
Insert/Remove Bookmark		**Ctrl+F2**
Remove All Bookmarks		**Ctrl+Shift+F2**
Goto Next Bookmark		**F2**
Goto Previous Bookmark		**Shift+F2**

Insert/Remove Bookmark

The **Insert/Remove Bookmark** command inserts a bookmark at the current cursor position, or removes the bookmark if the cursor is in a line that has been bookmarked previously. This command is only available in Text View.

Bookmarked lines are displayed in one of the following ways:

- If the bookmarks margin has been enabled, then a solid blue ellipse appears to the left of the text in the bookmark margin.
- If the bookmarks margin has not been enabled, then the entire line containing the cursor is highlighted.

The **F2** key cycles through all the bookmarks in the document.

Remove All Bookmarks

The **Remove All Bookmarks** command removes all the currently defined bookmarks. This command is only available in Text View. Note that the **Undo** command does not undo the effects of **Remove All Bookmarks**.

Goto Next Bookmark

The **Goto Next Bookmark** command places the text cursor at the beginning of the next bookmarked line. This command is only available in Text View.

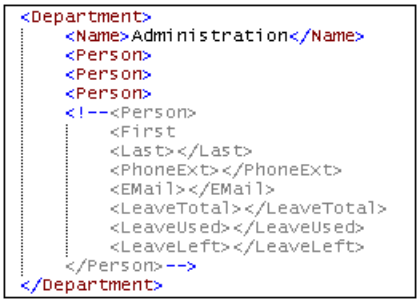
Goto Previous Bookmark

The **Goto Previous Bookmark** command places the text cursor at the beginning of the previous bookmarked line. This command is only available in Text View.

26.2.15 Comment In/ Out

The **Comment In/Out** command is available in Text View and is used to comment and uncomment XML text fragments. Text in an XML document can be commented out using the XML start-comment and end-comment delimiters, respectively `<!--` and `-->`. In XMLSpy, these comment delimiters can be inserted around a text selection by using the **Comment In/Out** menu command.

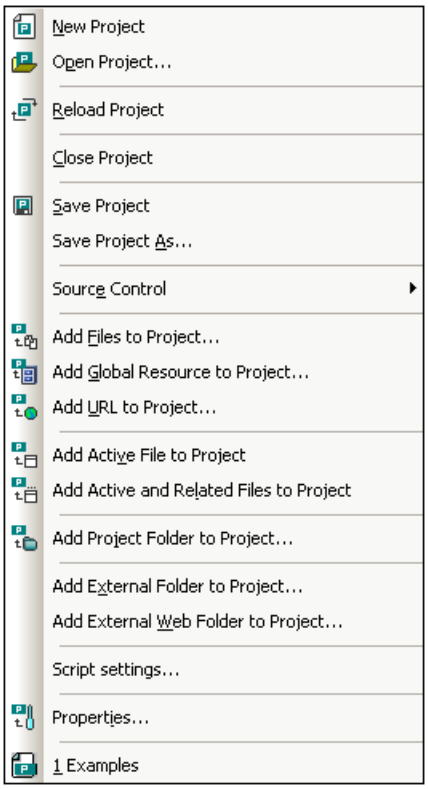
To comment out a block of text, select the text to be commented out and then select the command **Comment In/Out**, either from the **Edit** menu or the context menu you get on right-clicking the selected text. The commented text will be grayed out (*see screenshot below*).



To uncomment a commented block of text, place the cursor in the commented block and select the command **Comment In/Out**, either from the **Edit** menu or the context menu you get on right-clicking within the commented-out text. The comment delimiters will be removed and the text will no longer be grayed out.

26.3 Project Menu

XMLSpy uses the familiar tree view to manage multiple files or URLs in XML projects. Files and URLs can be grouped into folders by common extension or any arbitrary criteria, allowing for easy structuring and batch manipulation.



Please note: Most project-related commands are also available in the context menu, which appears when you right-click any item in the project window.

Absolute and relative paths

Each project is saved as a project file, and has the *spp* extension. These files are actually XML documents that you can edit like any regular XML File. In the project file, absolute paths are used for files/ folders on the same level or higher, and relative paths for files/ folders in the current folder or in sub- folders. For example, if your directory structure looks like this:

```
├Folder1
|     |
|     ├Folder2
|           |
|           ├Folder3
|                 |
|                 ├Folder4
```

If your *.spp* file is located in `Folder3`, then references to files in `Folder1` and `Folder2` will look something like this:

```
c:\Folder1\NameOfFile.ext
c:\Folder1\Folder2\NameOfFile.ext
```

References to files in Folder3 and Folder4 will look something like this:

```
.\NameOfFile.ext
.\Folder4\NameOfFile.ext
```

If you wish to ensure that all paths will be relative, save the .spp files in the root directory of your working disk.

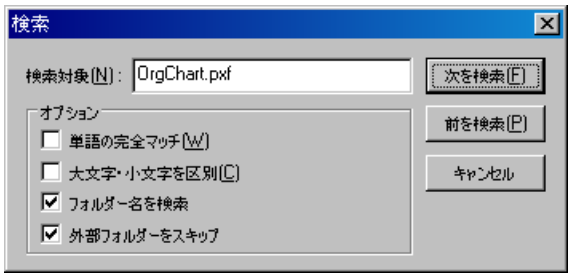
Drag-and-drop

In the Project window, a folder can be dragged to another folder or to another location within the same folder. A file can be dragged to another folder, but cannot be moved within the same folder (within which files are arranged alphabetically). Additionally, files and folders can be dragged from Windows File Explorer to the Project window.

Find in project

You can search for project files and folders using their names or a part of their name. If the search is successful, files or folders that are located are highlighted one by one.

To start a search, select the project folder in the Project sidebar that you wish to search, then select the command **Edit | Find** (or the shortcut **Ctrl+F**). In the Find dialog that pops up (*screenshot below*) enter the text string you wish to search for and select or deselect the search options (*explained below*) according to your requirements.



The following search options are available:

- Whole-word matching is more restricted since the entire string must match an entire word in the file or folder name. In file names, the parts before and after the dot (without the dot) are each treated as a word.
- It can be specified that casing in the search string must exactly match the text string in the file or folder name.
- Folder names can be included in the search. Otherwise, only file names are searched.
- External folders can be included or excluded from the search. External folders are actual folders on the system or network, as opposed to project folders, which are created within the project and not on the system.

If the search is successful, the first matching item is highlighted in the Project sidebar. You can then browse through all the returned matching items by clicking the **Find Next** and **Find Prev** buttons in the Find dialog.

Refreshing projects

If a change is made to an external folder, this change will not be reflected in the Project Window till the project is refreshed.

Global resources in the context menu

When you right-click a folder in the Project window, in the context menu that appears, you can select the **Add Global Resource** menu item to add a global resource. The menu command itself pops up the Choose Global Resource dialog, which lists all the file-type and folder-type global resources in the currently active Global Resources XML File. Select the required global resource, and it will be added to the selected project folder.

Projects and source control providers

If you intend to add an XMLSpy project to a source control repository, please ensure that the project files position in the hierarchical file system structure is one which enables you to add files only from below it (taking the root directory to be the top of the directory tree).

In other words, the directory where the **project file** is located, essentially represents the **root directory** of the project within the source control repository. Files added from above it (the project root directory) will be added to the XMLSpy project, but their location in the repository may be an unexpected one—if they are allowed to be placed there at all.

For example, given the directory structure show above, if a project file is saved in `Folder3` and placed under source control:

- Files added to Folder1 may not be placed under source control,
- Files added to Folder2 are added to the root directory of the repository, instead of to the project folder, but are still under source control,
- Files located in Folder3 and Folder4 work as expected, and are placed under source control.

26.3.1 New Project

The **New Project** command creates a **new** project in XMLSpy. If you are currently working with another project, a prompt appears asking if you want to close all documents belonging to the current project.

26.3.2 Open Project

The **Open Project...** command opens an existing project in XMLSpy. If you are currently working with another project, the previous project is closed first.

26.3.3 Reload Project

The **Reload Project** command reloads the current project from disk. If you are working in a multi-user environment, it can sometimes become necessary to reload the project from disk, because other users might have made changes to the project.

Please note: Project files (spp files) are actually XML documents that you can edit like any regular XML File.

26.3.4 Close Project

The **Close Project** command **closes** the active project. If the project has been modified, you will be asked whether you want to save the project first. When a project is modified in any way, an asterisk is added to the project name in the Project Window.

26.3.5 Save Project, Save Project As

The **Save Project** command **saves** the current project. You can also save a project by making the project window active and clicking the icon.

The **Save Project As** command **saves** the current project with a new name that you can enter when prompted for one.

26.3.6 Source Control

Your Altova application supports Microsoft SourceSafe and other compatible repositories. A list of supported systems is given in the section, Supported Source Control Systems. This section describes the commands in the **Project | Source Control** submenu, which are used to work with the source control system from within your Altova application.

Overview of the Source Control feature

The mechanism for placing files in an application project under source control is as follows:

1. In XMLSpy, an application project folder containing the files to be placed under source control is created. Typically, the application project folder will correspond to a local folder in which the project files are located. The path to the local folder is referred to as the local path.
2. In the source control system's database (also referred to as source control or repository), a folder is created that will contain the files to be placed under source control.
3. Application project files are added to source control using the command **Project | Source Control | Add to Source Control**.
4. Source control actions, such as checking in to, checking out from, and removing files from source control, can be carried out by using the commands in the **Project | Source Control** submenu. The commands in this submenu are listed in the sub-sections of this section.

Note: If you wish to change the current source control provider, this can be done in any of two ways: (i) via the Source Control options (**Tools | Options | Source Control**), or (ii) in the Change Source Control dialog (**Project | Source Control | Change Source Control**).

Note: Note that a source control project is not the same as an application project. Source control

projects are directory-dependent, while XMLSpy projects are logical constructions without direct directory dependence.

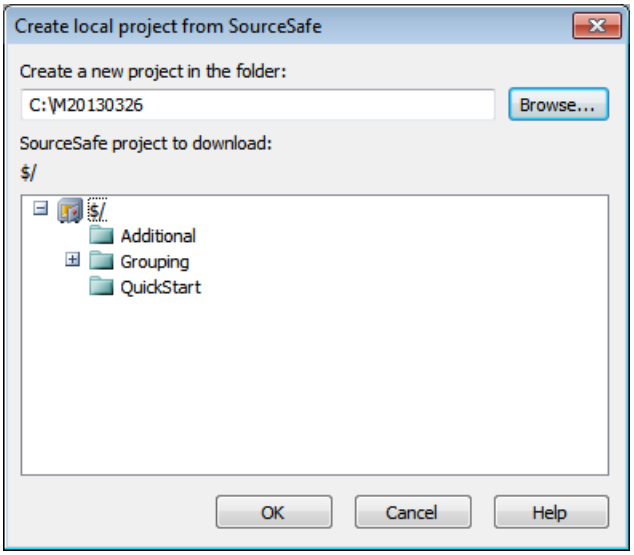
For additional information, see the section, Source Control.

Open from Source Control

The **Open from Source Control** command creates a new application project from a project under source control.

Create the new project as follows:

1. Depending on the source control system used, it might be necessary, before you create a new project from source control, to make sure that no file from the project is checked out.
2. No project need be open in the application, but can be.
3. Select the command Project | Source Control | Open from Source Control.
4. The source control system that is currently set will pop up its verification and connection dialogs. Make the connection to the repository you want, that is, to the bound folder in the repository that corresponds to the local folder.
5. In the dialog that pops up (*screenshot below*), browse for the local folder to which the contents of the bound folder in the repository (that you have just connected to) must be copied. In the screenshot below the bound folder is called `MyProject` and is represented by the `$` sign; the local folder is `C:\M20130326`.



6. Click **OK**. The contents of the bound folder (`MyProject`) will be copied to the local folder `C:\M20130326`., and a dialog pops up asking you to select the project file (`.spp` file) that is to be created as the new project.
7. Select the `.spp` file that will have been copied to the local folder. In our example, this will be `MyProject.spp` located in the `C:\M20130326` folder. A new project named `MyProject` will be created in the application and will be displayed in the Project window. The project's files will be in the folder `C:\M20130326`.

Source control symbols

Files and the project folder display certain symbols, the meanings of which are given below.

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

Enable Source Control

The **Enable Source Control** command allows you to enable or disable source control for an application project. Selecting this option on any file or folder, enables/disables source control for the whole project. After source control is enabled, the check in/out status of the various files are retrieved and displayed in the Project window.

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

Get Latest Version

The **Get Latest Version** command (in the **Project | Source Control** menu) retrieves and places the latest source control version of the selected file(s) in the working directory. The files are retrieved as read-only and are not checked out. This command works like the **Get** command, but does not display the Get dialog.

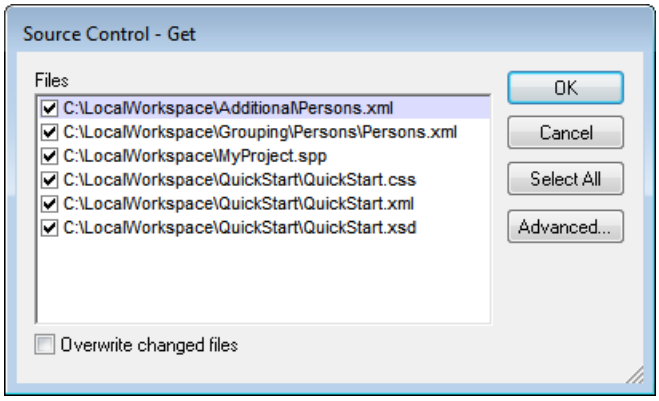
If the selected files are currently checked out, then the action taken will depend on how your source control system handles such a situation. Typically, the source control system will ask whether you wish to replace, merge with, or leave the checked-out file as it is.

Note: This command is recursive when performed on a folder, that is, it affects all files below the current one in the folder hierarchy.

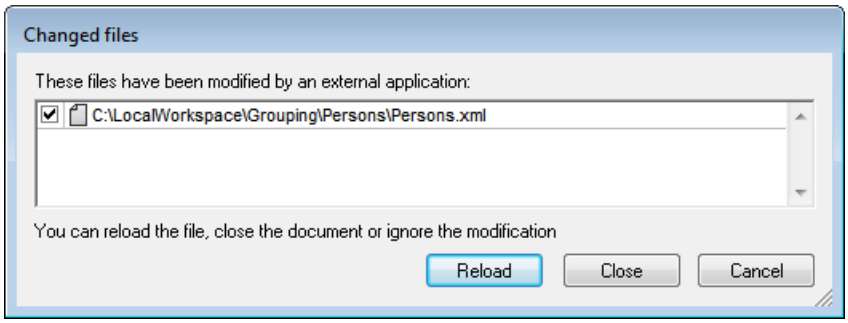
Get, Get Folders

The **Get** command (in the **Project | Source Control** menu) retrieves files from the repository as read-only files. (To be able to edit a file, you must check it out.) The Get dialog lists the files in the object (project or folder) on which the **Get** command was executed (*see screenshot below*). You can select the files to retrieve by checking them.

Note: The **Get Folders** command allows you to select individual sub-folders in the repository if this is allowed by your source control system, .

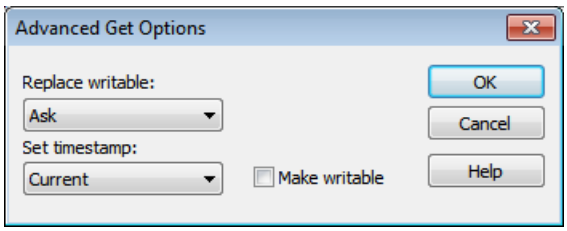


You can choose to overwrite changed checked-out files by checking this option at the bottom of the Get dialog. On clicking **OK**, the files will be overwritten. If any of the overwritten files is currently open, a dialog pops up (*screenshot below*) asking whether you wish to reload the file/s (**Reload** button), close the file/s (**Close**), or retain the current view of the file (**Cancel**).



Advanced Get Options

The Advanced Get Options dialog (*screenshot below*) is accessed via the **Advanced** button in the Get dialog (*see first screenshot in this section*).



Here you can set options for (i) replacing writable files that are checked out, (ii) the timestamp, and (iii) whether the read-only property of the retrieved file should be changed so that it will be writable.

Check Out, Check In

After a project file has been placed under source control, it can be checked out or checked in by selecting the file (in the Project window) and clicking the respective command in the **Project | Source Control** menu: **Check Out** and **Check In**.

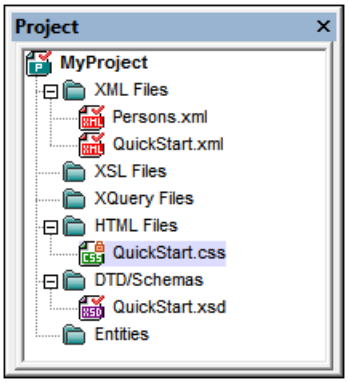
When a file is checked out, a copy from the repository is placed in the local folder. A file that is

checked out can be edited. If a file that is under source control is not checked out, it cannot be edited. After a file has been edited, the changes can be saved to the repository by checking in the file. Even if the file is not saved, checking it in will save the changes to the repository. Whether a file is checked out or not is indicated with a tick or lock symbol in its icon.

Files and the project folder display certain symbols, the meanings of which are given below.

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

Selecting the project or a folder within the project, selects all files in the selected object. To select multiple objects (files and folders), press the Ctrl key while clicking the objects. The screenshot below shows a project that has been checked out. The file `QuickStart.css` has subsequently been checked in.



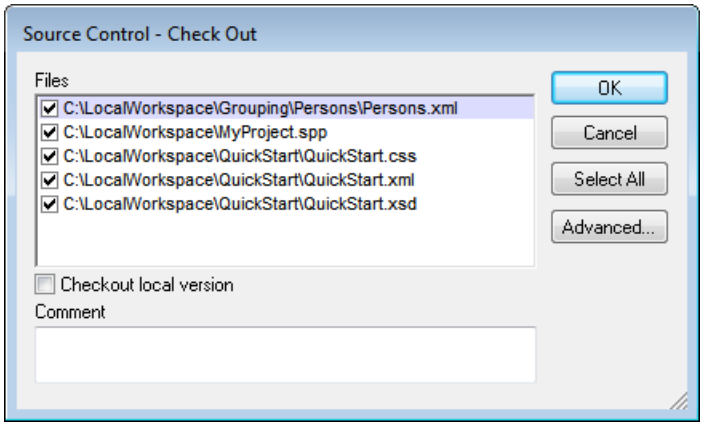
Saving and rejecting editing changes

Note that, when checking in a file, you can choose to leave the file checked out. What this does is save editing changes to the repository while continuing to keep the file checked out, which is useful if you wish to periodically save editing changes to the repository and then continue editing.

If you have checked out a file and made editing changes, and then wish to reject these changes, you can revert to the document version saved in the repository by selecting the command **Project | Source Control | Undo Check Out**.

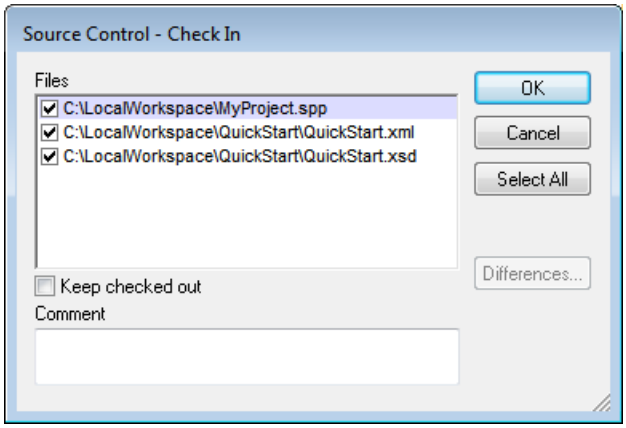
Checking out

The Check Out dialog (*screenshot below*) allows you: (i) to select the files to check out, and (ii) to select whether the repository version or the local version should be checked out.



Checking in

The Check In dialog (*screenshot below*) allows you: (i) to select the files to check in, and (ii) if you wish, to keep the file checked out.



Note: In both dialogs (Check Out and Check In), multiple files appear if the selected object (project or project folder/s) contain multiple files.

Undo Check Out

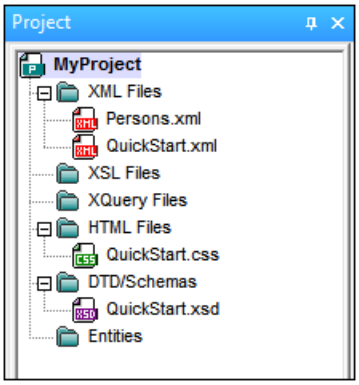
If you have checked out a file and made editing changes, and then wish to reject these changes, you can revert to the document version saved in the repository by selecting the command **Project | Source Control | Undo Check Out**.

Files and the project folder display certain symbols, the meanings of which are given below.

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

Add to Source Control

After a project has been added to source control, you can add files either singly or in groups to source control. Select the file in the Project window and then click the command **Project | Source Control | Add to Source Control**. To select multiple files, keep the **Ctrl** key pressed while clicking on the files you wish to add. Running the command on a (green) project folder (*see screenshot below*) adds all files in the folder and its sub-folders to source control.



When files are added to source control, the local folder hierarchy is replicated in the repository (not the project folder hierarchy). So, if a file is in a sub-folder X levels deep in the local folder, then the file's parent folder and all other ancestor folders are automatically created in the repository.

When the first file from a project is added to source control, the correct bindings are created in the repository and the project file (`.spp` file) is added automatically. For more details, see the section Add to Source Control.

Source control symbols

Files and the project folder display certain symbols, the meanings of which are given below.

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

Remove from Source Control

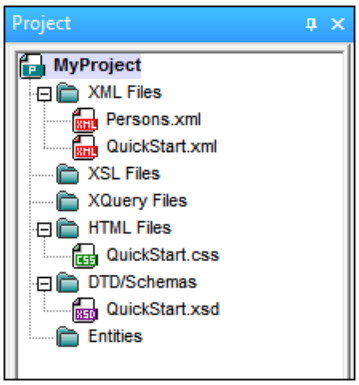
To remove a file from source control, select the file and click the command **Project | Source Control | Remove from Source Control**. You can also remove: (i) files in a project folder by executing the command on the folder, (ii) multiple files that you select while keeping the **Ctrl** key pressed, and (iii) the entire project by executing the command on the project.

Share from Source Control

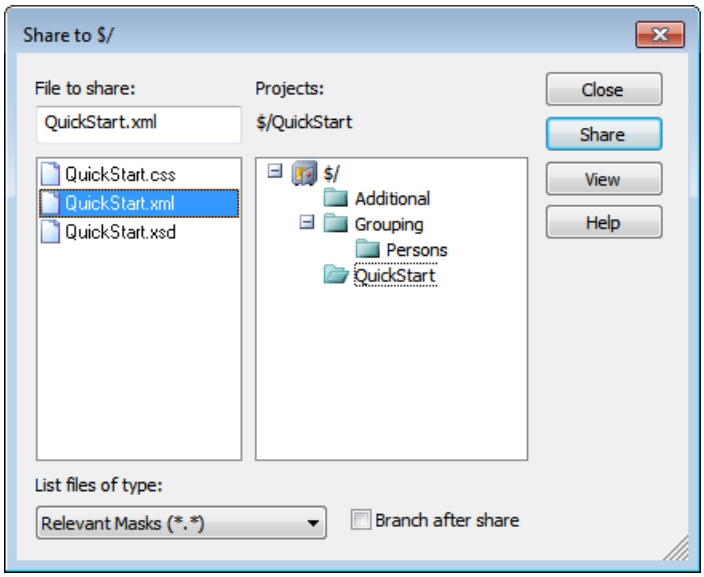
The **Share from Source Control** command is supported when the source control system being used supports shares. You can share a file, so that it is available at multiple local locations. A

change made to one of these local files will be reflected in all the other "shared" versions.

In the application's Project window first select the project (*highlighted in the screenshot below*). Then click the **Share from Source Control**.

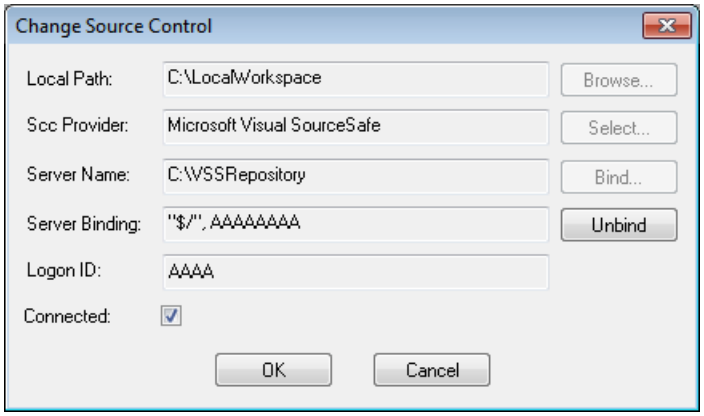


The Share To [Folder] dialog (*screenshot below*) pops up.



To select the files to share, first choose, in the project tree in the right.hand pane, the folder in which the files are. The files in the chosen folder are displayed in the left hand pane. Select the file you wish to share (multiple files by pressing the **Ctrl** key and clicking the files you want to share). The selected file/s will be displayed in the *Files to Share* text box (*at top left*). Click **Share** and then **Close** to copy the selected file/s to the local share folder.

The share folder is noted in the name of the Share to [Folder] dialog. In the screenshot above it is the local folder (since the `$` sign is the folder in the repository to which the local folder is bound). You can see and set the share folder in the Change Source Control dialog (*screenshot below*, **Change Source Control**) by changing the local path and server binding.

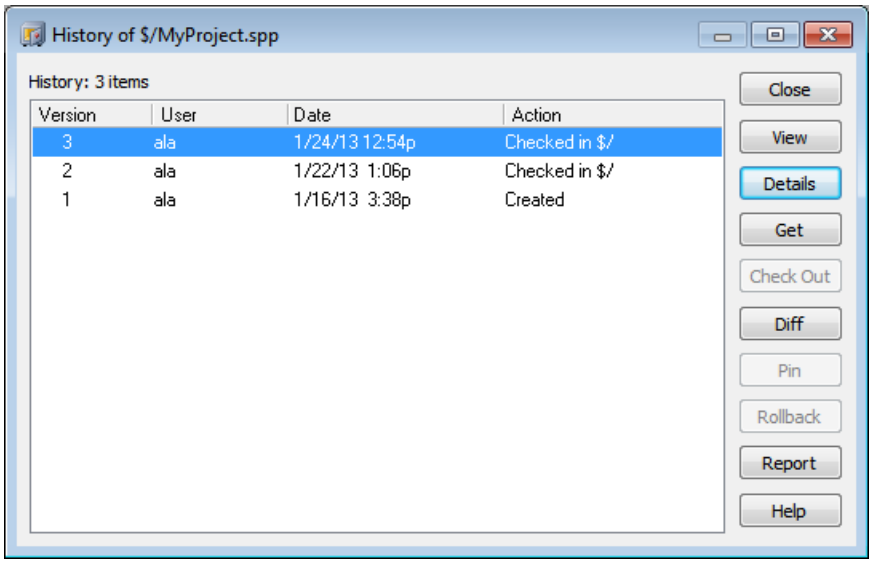


For more details about sharing using your source control system, see the source control system's user documentation.

Show History

The **Show History** command activates the Show History feature of the active source control system. It displays the history of the file selected in the Project window. Select the project title to display the history of the project file (`.spp` file). You can view information about previous versions of a file and differences, as well as retrieve previous versions of the file.

The screenshot below shows the History dialog of the Visual SourceSafe source control system. It lists the various versions of the `MyProject.spp` file.



This History dialog provides various ways of comparing and getting specific versions of the file in question. Double-clicking an entry in the list opens the History Details dialog box for that file. The buttons in the History dialog provide the following functionality:

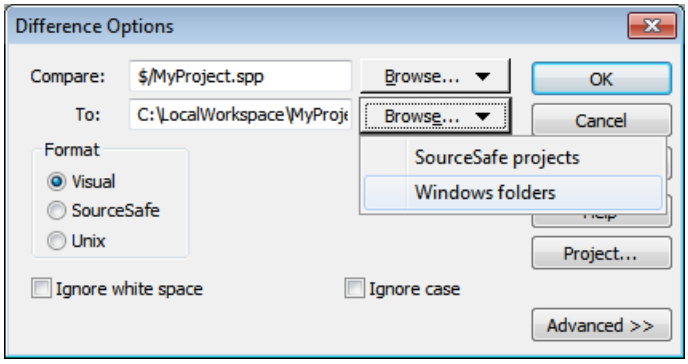
- *Close:* Closes this dialog box.
- *View:* Opens a dialog box in which you can select the type of file viewer.
- *Details:* Opens a dialog box in which you can see the properties of the currently active file.

- *Get:* Retrieves a previous file version and places it in the working directory.
- *Check Out:* Allows you to check out a previous version of the file.
- *Diff:* Opens the Difference options dialog box for differencing options between two file versions. Use **Ctrl+Click** to mark two file versions in this window, then click Diff to view the differences between them.
- *Pin:* Pins or unpins a version of the file, allowing you to define the specific file version to use when differencing two files.
- *Rollback:* Rolls back to the selected version of the file.
- *Report:* Generates a history report that you can send to a printer, file, or clipboard.
- *Help:* Opens the online help of the source control provider plugin.

Show Differences

The **Show Differences** command is enabled when a file in the Project window is selected. To select the project file (`.spp` file), select the project title in the Project window. The **Show Differences** command starts the source control system's differencing tool so that differences between files can be directly checked from your Altova application.

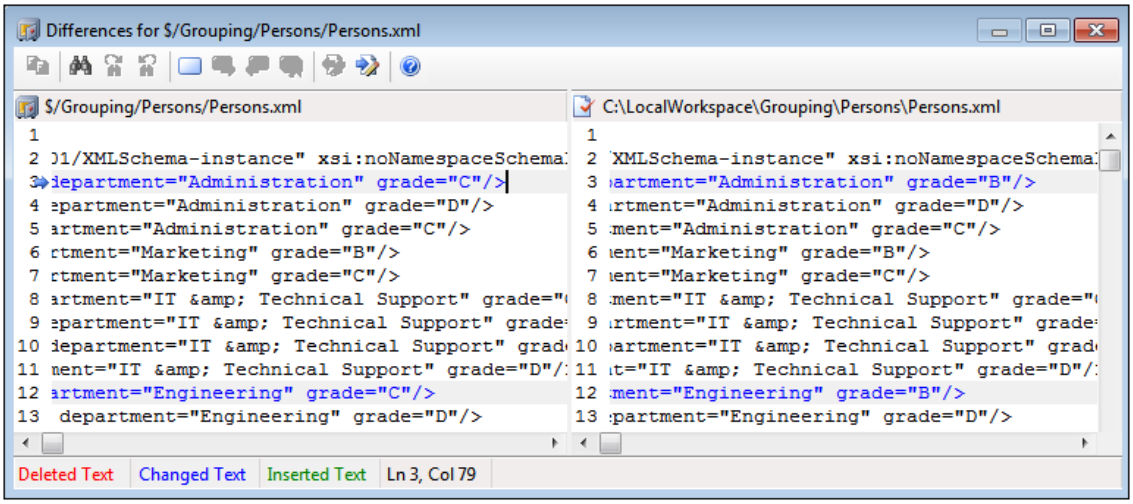
The screenshot below shows the differencing tool of the Visual SourceSafe source control system.



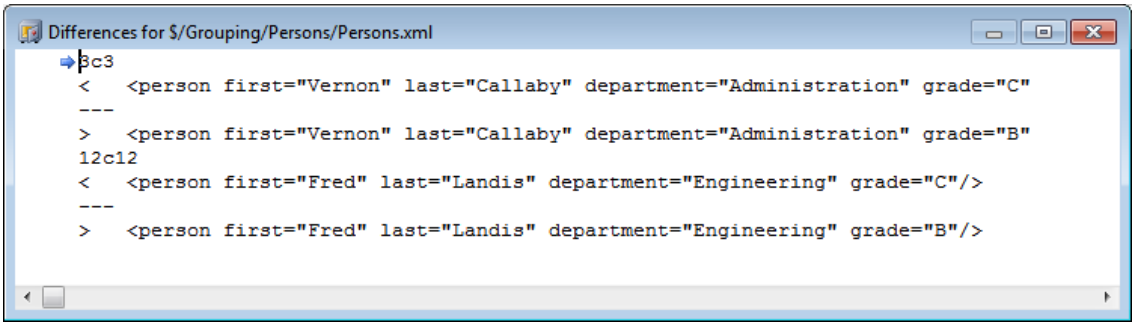
The repository and local versions are shown by default in the *Compare* and *To* text fields respectively. You can browse for other files as follows:

1. From the **Browse** button dropdown list, select SourceSafe projects (for browsing repository files) or Windows folders (for browsing local folders).
2. Browse for the files you want and select them.

Select the options you want and click **OK** to run the check. The differencing results are displayed in a separate window. The screenshots below show the results of a check in two formats.



The screenshot above shows the Visual SourceSafe differencing result in Visual format (*see Options dialog above*), while the screenshot below shows the result in Unix format. In both, there are two differences, each of which is a change of the grade from `C` to `B`.

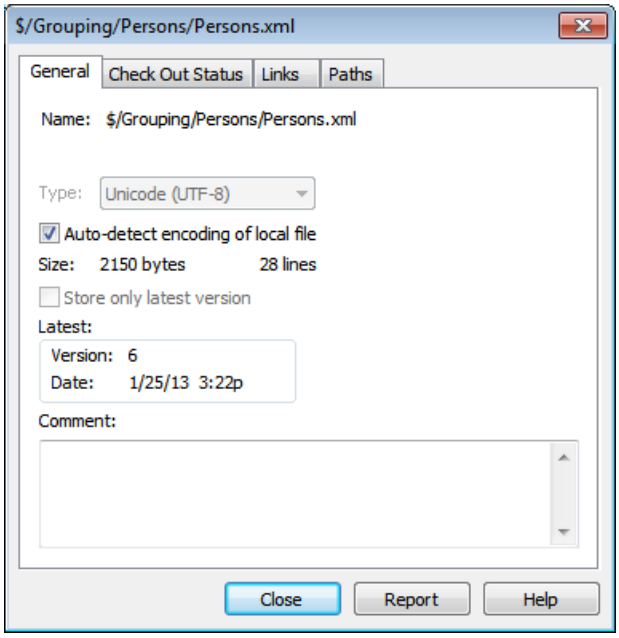


For a detailed description of how your source control system handles differencing, see the product's user documentation.

Show Properties

The **Show Properties** command displays the properties of the currently selected file (*screenshot below*). What properties are displayed depends on the source control system you are using. The screenshot below shows properties when Visual SourceSafe is the active source control system.

Note that this command is enabled only for single files.



For details, see the source control system's user documentation.

Refresh Status

The **Refresh Status** command refreshes the status of all project files independent of their current status.

Source Control Manager

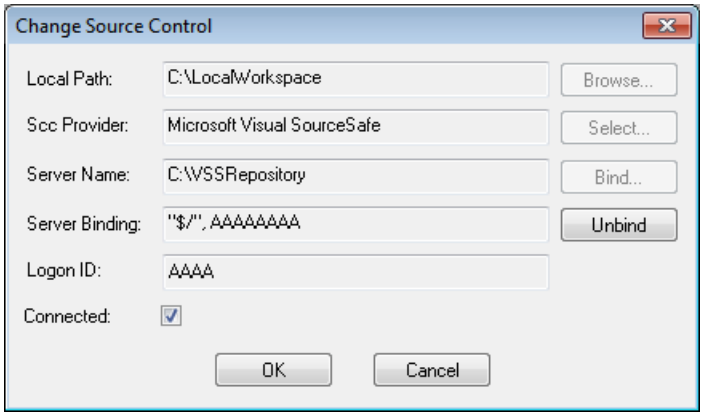
The **Source Control Manager** command starts your source control software with its native user interface.

Change Source Control

The current binding is what the active application project will use to connect to the source control database, so the current binding must be correct. By this is meant that the application project file (`.spp` file) must be in the local path folder and the bound folder on the repository must be the database where this project's files are stored. Typically the bound folder and its sub-structure will correspond with the local workspace folder and its sub-structure.

In the Change Source Control dialog (*screenshot below*), you can change the source control system (*SCC Provider*), the local folder (*Local Path*), and the repository binding (*Server Name* and *Server Binding*).

Only after unbinding the current binding can the settings be changed. Unbind the current binding with the **Unbind** button. All the settings are now editable.



Change source control settings as follows:

1. Use the **Browse** button to browse for the local folder and the **Select** button to select from among the installed source control systems.
2. After doing this you can bind the local folder to a repository database. Click the **Bind** button to do this. This pops up the connection dialog of your source control system.
3. If you have entered a *Logon ID*, this will be passed to the source control system; otherwise you might have to enter your logon details in the connection dialog.
4. Select the database in the repository that you wish to bind to this local folder. This setting might be spread over more than one dialog.
5. After the setting has been created, click **OK** in the Change Source Control dialog.

26.3.7 Add Files to Project

The **Project** | **Add Files to Project** command adds files to the current project. Use this command to add files to any folder in your project. You can either select a single file or any group of files (using **Ctrl+ click**) in the Open dialog box. If you are adding files to the project, they will be distributed among the respective folders based on the File Type Extensions defined in the Project Properties dialog box.

26.3.8 Add Global Resource to Project

The **Project** | **Add Global Resource to Project** command pops up the Choose Global Resource dialog, in which you can select a global resource of file or folder type to add to the project. If a file-type global resource is selected, then the file is added to the appropriate folder based on the File Type Extensions defined in the Project Properties dialog box. If a folder-type global resource is selected, that folder will be opened in a file-open dialog and you will be prompted to select a file; the selected file is added to the appropriate folder based on the File Type Extensions defined in the Project Properties dialog box. For a description of global resources, see the Global Resources section in this documentation.

26.3.9 Add URL to Project

The **Project** | **Add URL to Project** command adds a URL to the current project. URLs in a

project cause the target object of the URL to be included in the project. Whenever a batch operation is performed on a URL or on a folder that contains a URL object, XMLSpy retrieves the document from the URL, and performs the requested operation.

26.3.10 Add Active File to Project

The **Project** | **Add Active File to Project** command adds the active file to the current project. If you have just opened a file from your hard disk or through an URL, you can add the file to the current project using this command.

26.3.11 Add Active And Related Files to Project

The **Project** | **Add Active and Related Files to Project** command adds the currently active XML document and all related files to the project. When working on an XML document that is based on a DTD or Schema, this command adds not only the XML document but also all related files (for example, the DTD and all external parsed entities to which the DTD refers) to the current project.

Please note: Files referenced by processing instructions (such as XSLT files) are not considered to be related files.

26.3.12 Add Project Folder to Project

The **Project** | **Add Project Folder to Project** command adds a new folder to the current project. Use this command to add a new folder to the current project or a sub-folder to a project folder. You can also access this command from the context-menu when you right-click on a folder in the project window.

Note: A project folder can be dragged and dropped into another project folder or to any other location in the project. Also, a folder can be dragged from Windows (File) Explorer and dropped into any project folder.

Note: Project folders are green, while external folders are yellow.

26.3.13 Add External Folder to Project

The **Project** | **Add External Folder to Project** command adds a new external folder to the current project. Use this command to add a local or network folder to the current project. You can also access this command from the context-menu when you right-click a folder in the project window.

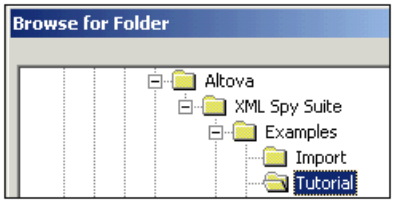
Note: External folders are yellow, while project folders are green.

Note: Files contained in external folders cannot be placed under source control.

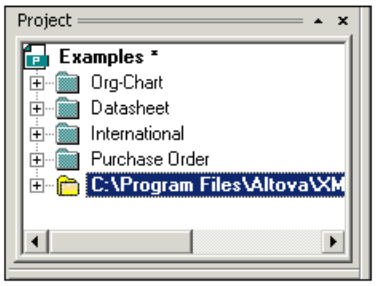
Adding external folders to projects

To add an external folder to the project:

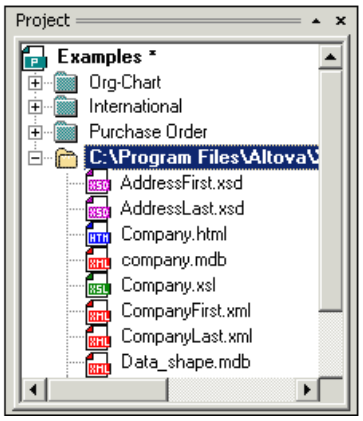
1. Select the menu option **Project | Add External Folder to Project**.
2. Select the folder you want to include from the Browse for Folder dialog box, and click **OK** to confirm.



The selected folder now appears in the project window.

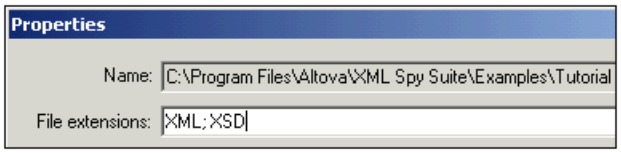


3. Click the plus icon to view the folder contents.

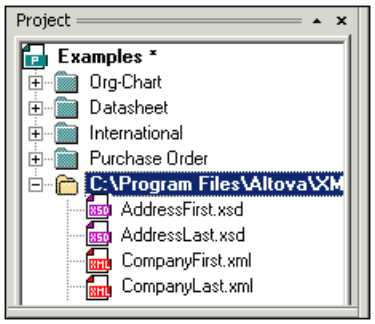


Filtering contents of folders
To filter the contents of the folder:

1. Right-click the local folder, and select the popup menu option **Properties**. This opens the Properties dialog box.



2. Click in the **File extensions** field and enter the file extensions of the file types you want to see. You can separate each file type with a **semicolon** to define multiple types (XML and Schema XSDs in this example).
3. Click **OK** to confirm.



The Project window now only shows the XML and XSD files of the tutorial folder.

Validating external folders
To validate and check an external folder for well-formedness:

1. Select the file types you want to see or check from the external folder,
2. Click the folder and click the **Check well-formedness** or **Validate** icon (hotkeys **F7** or **F8**). All the files visible under the folder are checked. If a file is malformed or invalid, then this file is opened in the Main Window, allowing you to edit it.
3. Correct the error and run the validation process once more to recheck.

Updating a project folder
You might add or delete files in the local or network directory at any time. To update the folder view, right-click the external folder, and select the popup menu option **Refresh external folder**.

Deleting external folders and files in them
Select an external folder and press the **Delete** key to delete the folder from the Project window. Alternatively, right-click the external folder and select the **Delete** command. Each of these actions only deletes the external folder from the Project window. The external folder is not deleted from the hard disk or network.

To delete a file in an external folder, you have to delete it physically from the hard disk or network. To see the change in the project, refresh the external folder contents (right-click the external folder and select **Refresh**).

Note: An external folder can be dragged and dropped into a project folder or to any other location in the project (but not into another external folder). Also, an external folder can be dragged from Windows (File) Explorer and dropped into any location in the project window except into another external folder.

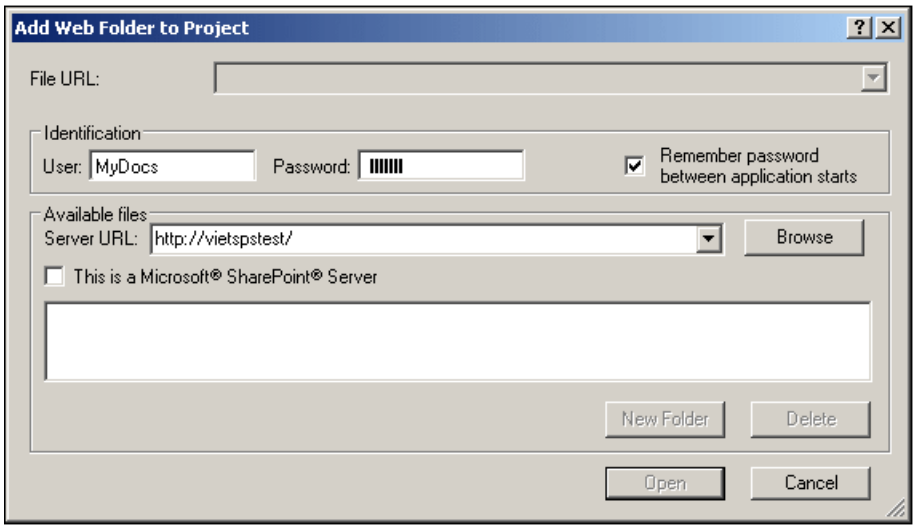
26.3.14 Add External Web Folder to Project

This command adds a new external web folder to the current project. You can also access this command from the context-menu when you right-click a folder in the project window. Note that files contained in external folders cannot be placed under source control.

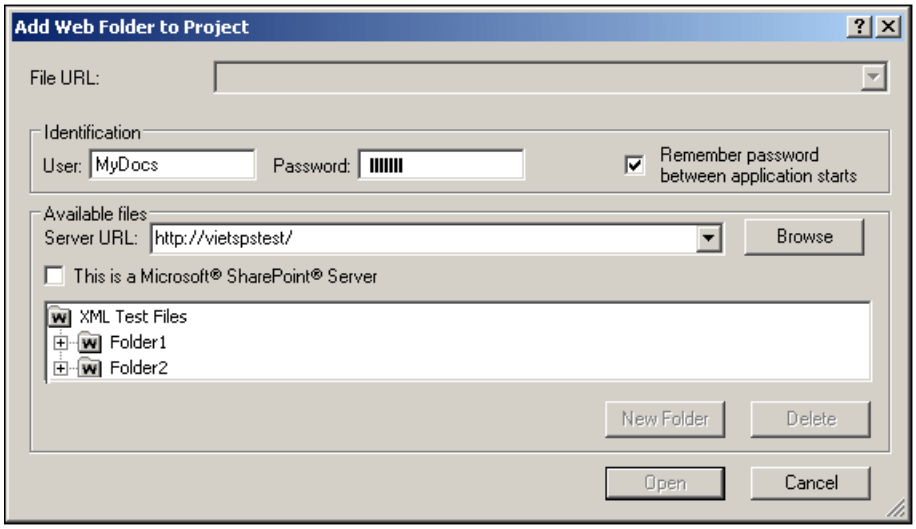
Adding an external web folder to the project

To add an external web folder to the project, do the following:

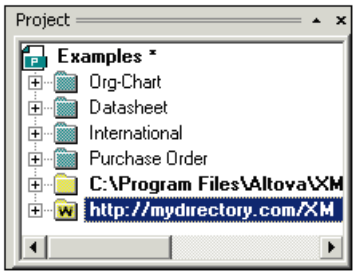
1. Select the menu option **Project | Add External Web Folder to Project**. This opens the Add Web Folder to Project dialog box (*screenshot below*).



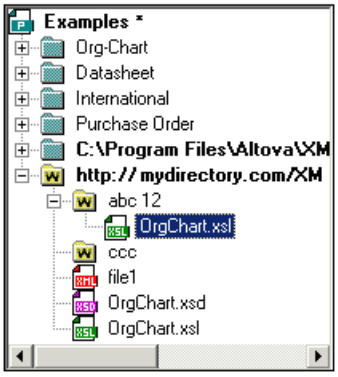
2. Click in the Server URL field and enter the URL of the server URL. If the server is a Microsoft® SharePoint® Server, check this option. See the *Folders on a Microsoft® SharePoint® Server* section below for further information about working with files on this type of server.
3. If the server is password-protected, enter your User ID and password in the *User* and *Password* fields.
4. Click **Browse** to connect to the server and view the available folders.



5. Click the folder you want to add to the project view. The **Open** button only becomes active once you do this. The URL of the folder now appears in the File URL field.
6. Click **Open** to add the folder to the project.



7. Click the plus icon to view the folder contents.



Filtering folder contents

To filter the contents of a folder, right-click the folder and select **Properties** from the context menu. In the Properties dialog that pops up, click in the *File Extensions* field and enter the file extensions of the file types you want to see (for example, XML and XSD files). Separate each file type with a semicolon (for example: `xml; xsd; sps`). The Project window will now show that folder only with files having the specified extension.

Validating and checking a folder for well-formedness

To check the files in a folder for well-formedness or to validate them, select the folder and then click the **Check well-formedness** or **Validate** icon (hotkeys **F7** or **F8**, respectively). All the files that are visible in the folder are checked. If a file is malformed or invalid, then this file is opened in the main window, allowing you to edit it. Correct the error and restart the process to recheck the rest of the folder. Note that you can select discontinuous files in the folder by holding **Ctrl** and clicking the files singly. Only these files are then checked when you press **F7** or **F8**.

Updating the contents of the project folder

Files may be added or deleted from the web folder at any time. To update the folder view, right-click the external folder and select the context menu option **Refresh**.

Deleting folders and files

Since it is the Web folder that has been added to the project, it is only the Web folder (and not files within it) that can be deleted from the project. You can delete a Web folder from a project, by either (i) right-clicking the folder and selecting **Delete**, or (ii) selecting the folder and pressing the **Delete** key. This only deletes the folder from the Project view; it does not delete anything on the web server.

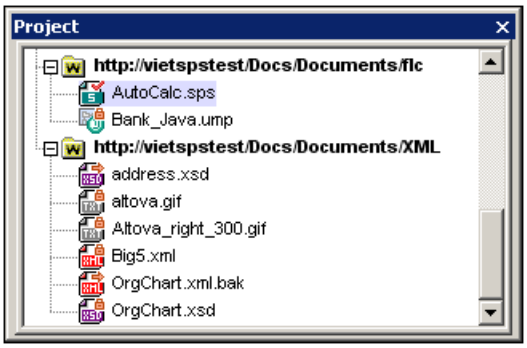
Note: Right-clicking a single file and pressing the **Delete** key does not delete a file from the Project window. You have to delete it physically on the server and then refresh the contents of the external folder.

Folders on a Microsoft® SharePoint® Server

When a folder on a Microsoft® SharePoint® Server has been added to a project, files in the folder can be checked out and checked in via commands in the context menu of the file listing in

the Project window (*see screenshot below*). To access these commands, right-click the file you wish to work with and select the command you want (**Check Out**, **Check In**, **Undo Check Out**).

The User ID and password can be saved in the properties of individual folders in the project, thereby enabling you to skip the verification process each time the server is accessed.



In the Project window (*screenshot below*), file icons have symbols that indicate the check-in/ check-out status of files. The various file icons are shown below:

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

The following points should be noted:

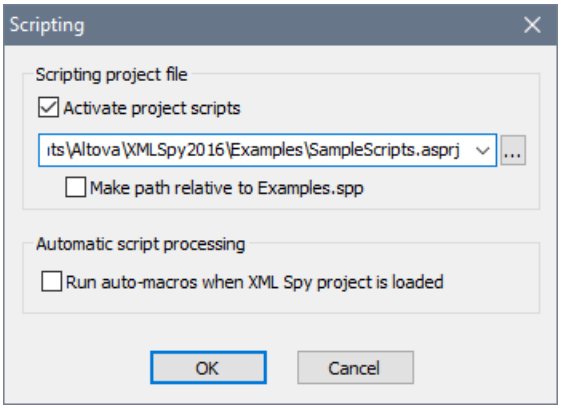
- After you check out a file, you can edit it in your Altova application and save it using **File | Save (Ctrl+S)**.
- You can check-in the edited file via the context menu in the Project window (*see screenshot above*), or via the context menu that pops up when you right-click the file tab in the Main Window of your application (*screenshot below*).

- When a file is checked out by another user, it is not available for check out.
- When a file is checked out locally by you, you can undo the check-out with the Undo Check-Out command in the context menu. This has the effect of returning the file unchanged to the server.
- If you check out a file in one Altova application, you cannot check it out in another Altova application. The file is considered to be already checked out to you. The available commands at this point in any Altova application supporting Microsoft® SharePoint® Server will be: **Check In** and **Undo Check Out**.

26.3.15 Script Settings

A scripting project is assigned to an XMLSpy project as follows:

1. In the XMLSpy GUI, open the required application project.
2. Select the menu command **Project | Script Settings**. The Scripting dialog (*screenshot below*) opens.



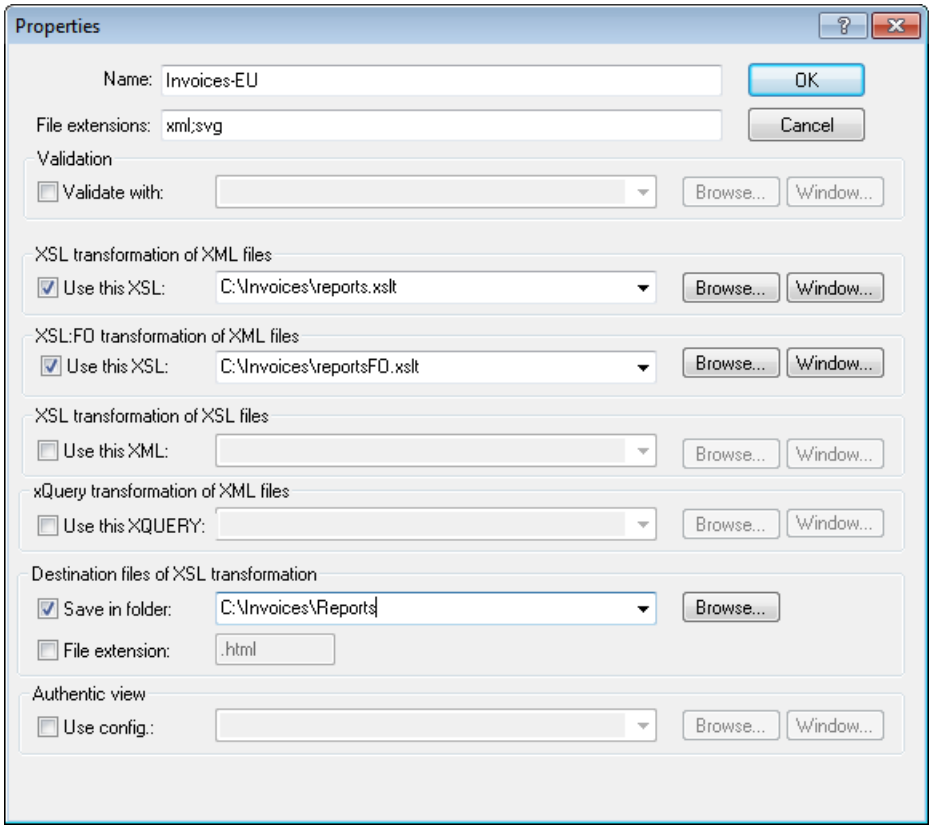
3. Check the *Activate Project Scripts* check box and select the required scripting project (`asprj` file). If you wish to run Auto-Macros when the XMLSpy project is loaded, check the *Run Auto-Macros* check box.
4. Click **OK** to finish.

Note: To deactivate (that is, unassign) the scripting project of an XMLSpy project, uncheck the *Activate Project Scripts* check box.

26.3.16 Properties

The **Project** | **Project Properties** command opens the Properties dialog (*screenshot below*) of the active project. If you right-click a folder in the Project window (as opposed to the project folder itself) and select **Properties**, the Properties dialog of that folder is opened. The dialog settings are described below.

Note: If your project file is under source control, a prompt appears asking if you want to check out the project (`.spp`) file. Click **OK** if you want to edit settings and be able to save them.



Settings

File extensions
The *File Extensions* setting is enabled for individual folders, and not for the project folder. When a file is added to a project, it will be added to the folder on which its file extension has been defined. For example, say a file named `MyReport.xml` is added to the project. If `.xml` file extensions have been set on the `Invoices-EU` folder (as shown in the screenshot above), then `MyReport.xml` will be added to the `Invoices-EU` folder. If there is more than one folder to which you wish to add XML files, then you should add individual XML files directly to the folder (instead of to the project).

User ID and password for external folders
On external folders (including external Web folders), you can save the user ID and password that might be required for accessing the server.

Validation
The DTD, XML Schema, or JSON schema that should be used to validate the files in the current folder (or entire project if the properties are those of the project).

XSL transformation of XML files
The XSLT stylesheet to be used for XSLT transformation of XML files in the folder.

XSL-FO transformation of XML files
The XSLT stylesheet to transform XML files in the folder to XSL-FO.

XSL transformation of XSL files
The XML file to be used for XSLT transformation with XSLT files in the folder.

XQuery transformation of XML files
The XQuery file to be used for XQuery executions of XML files in the folder.

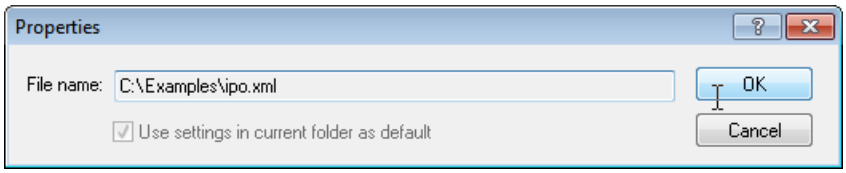
Destination files of XSL transformation
The destination directory of XSLT transformations, and, optionally, the file extension of the result document.

Authentic View
The *Use config* specifies the StyleVision Power Stylesheet (SPS file) to use for the Authentic View display of XML files in the folder. Note that the XML file must be valid against the same schema used for the SPS.

Notes about project properties

Notes about precedence
Note the following:

- When validations or XSLT/XQuery transformations are carried out via project folder context menus, then the validation or transformation files specified in this dialog take precedence over any assignment in the XML file. Also, settings specified for individual project folders take precedence over settings specified for ancestor folders.
- If one file is present in multiple folders of the project and has been assigned different validation or transformation files in the different folders, then you can set which assignment to use when the file is processed outside the project. Specify this as follows: Locate the file in the project folder whose assignment/s you wish to use. Right-click the file in that project folder, and select **Properties**. In the dialog that appears (*screenshot below*), select *Use settings in current folder as default*. (The current folder is the project folder in which the file is located.) If the option is disabled, it means that the settings of the current folder are already selected as the default settings to use. If you select a file instance that is in a project folder that is not the default, then the option is enabled, and you can switch the default settings to be this folder's settings. Note that, if the file has a local assignment (that is, an assignment within the file itself), then the local assignment will be used, and the default folder settings will be ignored.



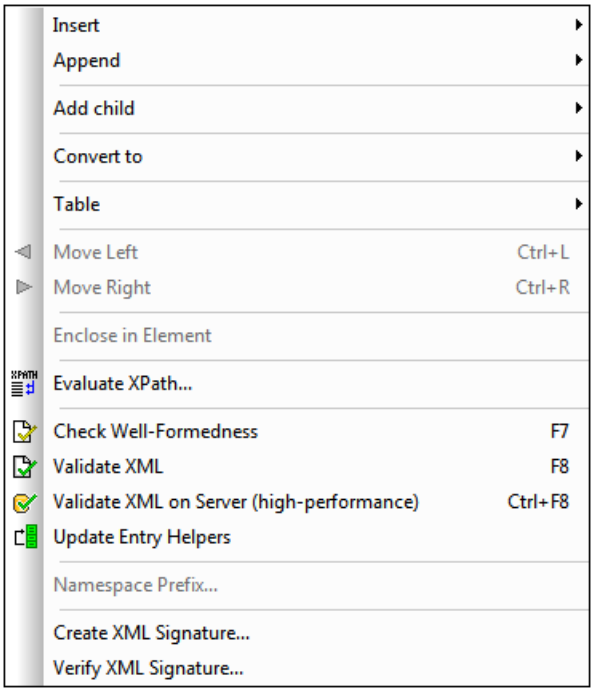
26.3.17 Most Recently Used Projects

This command displays the file name and path for the nine most recently used projects, allowing quick access to these files.

Also note, that XMLSpy can automatically open the last project that you used, whenever you start XMLSpy. (**Tools | Options | File** tab, Project | Open last project on program start).

26.4 XML Menu

The **XML** menu contains commands commonly used when working with XML documents. You will find commands to insert or append elements, modify the element hierarchy, set a namespace prefix, as well as to evaluate XPaths in the context of individual XML documents.

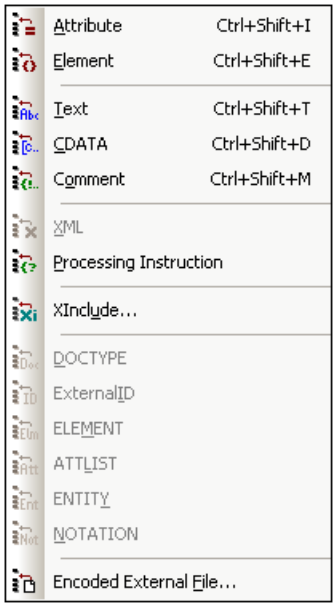


Among the most frequently used XML tasks are checks for the well-formedness of documents and validity of XML documents. Commands for these tasks are in this menu.

26.4.1 Insert

The **XML** | **Insert** command, though enabled in all views, can be used in Grid View only. It has a submenu (*see screenshot*) with which you can insert:

- The XML declaration and node types (Attribute, Element, Text, CDATA, Comment, Processing Instruction) in XML documents;
- DOCTYPE declarations and external DTD declarations in XML documents;
- DTD declarations (ELEMENT, ATTLIST, ENTITY, and NOTATION) in DTD documents and internal DTD declarations of XML documents.



Insert Attribute

Ctrl+Shift+I

The **XML** | **Insert | Attribute** command is available in Grid View only, and inserts a new attribute before the selected item. An inserted attribute may appear a few lines before the current item in Grid View. This is because attributes immediately follow their parent element in Grid View and precede all child elements of that parent element.

Insert Element

Ctrl+Shift+E

The **XML** | **Insert | Element** command is available in Grid View only, and inserts a new element before the selected item. If the current selection is an attribute, the new element is before the first child element of the attribute's parent element.

Insert Text

Ctrl+Shift+T

The **XML** | **Insert | Text** command is available in Grid View only, and inserts a new text row before the selected item. If the current selection is an attribute, the text row is inserted after the attribute and before the first child element of the attribute's parent element.

Insert CDATA

Ctrl+Shift+D

The **XML** | **Insert | Cdata** command is available in Grid View only, and inserts a new CDATA block before the selected item. If the current selection is an attribute, the CDATA block is inserted after the attribute and before the first child element of the attribute's parent element.

Insert Comment

Ctrl+Shift+M

The **XML** | **Insert | Comment** command is available in Grid View only, and inserts a new comment before the selected item. If the current selection is an attribute, the new comment row is inserted after the attribute and before the first child element of the attribute's parent element.

Insert XML

The **XML** | **Insert | XML** command is available in Grid View only, and inserts a row for the XML declaration before the selected item. You must insert the child attributes of the XML declaration and the values of this attribute. An XML declaration must look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
```

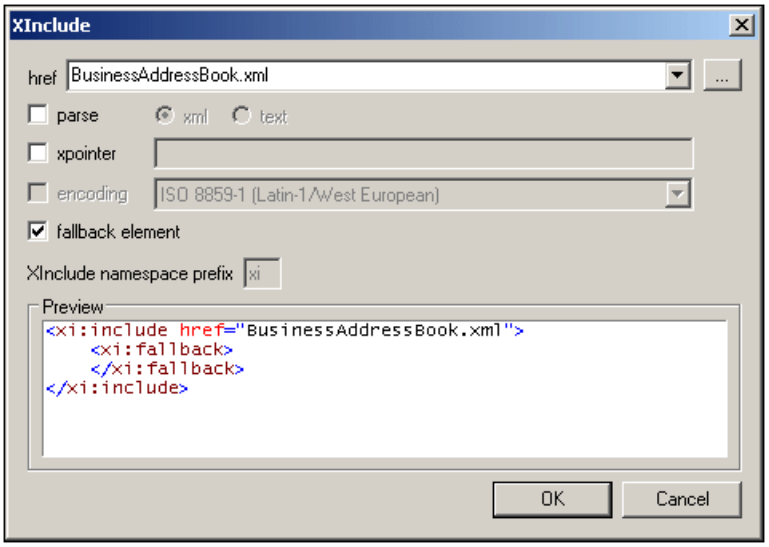
Please note: Since an XML document may only contain one XML declaration at the very top of the file, this command should only be used with the topmost row selected and if an XML declaration does not already exist.

Insert Processing Instruction

The **XML** | **Insert | Processing Instruction** command is available in Grid View only, and inserts a new processing instruction (PI) before the selected item. If the current selection is an attribute, the PI is inserted after the attribute and before the first child element of the attribute's parent element.

Insert XInclude

The **XML** | **Insert | XInclude** command is available in Grid View only, and enables you to insert a new XInclude element before the selected item. If the current selection is an attribute, the XInclude element is inserted after the attribute and before the first child element of the attribute's parent element. Selecting this command pops up the XInclude dialog (*screenshot below*).



The XML file to be included is entered in the `href` text box (alternatively, you can browse for the file by clicking the **Browse (...)** button to the right of the text box). The filename will be entered in the XML document as the value of the `href` attribute. The `parse`, `xpointer`, and `encoding` attributes of the XInclude element (`xi:include`), and the `fallback` child element of `xi:include` can also be inserted via the dialog. Do this by first checking the appropriate check box and then selecting/entering the required values. In the case of the `fallback` element, checking its check box only inserts the empty element. The content of the `fallback` element must be added subsequently in one of the editing views.

The `parse` attribute determines whether the included document is to be parsed as XML or text. (XML is the default value and therefore need not be specified.) The `xpointer` attribute identifies a specific fragment of the document located with the `href` attribute; it is this fragment that will be included. The `encoding` attribute specifies the encoding of the included document so that XMLSpy can transcode this document (or the part of it to be included) into the encoding of the including document. The contents of the `fallback` child element replace the `xi:include` element if the document to be included cannot be located.

Here is an example of an XML document that uses XInclude to include two XML documents:

```
<?xml version="1.0" encoding="UTF-16"?>
<AddressBook xsi:schemaLocation="http://www.altova.com/sv/myaddresses
AddressBook.xsd"
       xmlns="http://www.altova.com/stylevision/tutorials/myaddresses"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="BusinessAddressBook.xml"/>
       <xi:include href="PersonalAddressBook.xml"/>
</AddressBook>
```

When this XML document is parsed, it will replace the two XInclude elements with the files specified in the respective `href` attributes.

xml:base

When the XML validator of XMLSpy reads an XML document and encounters the `include` element in the XInclude namespace (hereafter `xi:include`), it replaces this element (`xi:include`) with the XML document named in the `href` attribute of the `xi:include` element. The document element (root element) of the included XML document (or the element identified by

an XPointer) will be included with an attribute of `xml:base` in order to preserve the base URIs of the included element. If the resulting XML document (containing the included XML document/s or tree fragment/s) must be valid according to a schema, then the document element of the included document (or the top-level element of the tree fragment) must be created with a content model that allows an attribute of `xml:base`. If, according to the schema, the `xml:base` attribute is not allowed on this element, then the resulting document will be invalid. How to define an `xml:base` attribute in an element's content model using XMLSpy's Schema View is described in the xml:Prefixed Attributes section of the Schema View section of the documentation.

XPointers

XMLSpy supports XPointers in XInclude. The relevant W3C recommendations are the XPointer Framework and XPointer element() Scheme recommendations. The use of an XPointer in an XInclude element enables a specific part of the XML document to be included, instead of the entire XML document. XPointers are used within an XInclude element as follows:

```
<xi:include href="PersonalAddressBook.xml" xpointer="element(usa)"/>
<xi:include href="BusinessAddressBook.xml" xpointer="element(/1/1)"/>
<xi:include href="BobsAddressBook.xml" xpointer="element(usa/3/1)"/>
<xi:include href="PatsAddressBook.xml" xpointer="element(usa)
element(/1/1)"/>
```

In the element() scheme of XPointer, an NCName or a child sequence directed by integers may be used.

- In the first `xi:include` element listed above, the `xpointer` attribute uses the element scheme with an NCName of `usa`. According to the XPointer Framework, this NCName identifies the element that has an ID of `usa`.
- In the second `xi:include` listed above, the `xpointer` attribute with a value of `element(/1/1)` identifies, in the first step, the first child element of the document root (which, if the document is well-formed, will be its document (or root) element). In the second step, the first child element of the element located in the previous step is located; in our example, this would be the first child element of the document element.
- The `xpointer` attribute of the third `xi:include` listed above uses a combination of NCName and child sequence. This XPointer locates the first child element of the third child element of the element having an ID of `usa`.
- If you are not sure whether your first XPointer will work, you can back it up with a second one as shown in the fourth `xi:include` listed above: `xpointer="element(usa) element(/1/1)"`. Here, if there is no element with an ID of `usa`, the back-up XPointer specifies that the first child element of the document element is to be selected. Additional backups are also allowed. Individual XPointers may not be separated, or they may be separated by whitespace: for example, `xpointer="element(usa) element(addresses/1) element(/1/1)"`.

Note: The namespace binding context is not used in the element() scheme because the element() scheme does not support qualified names.

Insert DOCTYPE

The **XML** | **Insert** | **DOCTYPE** command is available in the Grid View of an XML file when a top-level node is selected. It appends a DOCTYPE declaration at the top of the XML document. You must enter the name of the DOCTYPE, and this name must be the same as the name of the document element.

After you have entered the name of the DOCTYPE, you can enter the declarations you wish to use in the internal DTD subset.

Please note:

- A DOCTYPE declaration may only appear between the XML declaration and the XML document element.
- You could use the Assign DTD command instead to create a DOCTYPE statement that refers to an external DTD document.

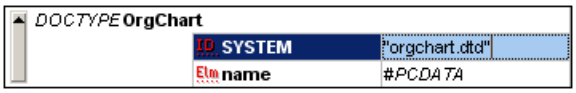
Insert ExternalID

A DOCTYPE declaration in an XML file can contain a reference to an external resource containing DTD declarations. This resource is referenced either through a public or system identifier. For example:

```
<!DOCTYPE doc_element_name PUBLIC "publicID" "systemID">
<!DOCTYPE doc_element_name SYSTEM "systemID">
```

A system identifier is a URI that identifies the external resource. A public identifier is location-independent and can be used to dereference the location of an external resource. For example, in your XMLSpy installation, URIs for popular DTDs and XML Schemas are listed in the catalog files named `catalog.xml` in the various schema folders in `C:\Program Files\Altova\Common2017\Schemas\`. A public identifier in an XML document can be used to dereference a DTD listed in these catalog files.

The **XML** | **Insert** | **ExternalID** command is available when a "child" item of the DOCTYPE declaration in an XML file is selected in Grid View. This command inserts a Grid View row for an external identifier (`PUBLIC` or `SYSTEM`). You must enter the type of identifier and its value.



The Text View corresponding to the screenshot of the Grid View shown above looks something like this:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

Please note: A row for External-ID can be added as a child when the DOCTYPE item is selected, or it can be inserted or appended when one of the child items of the DOCTYPE item is selected, for example, the ELEMENT declaration `name` in the example above.

Insert ELEMENT

The **XML** | **Insert** | **ELEMENT** command is available in Grid View only, for DTD documents or when an item in the DOCTYPE declaration of an XML document is selected. It inserts an

ELEMENT declaration before the selected declaration.

Insert ATTLIST

The **XML** | **Insert** | **ATTLIST** command is available in Grid View only, for DTD documents or when an item in the DOCTYPE declaration of an XML document is selected. It inserts an ATTLIST declaration before the selected declaration.

Insert ENTITY

The **XML** | **Insert** | **ENTITY** command is available in Grid View only, for DTD documents or when an item in the DOCTYPE declaration of an XML document is selected. It inserts an ENTITY declaration before the selected declaration.

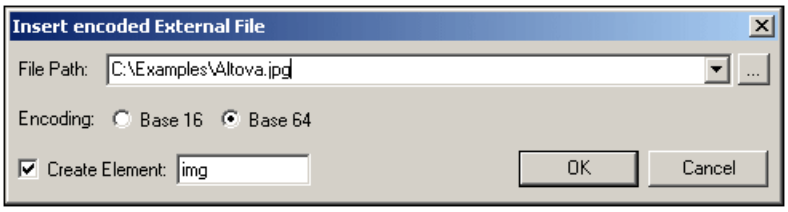
Insert NOTATION

The **XML** | **Insert** | **NOTATION** command is available in Grid View only, for DTD documents or when an item in the DOCTYPE declaration of an XML document is selected. It inserts a NOTATION declaration before the selected declaration.

Insert Encoded External File

The **XML** | **Insert** | **Encoded External File** command is available in Grid View only. It inserts a binary encoded file, such as an image file, as encoded characters. The encoded external file is inserted before the Grid View selection.

On clicking the command, the Insert Encoded External File dialog (*screenshot below*) pops up. In it you enter the path to the file, select the encoding you want, and specify whether the encoded file is to be inserted in an element or not.

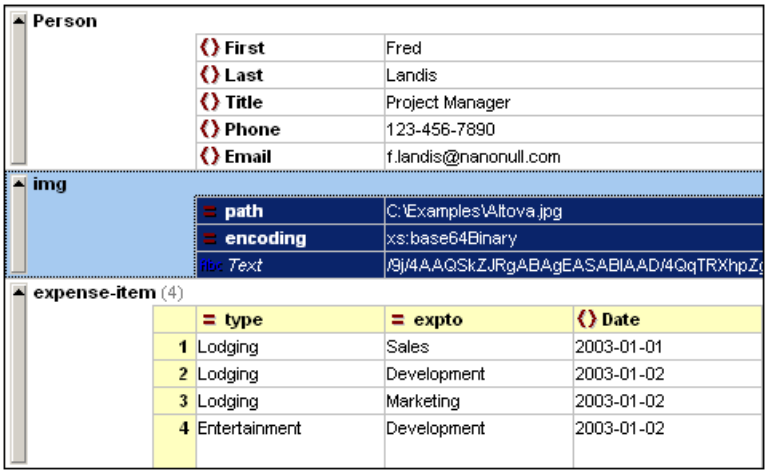


You can browse for or enter the name of the external file to be encoded and embedded. Either a Base-16 or Base-64 encoding must be specified. If you wish to enclose the encoded text in an element, then check the Create Element check box and specify the name of the desired element in the Create Element text box. If the Create Element check box is not checked, then the encoded text will be inserted directly at the cursor location.

On clicking **OK**, the encoded text of the selected file is inserted at the cursor location, with an

enclosing element if this has been specified.

The encoded file is inserted in Grid View (*the highlighted element in the screenshot below*).



In Text View, the file will be inserted as below.

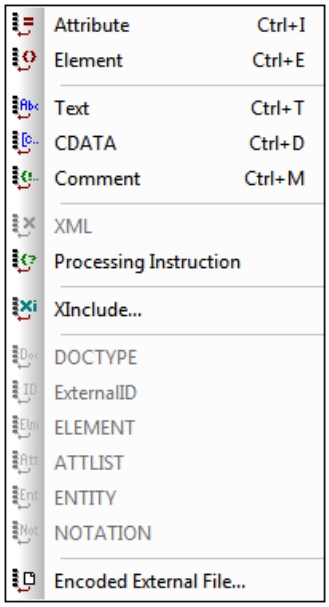
```
<img ext="jpg" encoding="xs:base64Binary">
 iVBORw0KGgoAAAANSUhEUgAAABAAAAAQAQMAAAA1PW0iAAAAB1BMVEUAAAD/
 //+l2Z/dAAAAM0lEQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83NDDeNGe4U
 g9C9zwz3gVLMDA/A6P9/AFGGFyjOXZtQAAAAAElFTkSuQmCC
</img>
```

The listing above shows the encoded text of a JPG image file. An `img` element was created around the encoded text.

26.4.2 Append

The **XML** | **Append** command, though enabled in all views, can be used in Grid View only. It opens a submenu (*see screenshot*) with which you can append:

- The XML declaration and node types (Attribute, Element, Text, CDATA, Comment, Processing Instruction) in XML documents;
- DOCTYPE declarations and external DTD declarations in XML documents
- DTD declarations (ELEMENT, ATTLIST, ENTITY, and NOTATION) in DTD documents and internal DTD declarations of XML documents.



Append Attribute

Ctrl+I

The **XML** | **Append** | **Attribute** command is available in Grid View only, and appends a new attribute.

Append Element

Ctrl+E

The **XML** | **Append** | **Element** command is available in Grid View only, and appends an element node after the last sibling element of the selected element. If an attribute node is selected, then the element node is appended after the last child of the selected attribute's parent element.

Append Text

Ctrl+T

The **XML** | **Append** | **Text** command is available in Grid View only, and appends a text block after the last sibling element of the selected element. If an attribute node is selected, then the text block is appended after the last child of the selected attribute's parent element.

Append CDATA

Ctrl+D

The **XML** | **Append** | **Cdata** command is available in Grid View only, and appends a CDATA node after the last sibling of any selected node other than an attribute node. If an attribute node

is selected, then the CDATA section is appended after the last child of the selected attribute's parent element.

Append Comment

Ctrl+M

The **XML** | **Append** | **Comment** command is available in Grid View only, and appends a comment node after the last sibling of any selected node other than an attribute node. If an attribute node is selected, then the comment node is appended after the last child of the selected attribute's parent element.

Append XML

The **XML** | **Append** | **XML** command inserts a new XML declaration `<?xml version="1.0" encoding="UTF-8"?>` as the first item in a document.

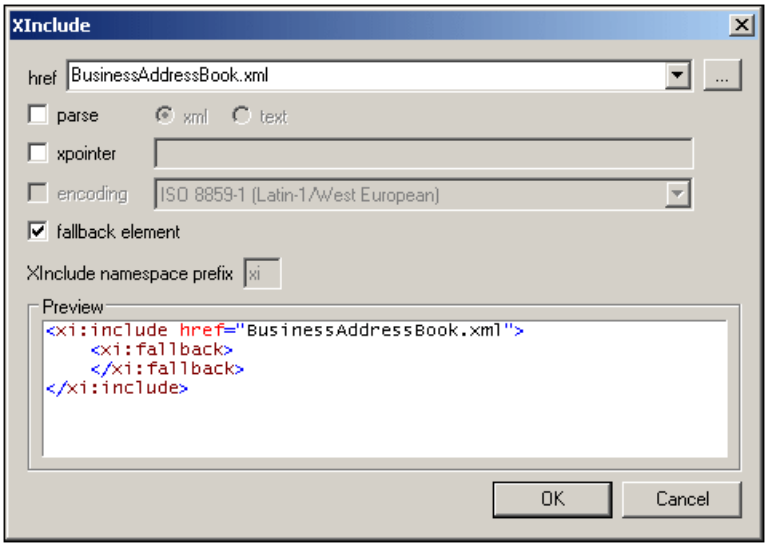
Please note: An XML document may contain only one XML declaration, which must appear at the very top of the file.

Append Processing Instruction

The **XML** | **Append** | **Processing Instruction** command is available in Grid View only, and appends a processing instruction node after the last sibling of any selected node other than an attribute node. If an attribute node is selected, then the processing instruction node is appended after the last child of the selected attribute's parent element.

Append XInclude

The **XML | Append | XInclude** command is available in Grid View only, and enables you to append an XInclude element after the last sibling of any selected node other than an attribute node. If the current selection is an attribute, the XInclude element is appended after the the last child of the selected attribute's parent element. Selecting this command pops up the XInclude dialog (*screenshot below*).



The XML file to be included is entered in the `href` text box (alternatively, you can browse for the file by clicking the **Browse (...)** button to the right of the text box). The filename will be entered in the XML document as the value of the `href` attribute. The `parse`, `xpointer`, and `encoding` attributes of the XInclude element (`xi:include`), and the `fallback` child element of `xi:include` can also be inserted via the dialog. Do this by first checking the appropriate check box and then selecting/entering the required values. In the case of the `fallback` element, checking its check box only inserts the empty element. The content of the `fallback` element must be added subsequently in one of the editing views.

The `parse` attribute determines whether the included document is to be parsed as XML or text. (XML is the default value and therefore need not be specified.) The `xpointer` attribute identifies a specific fragment of the document located with the `href` attribute; it is this fragment that will be included. The `encoding` attribute specifies the encoding of the included document so that XMLSpy can transcode this document (or the part of it to be included) into the encoding of the including document. The contents of the `fallback` child element replace the `xi:include` element if the document to be included cannot be located.

Here is an example of an XML document that uses XInclude to include two XML documents:

```
<?xml version="1.0" encoding="UTF-16"?>
<AddressBook xsi:schemaLocation="http://www.altova.com/sv/myaddresses
AddressBook.xsd"
       xmlns="http://www.altova.com/stylevision/tutorials/myaddresses"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="BusinessAddressBook.xml"/>
       <xi:include href="PersonalAddressBook.xml"/>
</AddressBook>
```

When this XML document is parsed, it will replace the two XInclude elements with the files specified in the respective `href` attributes.

xml:base
When the XML validator of XMLSpy reads an XML document and encounters the `include` element in the XInclude namespace (hereafter `xi:include`), it replaces this element (`xi:include`) with the XML document named in the `href` attribute of the `xi:include` element. The document element (root element) of the included XML document (or the element identified by

an XPointer) will be included with an attribute of `xml:base` in order to preserve the base URIs of the included element. If the resulting XML document (containing the included XML document/s or tree fragment/s) must be valid according to a schema, then the document element of the included document (or the top-level element of the tree fragment) must be created with a content model that allows an attribute of `xml:base`. If, according to the schema, the `xml:base` attribute is not allowed on this element, then the resulting document will be invalid. How to define an `xml:base` attribute in an element's content model using XMLSpy's Schema View is described in the xml:Prefixed Attributes section of the Schema View section of the documentation.

XPointers

XMLSpy supports XPointers in XInclude. The relevant W3C recommendations are the XPointer Framework and XPointer element() Scheme recommendations. The use of an XPointer in an XInclude element enables a specific part of the XML document to be included, instead of the entire XML document. XPointers are used within an XInclude element as follows:

```
<xi:include href="PersonalAddressBook.xml" xpointer="element(usa)"/>
<xi:include href="BusinessAddressBook.xml" xpointer="element(/1/1)"/>
<xi:include href="BobsAddressBook.xml" xpointer="element(usa/3/1)"/>
<xi:include href="PatsAddressBook.xml" xpointer="element(usa)
element(/1/1)"/>
```

In the element() scheme of XPointer, an NCName or a child sequence directed by integers may be used.

- In the first `xi:include` element listed above, the `xpointer` attribute uses the element scheme with an NCName of `usa`. According to the XPointer Framework, this NCName identifies the element that has an ID of `usa`.
- In the second `xi:include` listed above, the `xpointer` attribute with a value of `element(/1/1)` identifies, in the first step, the first child element of the document root (which, if the document is well-formed, will be its document (or root) element). In the second step, the first child element of the element located in the previous step is located; in our example, this would be the first child element of the document element.
- The `xpointer` attribute of the third `xi:include` listed above uses a combination of NCName and child sequence. This XPointer locates the first child element of the third child element of the element having an ID of `usa`.
- If you are not sure whether your first XPointer will work, you can back it up with a second one as shown in the fourth `xi:include` listed above: `xpointer="element(usa) element(/1/1)"`. Here, if there is no element with an ID of `usa`, the back-up XPointer specifies that the first child element of the document element is to be selected. Additional backups are also allowed. Individual XPointers may not be separated, or they may be separated by whitespace: for example, `xpointer="element(usa) element(addresses/1) element(/1/1)"`.

Note: The namespace binding context is not used in the element() scheme because the element() scheme does not support qualified names.

Append DOCTYPE

The **XML** | **Append** | **DOCTYPE** command is available in the Grid View of an XML file when a top-level node is selected. It appends a DOCTYPE declaration at the top of the XML document. You must enter the name of the DOCTYPE, and this name must be the same as the name of the document element.

After you have entered the name of the DOCTYPE, you can enter the declarations you wish to use in the internal DTD subset.

Please note:

- A DOCTYPE declaration may only appear between the XML declaration and the XML document element.
- You could use the Assign DTD command instead to create a DOCTYPE statement that refers to an external DTD document.

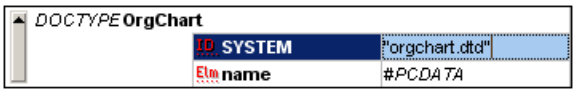
Append ExternalID

A DOCTYPE declaration in an XML file can contain a reference to an external resource containing DTD declarations. This resource is referenced either through a public or system identifier. For example:

```
<!DOCTYPE doc_element_name PUBLIC "publicID" "systemID">
<!DOCTYPE doc_element_name SYSTEM "systemID">
```

A system identifier is a URI that identifies the external resource. A public identifier is location-independent and can be used to dereference the location of an external resource. For example, in your XMLSpy installation, URIs for popular DTDs and XML Schemas are listed in the catalog files named `catalog.xml` in the various schema folders in `C:\Program Files\Altova\Common2017\Schemas\`. A public identifier in an XML document can be used to dereference a DTD listed in these catalog files.

The **XML** | **Append** | **ExternalID** command is available when a "child" item of the DOCTYPE declaration in an XML file is selected in Grid View. This command inserts a Grid View row for an external identifier (`PUBLIC` or `SYSTEM`). You must enter the type of identifier and its value.



The Text View corresponding to the screenshot of the Grid View shown above looks something like this:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

Please note: A row for External-ID can be added as a child when the DOCTYPE item is selected, or it can be inserted or appended when one of the child items of the DOCTYPE item is selected, for example, the ELEMENT declaration `name` in the example above.

Append ELEMENT

The **XML** | **Append** | **ELEMENT** command is available in Grid View only, for DTD documents or

when an item in the DOCTYPE declaration of an XML document is selected. It appends an ELEMENT declaration to the list of declarations.

Append ATTLIST

The **XML** | **Append** | **ATTLIST** command is available in Grid View only, for DTD documents or when an item in the DOCTYPE declaration of an XML document is selected. It appends an ATTLIST declaration to the list of declarations.

Append ENTITY

The **XML** | **Append** | **ENTITY** command is available in Grid View only, for DTD documents or when an item in the DOCTYPE declaration of an XML document is selected. It appends an ENTITY declaration to the list of declarations.

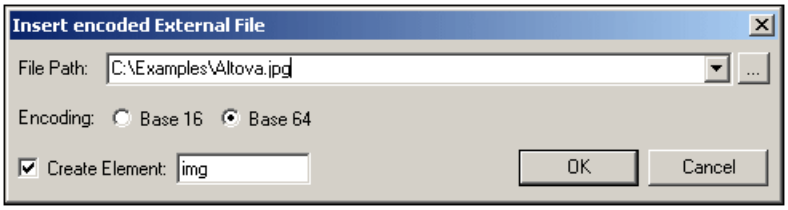
Append NOTATION

The **XML** | **Append** | **NOTATION** command is available in Grid View only, for DTD documents or when an item in the DOCTYPE declaration of an XML document is selected. It appends a NOTATION declaration to the list of declarations.

Append Encoded External File

The **XML** | **Append** | **Encoded External File** command is available in Grid View only. It appends a binary encoded file, such as an image file, as encoded characters. The encoded external file is appended after the Grid View selection.

On clicking the command, the Insert Encoded External File dialog (*screenshot below*) pops up. In it you enter the path to the file, select the encoding you want, and specify whether the encoded file is to be inserted in an element or not.



You can browse for or enter the name of the external file to be encoded and embedded. Either a Base-16 or Base-64 encoding must be specified. If you wish to enclose the encoded text in an element, then check the Create Element check box and specify the name of the desired element in the Create Element text box. If the Create Element check box is not checked, then the encoded text will be inserted directly at the cursor location.

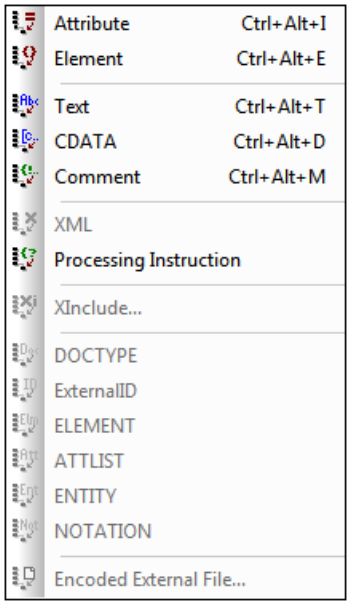
On clicking **OK**, the encoded text of the selected file is inserted at the cursor location, with an enclosing element if this has been specified.

The encoded file is appended in Grid View.

26.4.3 Add Child

The **XML** | **Add Child** command, though enabled in all views, can be used in Grid View only. It opens a submenu (*see screenshot*) with which you can add the following child items to the currently selected element.

- The XML declaration and node types (Attribute, Element, Text, CDATA, Comment, Processing Instruction) in XML documents;
- DOCTYPE declarations and external DTD declarations in XML documents
- DTD declarations (ELEMENT, ATTLIST, ENTITY, and NOTATION) in DTD documents and internal DTD declarations of XML documents.



Add Child Attribute

Ctrl+Alt+I

The **XML** | **Add Child** | **Attribute** command is available in Grid View only and when an element node is selected. It inserts a new attribute as a child of the selected element node.

Add Child Element

Ctrl+Alt+E

The **XML** | **Add Child** | **Element** command is available in Grid View only. It inserts a new element as a child of the selected node.

Add Child Text

Ctrl+Alt+T

The **XML** | **Add Child | Text** command is available in Grid View only, and inserts new text content as a child of the selected item.

Add Child CDATA

Ctrl+Alt+D

The **XML** | **Add Child | Cdata** command is available in Grid View only, and inserts a new CDATA section as a child of the selected item.

Add Child Comment

Ctrl+Alt+M

The **XML** | **Add Child | Comment** command is available in Grid View only, and inserts a new Comment node as a child of the selected item.

Add Child XML

The **XML** | **Add Child | XML** command is available in Grid View only and when the file is **empty**. It inserts a new XML declaration `<?xml version="1.0" encoding="UTF-8"?>` as the first item in a document.

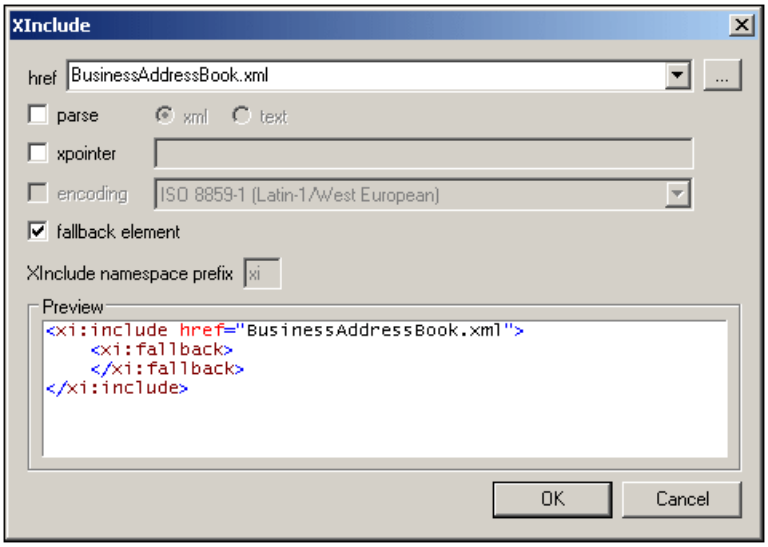
Please note: An XML document may contain only one XML declaration, which must appear at the very top of the file.

Add Child Processing Instruction

The **XML** | **Add Child | Processing Instruction** command is available in Grid View only and inserts a new Processing Instruction (PI) as a child of the selected item.

Add Child XInclude

The **XML | Add Child | XInclude** command is available in Grid View only, and enables you to insert an XInclude element as a child of the selected item. Selecting this command pops up the XInclude dialog (*screenshot below*).



The XML file to be included is entered in the `href` text box (alternatively, you can browse for the file by clicking the **Browse (...)** button to the right of the text box). The filename will be entered in the XML document as the value of the `href` attribute. The `parse`, `xpointer`, and `encoding` attributes of the XInclude element (`xi:include`), and the `fallback` child element of `xi:include` can also be inserted via the dialog. Do this by first checking the appropriate check box and then selecting/entering the required values. In the case of the `fallback` element, checking its check box only inserts the empty element. The content of the `fallback` element must be added subsequently in one of the editing views.

The `parse` attribute determines whether the included document is to be parsed as XML or text. (XML is the default value and therefore need not be specified.) The `xpointer` attribute identifies a specific fragment of the document located with the `href` attribute; it is this fragment that will be included. The `encoding` attribute specifies the encoding of the included document so that XMLSpy can transcode this document (or the part of it to be included) into the encoding of the including document. The contents of the `fallback` child element replace the `xi:include` element if the document to be included cannot be located.

Here is an example of an XML document that uses XInclude to include two XML documents:

```
<?xml version="1.0" encoding="UTF-16"?>
<AddressBook xsi:schemaLocation="http://www.altova.com/sv/myaddresses
AddressBook.xsd"
       xmlns="http://www.altova.com/stylevision/tutorials/myaddresses"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="BusinessAddressBook.xml"/>
       <xi:include href="PersonalAddressBook.xml"/>
</AddressBook>
```

When this XML document is parsed, it will replace the two XInclude elements with the files specified in the respective `href` attributes.

xml:base

When the XML validator of XMLSpy reads an XML document and encounters the `include` element in the XInclude namespace (hereafter `xi:include`), it replaces this element (`xi:include`) with the XML document named in the `href` attribute of the `xi:include` element. The document element (root element) of the included XML document (or the element identified by

an XPointer) will be included with an attribute of `xml:base` in order to preserve the base URIs of the included element. If the resulting XML document (containing the included XML document/s or tree fragment/s) must be valid according to a schema, then the document element of the included document (or the top-level element of the tree fragment) must be created with a content model that allows an attribute of `xml:base`. If, according to the schema, the `xml:base` attribute is not allowed on this element, then the resulting document will be invalid. How to define an `xml:base` attribute in an element's content model using XMLSpy's Schema View is described in the xml:Prefixed Attributes section of the Schema View section of the documentation.

XPointers

XMLSpy supports XPointers in XInclude. The relevant W3C recommendations are the XPointer Framework and XPointer element() Scheme recommendations. The use of an XPointer in an XInclude element enables a specific part of the XML document to be included, instead of the entire XML document. XPointers are used within an XInclude element as follows:

```
<xi:include href="PersonalAddressBook.xml" xpointer="element(usa)"/>
<xi:include href="BusinessAddressBook.xml" xpointer="element(/1/1)"/>
<xi:include href="BobsAddressBook.xml" xpointer="element(usa/3/1)"/>
<xi:include href="PatsAddressBook.xml" xpointer="element(usa)
element(/1/1)"/>
```

In the element() scheme of XPointer, an NCName or a child sequence directed by integers may be used.

- In the first `xi:include` element listed above, the `xpointer` attribute uses the element scheme with an NCName of `usa`. According to the XPointer Framework, this NCName identifies the element that has an ID of `usa`.
- In the second `xi:include` listed above, the `xpointer` attribute with a value of `element(/1/1)` identifies, in the first step, the first child element of the document root (which, if the document is well-formed, will be its document (or root) element). In the second step, the first child element of the element located in the previous step is located; in our example, this would be the first child element of the document element.
- The `xpointer` attribute of the third `xi:include` listed above uses a combination of NCName and child sequence. This XPointer locates the first child element of the third child element of the element having an ID of `usa`.
- If you are not sure whether your first XPointer will work, you can back it up with a second one as shown in the fourth `xi:include` listed above: `xpointer="element(usa) element(/1/1)"`. Here, if there is no element with an ID of `usa`, the back-up XPointer specifies that the first child element of the document element is to be selected. Additional backups are also allowed. Individual XPointers may not be separated, or they may be separated by whitespace: for example, `xpointer="element(usa) element(addresses/1) element(/1/1)"`.

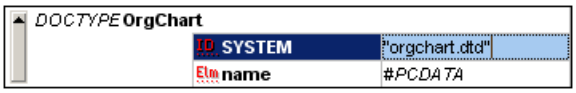
Note: The namespace binding context is not used in the element() scheme because the element() scheme does not support qualified names.

Add Child DOCTYPE

The **XML** | **Add Child** | **DOCTYPE** command is only available in the Grid View of an **empty** document. It inserts a DOCTYPE declaration in an XML document. The DOCTYPE declaration can be used to declare an internal DTD subset.

Add Child ExternalID

The **XML** | **Add Child** | **ExternalID** command is available only when the DOCTYPE declaration of an XML file is selected in Grid View. This command inserts a Grid View row for an external identifier (`PUBLIC` or `SYSTEM`). You must enter the type of identifier and its value.



The Text View corresponding to the screenshot of the Grid View shown above looks something like this:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

Please note: A row for External-ID can be added as a child when the DOCTYPE item is selected, or it can be inserted or appended when one of the child items of the DOCTYPE item is selected, for example, the ELEMENT declaration `name` in the example above.

Add Child ELEMENT

The **XML** | **Add Child** | **ELEMENT** command is available in Grid View only, for DTD documents, or when the DOCTYPE declaration of an XML document is selected. It appends an ELEMENT declaration to the list of declarations.

Add Child ATTLIST

The **XML** | **Add Child** | **ATTLIST** command is available in Grid View only, for DTD documents, or when the DOCTYPE declaration of an XML document is selected. It appends an ATTLIST declaration to the list of declarations.

Add Child ENTITY

The **XML** | **Add Child** | **ENTITY** command is available in Grid View only, for DTD documents, or when the DOCTYPE declaration of an XML document is selected. It appends an ENTITY declaration to the list of declarations.

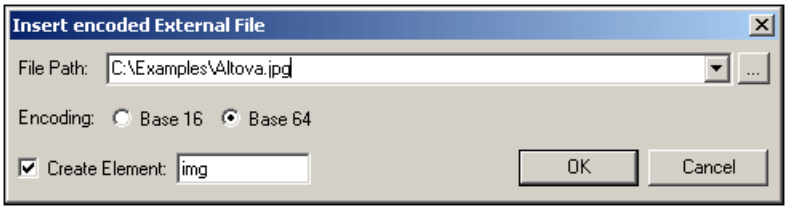
Add Child NOTATION

The **XML** | **Add Child | NOTATION** command is available in Grid View only, for DTD documents, or when the DOCTYPE declaration of an XML document is selected. It appends an NOTATION declaration to the list of declarations.

Add Child Encoded External File

The **XML** | **Add Child | Encoded External File** command is available in Grid View only. It adds a binary encoded file as a child node. The encoded external file is inserted as a child of the Grid View selection.

On clicking the command, the Insert Encoded External File dialog (*screenshot below*) pops up. In it you enter the path to the file, select the encoding you want, and specify whether the encoded file is to be inserted in an element or not.

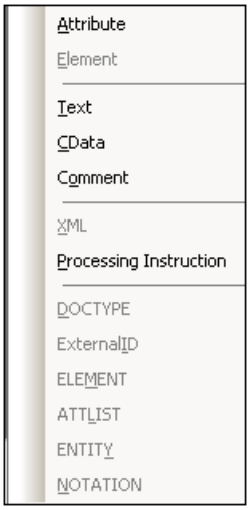


On clicking **OK**, the encoded text of the selected file is inserted at the cursor location, with an enclosing element if this has been specified.

The encoded file is added as a child in Grid View.

26.4.4 Convert To

The **XML** | **Convert to** command converts a selected item in Grid View to a different item type. This operation is available only in Grid View on individual items that do not contain any child node. Placing the cursor over the **Convert to** command displays a submenu (*see screenshot*) which contains the items to which the selected item can be converted.



Please note: If the operation you select would result in a loss of data (for example, converting an attribute to a comment would result in a loss of the attribute name), a warning dialog box will appear.

Convert To Attribute

The **XML** | **Convert to** | **Attribute** command converts the selected item to a attribute.

Convert To Element

The **XML** | **Convert to** | **Element** command converts the selected item to an element.

Convert To Text

The **XML** | **Convert to** | **Text** command converts the selected item into text content.

Convert To CDATA

The **XML** | **Convert to** | **Cdata** command converts the selected item into a CDATA segment.

Convert To Comment

The **XML** | **Convert to** | **Comment** command converts the selected item into a comment.

Convert To XML

The **XML** | **Convert to** | **XML** command converts the selected item to an XML declaration:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Please note: Each XML document may only contain one XML declaration and it must appear at the very top of the file.

Convert To Processing Instruction

The **XML** | **Convert to** | **Processing Instruction** command converts the selected item to a new Processing Instruction (PI).

Convert To DOCTYPE

The **XML** | **Convert to** | **DOCTYPE** command converts the selected item to a DOCTYPE declaration (in an XML file).

Please note: A DOCTYPE declaration may only appear at the top of an XML instance document between the XML Declaration and the document element of the XML document.

Convert To ExternalID

The **XML** | **Convert to** | **ExternalID** command converts the selected item to an external DTD reference in a DOCTYPE declaration (`PUBLIC` or `SYSTEM` identifier).

Convert To ELEMENT

The **XML** | **Convert to** | **ELEMENT** command converts the selected item to an element declaration in a DOCTYPE declaration or in an external DTD.

Convert To ATTLIST

The **XML** | **Convert to** | **ATTLIST** command converts the selected item to an attribute list declaration in a DOCTYPE declaration or in an external DTD.

Convert To ENTITY

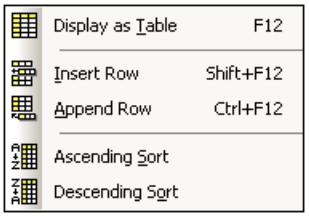
The **XML** | **Convert to** | **ENTITY** command converts the selected item to an entity declaration in a DOCTYPE declaration or in an external DTD.

Convert To NOTATION

The **XML** | **Convert to** | **NOTATION** command converts the selected item to a notation declaration in a DOCTYPE declaration or in an external DTD.

26.4.5 Table

The **XML** | **Table** command, though enabled in all views, can be used only in Grid View. It displays a submenu with all the commands relevant to the Database/Table View of Grid View.



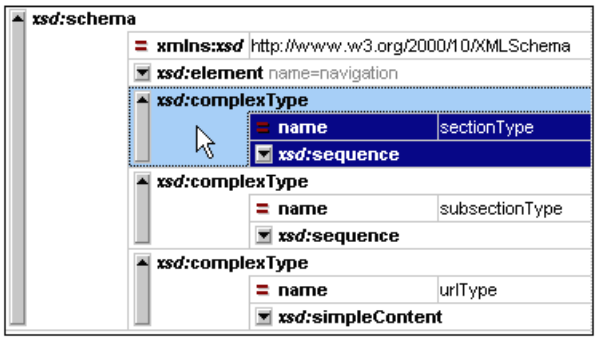
Display as Table

Display as Table (F12) menu command and toolbar icon

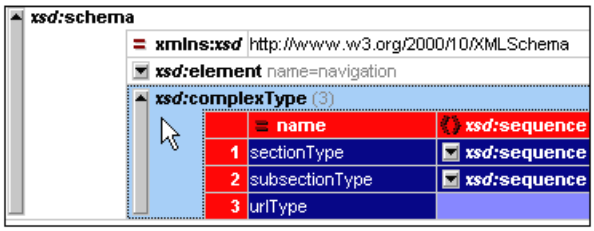
The **XML** | **Table** | **Display as Table** command allows you to switch between the standard Grid View and Database/Table View (or Table View) of a document element. The Table View enables you to view repeated elements as a table in which the rows represent the occurrences while the columns represent child nodes (including comments, CDATA sections, and PIs).

To switch to Table View:

1. Select any one occurrence of the repeating element you wish to view as a table.



2. Click **XML | Table | Display as Table** or **F12** or the toolbar icon.



The element is displayed as a table and the **Display as Table** toolbar icon is toggled on.

To switch from the Table View of a document element to the normal Grid View of that element, select the table or any of its rows or columns, and click the **Display as Table** toolbar icon. That table element switches to Grid View, and the icon is toggled off.

Note: Table View colors can be set in the Colors tab of the Options dialog (**Tools | Options | Colors**)

Insert Row

Shift+F12

The **XML | Table | Insert Row** command is enabled in Database/Table View when a row or cell is selected. It inserts a new row before the selected row. The new row corresponds to an occurrence of the table element. Mandatory child elements are created for the new element

Append Row

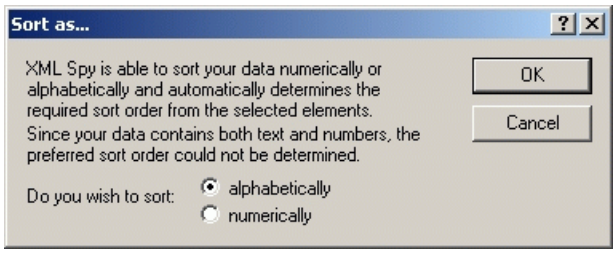
Ctrl+F12

The **XML | Table | Append Row** command is enabled in Database/Table View when a row or cell is selected. It appends a new row after the last row of the table. The new row corresponds to an occurrence of the table element. Mandatory child elements are created for the new element

Ascending Sort

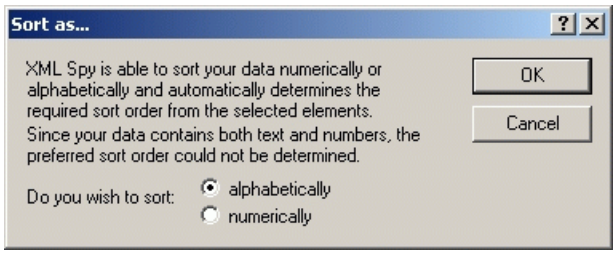
The **XML | Table | Ascending Sort** command is enabled in Database/Table View when a column

or cell is selected. It sorts the column in either alphabetic or numeric ascending order. XMLSpy tries to automatically determine what kind of data is used in the column, and sorts on alphabetic or numeric order, as required. In case of uncertainty, you will be prompted for the sort method to use (*see screenshot*).



Descending Sort

The **XML | Table | Descending Sort** command is enabled in Database/Table View when a column or cell is selected. It sorts the column in either alphabetic or numeric descending order. XMLSpy tries to automatically determine what kind of data is used in the column, and sorts on alphabetic or numeric order, as required. In case of uncertainty, you will be prompted for the sort method to use (*see screenshot*).



26.4.6 Move Left

Ctrl+L

The **XML | Move Left** command is available in Grid View only. It moves the selected node to the left by one level, thereby changing a child element into a sibling of its parent. This command is often referred to as the **Promote** command.

26.4.7 Move Right

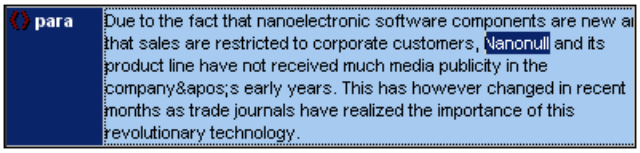
Ctrl+R

The **XML | Move Right** command is available in Grid View only. It moves the selected node to the right by one level, thereby turning it into a child element of the preceding sibling element. This command is often referred to as the **Demote** command.

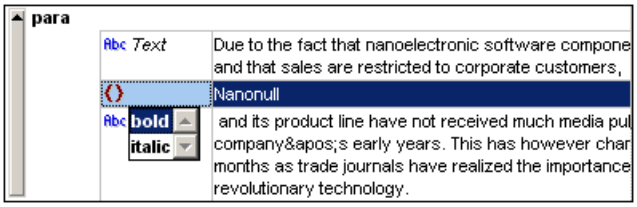
26.4.8 Enclose in Element

The **XML** | **Enclose in Element** command is enabled in Grid View only. It encloses a selected text range in a new element. The new element is created inline around the selected text. If you are editing a document based on a Schema or DTD, you will automatically be presented with a list of valid choices for the name of the element in which the text is to be enclosed.

For example, in the screenshot below, the text `Nanonull` in the `para` element is highlighted.



When you select the command **XML | Enclose in Element**, the text Nanonull is enclosed in a newly created inline element and a list appears offering a choice of `bold` or `italic` for the name of the element. These elements are defined in the schema as children of `para`.



The selection you make will be the name of the new element. Alternatively, you can enter some other name for the element.

26.4.9 Evaluate XPath

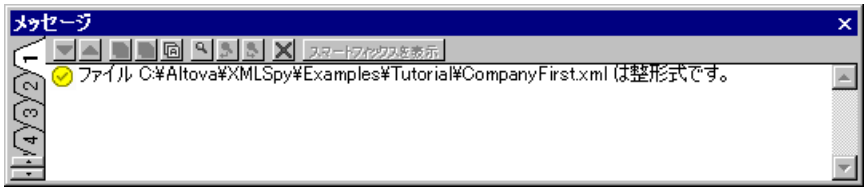
The **XML** | **Evaluate XPath** command opens the Output Windows if these are not open and activates the XPath tab in the Output Windows. In the XPath tab, you can evaluate an XPath expression on the active document and see the results in the Output Window.

26.4.10 Check Well- Formedness

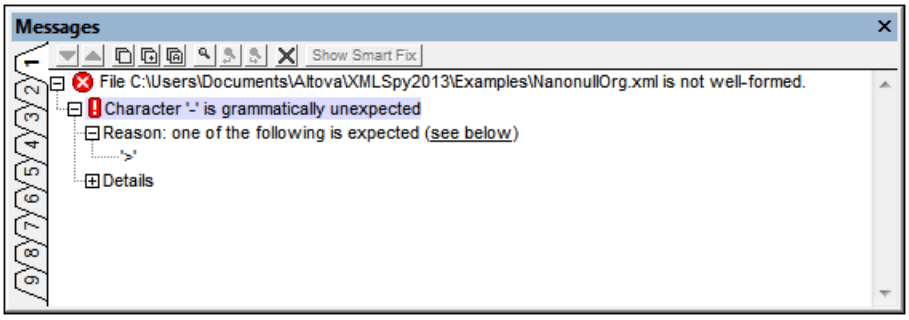
F7

The **XML** | **Check well-formedness (F7)** command checks the active document for well-formedness by the definitions of the XML 1.0 specification. Every XML document **must** be well-formed. XMLSpy checks for well-formedness whenever a document is opened or saved, or when the view is changed from Text View to any other view. You can also check for well-formedness at any time while editing by using this command.

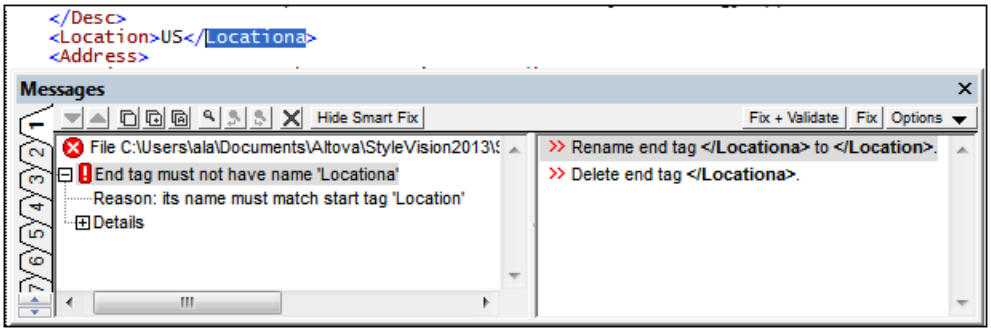
If the well-formedness check succeeds, a message is displayed in the Messages window (*screenshot below*).



If an error is encountered during the well-formedness check, a corresponding error message is displayed (*screenshot below*).



Based on information in the schema, options for a smart fix are also suggested if the well-formedness check was carried out in **Text View or Grid View**. To view a list of smart fix options, click the **Show Smart Fix** button (*see screenshot above*). A pane with suggested smart fix options appears in the Messages window (*see screenshot below*). Note that errors in the Messages window are displayed one error at a time.



In the Smart Fix pane, select one of the suggested smart fixes and click either the **Fix + Validate** button or the **Fix** button (*see screenshot above*). The error will be corrected with the smart fix. Alternatively, you can double-click the smart fix you want. This action either fixes the error, or fixes the error and validates the document, according to the option selected in the dropdown *Options* list. The **Fix + Validate** command is useful because the validator will now be able to validate beyond the fixed error and pick up the next well-formed error or validation error, if there is any.

To hide the Smart Fix pane, click the **Hide Smart Fix** button (*see screenshot above*).

Note: The Messages window has nine tabs. The validation result is always displayed in the active tab. So you can validate one XML document in Tab-1 and retain the result in that tab. To validate a second document, switch to Tab-2 (or Tab-3 if you like) before running the check. If you do not switch tabs, Tab-1 (or the active tab) will be overwritten with the results of the latest validation.

Validating from the Project window

The **Validate** command can also be applied to a file, folder, or group of files in the active project. Select the required file or folder in the Project Window (by clicking on it). Then click **XML | Validate** or **F8**. Invalid files in a project will be opened and made active in the Main Window, and the *File is not valid* error message will be displayed.

Note: The Messages window has nine tabs. The result of the well-formed check is always displayed in the active tab. So you can check the well-formedness of one XML document in Tab-1 and retain the result in that tab. To check the well-formedness of a second document, switch to Tab-2 (or Tab-3 if you like) before running the check. If you do not switch tabs, Tab-1 (or the active tab) will be overwritten with the results of the latest check.

It is generally not permitted to save a malformed XML document, but XMLSpy gives you a Save Anyway option. This is useful when you want to suspend your work temporarily (in a not well-formed condition) and resume it later.

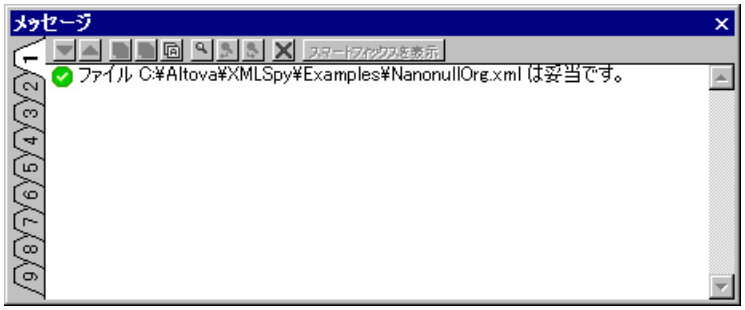
Note: You can also use the **Check well-formedness** command on any file, folder, or group of files in the active project window. Click on the respective item, and then on the Check Well-Formedness icon.

26.4.11 Validate XML

F8

The **XML** | **Validate (F8)** command enables you to validate XML documents against DTDs, XML Schemas, and other schemas. Validation is automatically carried out when you switch from Text View to any other view. You can specify that a document be automatically validated when a file is opened or saved (**Tools | Options | File**). The **Validate** command also carries out a well-formedness check before checking validity, so there is no need to use the **Check Well-Formedness** command before using the **Validate** command.

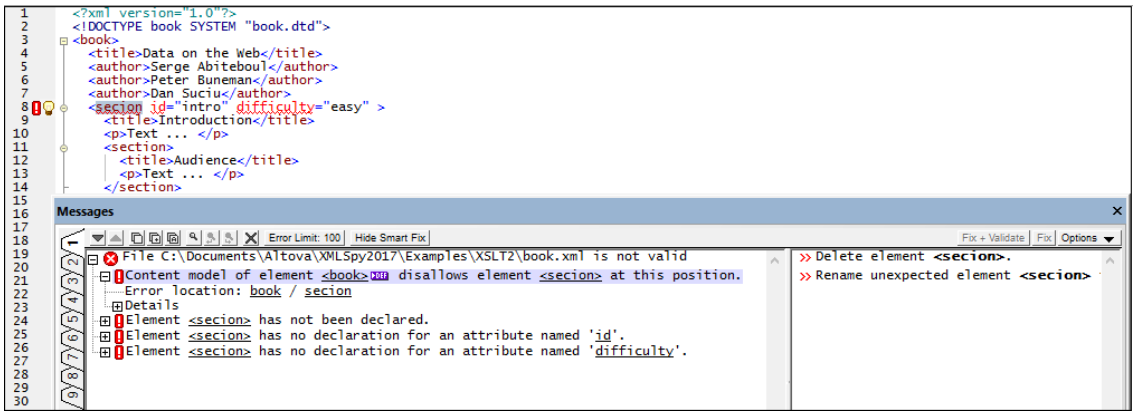
If a document is valid, a successful validation message is displayed in the Messages window.



Otherwise, a message that describes the error is displayed. You can click on the links in the error message to jump to the node in the XML document where the error was found. See the next section below for a description of the error message and how to fix validation errors with the smart fixes of XMLSpy.

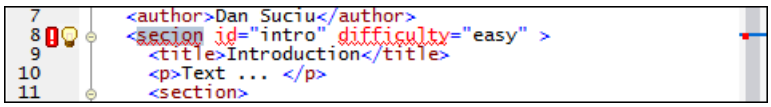
Validation errors and their fixes

When a validation error is displayed in the Messages window, the causes of the error are displayed in the left-hand pane (*see screenshot below*). If a cause is selected in the left-hand pane, then smart fixes for it, if available, are displayed in the right-hand pane. Smart fix suggestions are available in **Text View** and **Grid View**, and are based on information in the associated schema. To view smart fixes, click the **Show Smart Fix** button. Click **Hide Smart Fix** if you do not want these suggestions to be displayed. Note that errors of well-formedness (such as mismatched start and end tags), if such exist, are displayed prior to validation errors being displayed. So the **Show/Hide Smart Fix** button will be enabled only when a validation error is reached (that is, after alll well-formedness errors have been corrected).

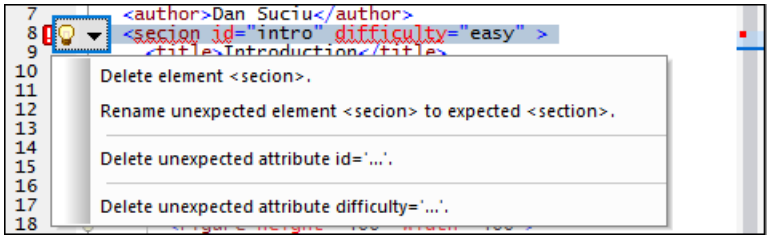


To apply a smart fix, either (i) double-click it, or (ii) select it and click either the **Fix** or **Fix + Validate** options (*see screenshot below*). The **Fix + Validate** command will validate beyond the fixed error and pick up the next error, if there is any.

In Text View, there are two additional indicators of a validation error (*see screenshot below*): (i) a red exclamation-mark icon in the line-numbering margin, and (ii) a red marker-square in the scroll bar (on the right of the window).



The light-bulb icon next to the exclamation-mark icon (*see screenshot above*) is the smart-fix icon. If you hover over it, all smart fixes across all causes of the error are displayed (*see screenshot below*). Select a smart fix to apply it.



Note: The validation error indicators and smart fixes described above are refreshed only when the **XML** | **Validate (F8)** command is executed; they are not updated in the background. So, after correcting an error, you must run the **Validate (F8)** command again to make sure that the error has indeed been fixed.

Note: The Messages window has nine tabs. The validation result is always displayed in the active tab. So you can validate one XML document in Tab-1 and retain the result in that tab. To validate a second document, switch to Tab-2 (or Tab-3 if you like) before running the check. If you do not switch tabs, Tab-1 (or the active tab) will be overwritten with the results of the latest validation.

Validating from the Project window

The **Validate** command can also be applied to a file, folder, or group of files in the active project. Select the required file or folder in the Project Window (by clicking on it). Then click **XML | Validate** or **F8**. Invalid files in a project will be opened and made active in the Main Window, and the *File is not valid* error message will be displayed..

Validating XML documents

To validate an XML file, make the XML document active in the Main Window, and click **XML | Validate** or **F8**. The XML document is validated against the schema referenced in the XML file. If no reference exists, an error message is displayed in the Messages window. As long as the XML document is open, the schema is kept in memory (see Flush Memory Cache in the DTD/Schema menu).

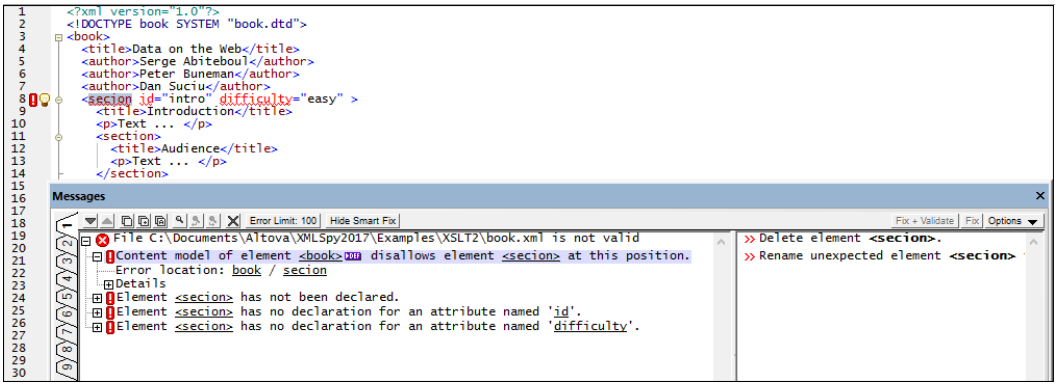
Validating schema documents (DTDs and XML Schema)

XMLSpy supports major schema dialects, including DTD and XML Schema. To validate a schema document, make the document active in the Main Window, and click **XML | Validate** or **F8**.

Validation messages

There are two kinds of messages:

- If the schema (DTD or XML Schema) is valid, a successful validation message is displayed in the Messages window.
- If the schema is not valid, an error message is displayed in the Messages window (*screenshot below*).



An error message shows each possible cause of that error separately. For example, in the screenshot above, four possible causes of the validation error are reported; the first one is expanded, the other three are collapsed. Each cause is divided into three parts:

1. A description of the possible cause. The description contains links to the relevant definition in the associated schema document. You can quickly go to the specific schema definition to see why exactly the document is invalid.
2. The location path to the node in the XML document that has caused the error. Clicking any node in this location path highlights that node in the document.
3. Detailed information about the error, as well as a link to the relevant paragraph in the schema specification. This is where the schema rules that specify the relevant legality are specified.

Note: If the validation is done in Text View, then clicking a link in the Messages window will *highlight* the corresponding definition in Text View. If the validation is done in Schema View, then clicking a definition link will *open* the definition in Schema View and allow you to *edit the component directly*.

Catalogs

XMLSpy supports a subset of the OASIS XML catalogs mechanism. The catalog mechanism enables XMLSpy to retrieve commonly used schemas (as well as stylesheets and other files) from local user folders. This increases the overall processing speed, enables users to work offline (that is, not connected to a network), and improves the portability of documents (because URIs need to be changed in the catalog files only.) The catalog mechanism in XMLSpy works as follows:

- XMLSpy loads a file called `RootCatalog.xml`, which contains a list of catalog files that will be looked up. You can enter as many catalog files to look up, each in a `nextCatalog` element in `RootCatalog.xml`.
- The catalog files included in `RootCatalog.xml` are looked up and the URIs are resolved according to the mappings specified in the catalog files. You should take care not to duplicate mappings, as this could lead to errors.
- Two catalog files are supplied with XMLSpy. How these work is described in the section Catalogs in XMLSpy.
- The `PUBLIC` or `SYSTEM` identifier in the `DOCTYPE` statement of your XML file will be used for the catalog lookup. For popular schemas, the `PUBLIC` identifier is usually pre-defined, thus requiring only the URI in the catalog file to be changed when XML documents are used on multiple machines.

When writing your `CustomCatalog.xml` file (or other custom catalog file), use only the following

subset of the OASIS catalog in order for XMLSpy to process the catalog correctly. Each of the elements in the supported subset can take the `xml:base` attribute, which is used to specify the base URI of that element.

```
<catalog...>
...
<public publicId="PublicID of Resource" uri="URL of local file"/>
<system systemId="SystemID of Resource" uri="URL of local file"/>
<rewriteURI uriStartString="StartString of URI to rewrite"
rewritePrefix="String to replace StartString"/>
<rewriteSystem systemIdStartString="StartString of SystemID"
rewritePrefix="Replacement string to locate resource locally"/>
<uri name="filename" uri="URL of file identified by filename"/>
...
</catalog>
```

Note:

- The `catalog.xml` file in the `%AltovaCommonFolder%\Schemas\schema` folder contains references to DTDs that implement older XML Schema specifications. You should not validate your XML Schema documents against any of these schemas. The referenced DTD files are included solely to provide XMLSpy with entry helper info for editing purposes should you wish to create documents according to these older recommendations. *Also see next point*.
- If you create a custom file extension for a particular schema (for example, the `.myhtml` extension for (HTML) files that are to be valid according to the HTML DTD), then you can enable intelligent editing for files with these extensions by adding a line of text to `CustomCatalog.xml`. For the example extension mentioned, you should add the element `<spy:fileExtHelper ext="myhtml" uri="schemas/xhtml/xhtml1-transitional.dtd"/>` as a child of the `<catalog>` element. This would enable intelligent editing (auto-completion, entry helpers, etc) of `.myhtml` files in XMLSpy according to the XHTML 1.0 Transitional DTD.
- For more information on catalogs, see the XML Catalogs specification.

Automating validation with RaptorXML 2017

RaptorXML is Altova's standalone application for XML validation, XSLT transformation, and XQuery transformation. It can be used from the command line, via a COM interface, in Java programs, and in .NET applications. Validation tasks can therefore be automated with the use of RaptorXML. For example, you can create a batch file that calls RaptorXML to perform validation on a set of documents and sends the output to a text file. See the RaptorXML documentation for details.

26.4.12 Validate XML on Server (high-performance)

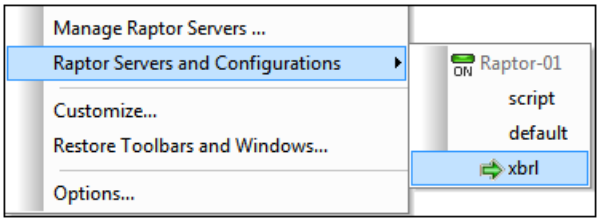
Ctrl+F8

The **XML | Validate on Server (high-performance) (Ctrl+F8)** command validates the active XML document on the currently active RaptorXML Server using the active configuration. The command immediately carries out the validation and displays the results in the Messages window.

Note: The actual performance depends on the number of PC processor cores used by

RaptorXML Server for the validation: The higher the number of cores used, the faster will be the processing.

If you have defined multiple configurations on multiple servers, you can select a server and one of its configurations as the active configuration. The active configuration will be used for subsequent validations. On placing the cursor over the **Tools | Raptor Servers and Configurations** command (*see screenshot below*), a submenu appears that contains all the added servers, together with the configuration of each. Select the server configuration you want to make the active configuration. In the screenshot below, the `xbrl` configuration of the server named `Raptor-01` has been selected as the active configuration (indicated by the green arrow).



Note: You can also select the active configuration in the dropdown menu of the **Validate on Server** icon . This menu also has a command to validate EDGAR on the active server.

The **Validate XML on Server (high-performance) (Ctrl+F8)** command is also available in the Project entry helper. Right-click the project, a folder, or a file, and select **Validate XML on Server** to validate XML or XBRL data in the selected object.

Note: Raptor validation is available in Text View, Grid View, and XBRL View.

26.4.13 Validating WSDL Files

The following list contains important information about WSDL validation behavior in the Enterprise and Professional Editions of XMLSpy:

- The Professional Edition performs only simple schema validation, i.e., it treats the WSDL file as an XML file and validates it according to the appropriate schema defined at http://schemas.xmlsoap.org/wsdl/.
- The Enterprise Edition provides WSDL validation that goes beyond the validation provided by the Professional Edition. It does not validate against http://schemas.xmlsoap.org/wsdl/ but rather uses the http://www.altova.com/specs_wsdl.html#_document-s as well as its own logic to provide more valuable validation information in the context of WSDL. Thus it can happen that a WSDL file is valid in the Professional Edition, but not valid in the Enterprise Edition (see Example).
- There is a difference between http://schemas.xmlsoap.org/wsdl/ and http://www.altova.com/specs_wsdl.html#_document-s. http://schemas.xmlsoap.org/wsdl/ is missing the extensibility elements (the specification and the schema should be the same but are not; this difference appears to be an error in the schema).
- The fact that the Professional Edition uses http://schemas.xmlsoap.org/wsdl/ for validation means that any extensibility elements will be considered invalid. The Enterprise Edition used http://www.altova.com/specs_wsdl.html#_document-s, which allows extensibility elements. Thus a WSDL file containing extensibility elements is valid when using the Enterprise edition, and not valid when using the Professional Edition.
- The fact that the file is invalid in the Professional Edition is because we are using the official schema provided by the W3C working group. Any errors in this schema are

beyond our control.

Example
The following example is part of a WSDL file. Notice the element "getCityTime" that has been declared in the file. This element is mistakenly referenced as "getCityTimes". The Enterprise Edition checks if elements that are referenced have previously been declared in the file; the Professional Edition does not. This file (assuming that the rest of the file is valid) would be found to be valid in the Professional Edition, but not valid in the Enterprise Edition (assuming that "getCityTimes" is not defined somewhere else in the file).

```
<s:element name="getCityTime">
      <s:complexType>
            <s:sequence>
                  <s:element minOccurs="0" maxOccurs="1" name="city" type="s:string"/>
            </s:sequence>
      </s:complexType>
</s:element>
<s:element name="abc">
      <s:complexType>
            <s:sequence>
                  <s:element ref="getCityTimes"/>
            </s:sequence>
      </s:complexType>
</s:element>
```

26.4.14 Update Entry Helpers

The **XML** | **Update Entry Helpers** command updates the Entry Helper windows by reloading the underlying DTD or Schema. If you have modified the XML Schema or DTD that an open XML document is based upon, it is advisable to update the Entry Helpers so that the intelligent editing information reflects the changes in the schema.

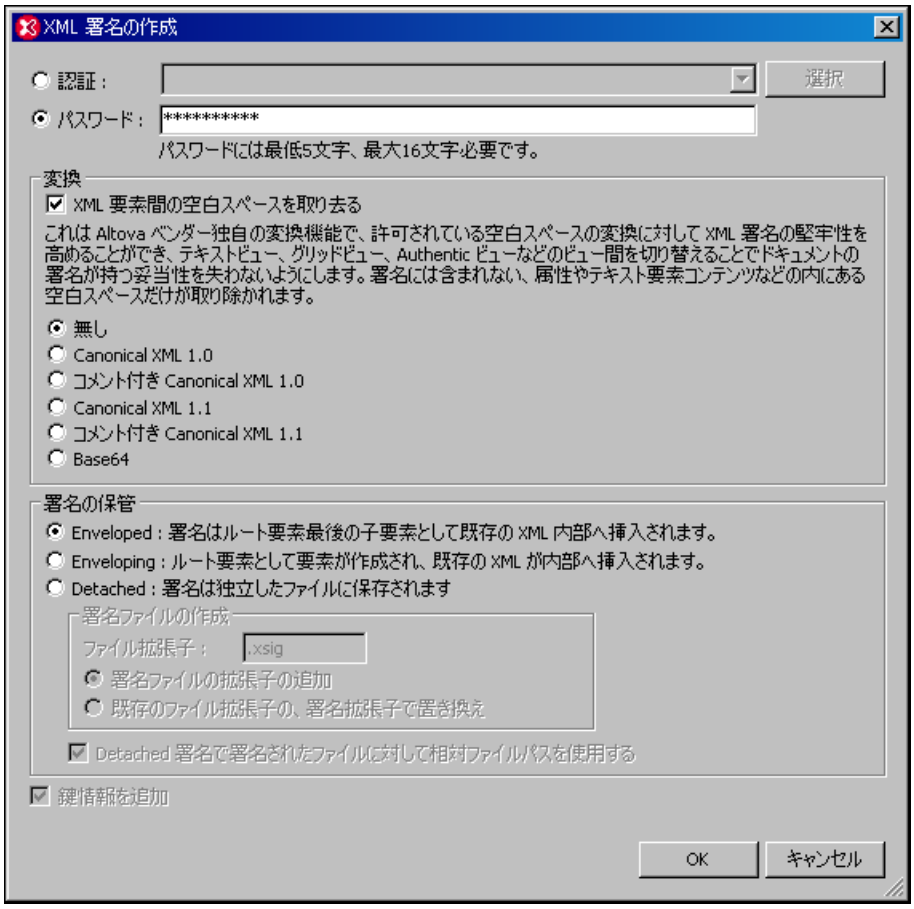
26.4.15 Namespace Prefix

The **XML** | **Namespace Prefix** command is available in Grid View and opens a dialog box in which you can set the namespace prefix of the selected element or attribute, and, in the case of elements, of its descendants as well.

You can choose to set the namespace prefix on either elements, attributes, or both. The namespace prefix is applied to the selected element or attribute, and, if an element is selected, to descendant nodes of the selected element.

26.4.16 Create XML Signature

The **Create XML Signature** command is enabled in Text View, Grid View, Schema View, WSDL View and XBRL View, and enables you to create an XML signature for the active XML document. Clicking the command opens the Create XML Signature dialog (*screenshot below*), the settings of which are explained below.



Authentication method: certificate or password

The signature can be based on a certificate or a password. Select the radio button of the method you wish to use.

- *Certificate:* If you wish to use a certificate, the certificate must have a private key and be located in an accessible certificate store. The signature is generated using the private key of the certificate. To verify the signature, access to the certificate (or a public-key version of it) is required. The public key of the certificate is used to verify the signature. To select the private-public-key certificate you wish to use, click the **Select** button and browse for the certificate. For more details about certificates, see the section Working with Certificates.
- *Password:* Enter a password with a length of five to 16 characters. This password will subsequently be required to verify the signature.

Transformations

The XML data is transformed and the result of the transformation is used for the creation of the

signature. You can specify the canonicalization algorithm to be applied to the file's XML data (the `SignedInfo` content) prior to performing signature calculations. Significant points of difference between the algorithms are noted below:

- *Canonical XML with or without comments:* If comments are included for signature calculation, then any change to comments in the XML data will result in verification failure. Otherwise, comments may be modified or be added to the XML document after the document has been signed, and the signature will still be verified as authentic.
- *Base64:* The root (or document) element of the XML document is considered to be Base64 encoded, and is read in its binary form. If the root element is not Base64, an error is returned or the element is read as empty, depending on what type of element is encountered.
- *None:* No transformation is carried out and the XML data from the binary file saved on disk is passed directly for signature creation. Any subsequent change in the data will result in a failed verification of the signature. However, if the *Strip Whitespace* check box option is selected, then all whitespace is stripped and changes in whitespace will be ignored. A major difference between the *None* option and a *Canonicalization* option is that canonicalization produces an XML data stream, in which some differences, such as attribute order, are normalized. As a result, a canonicalization transformation will normalize any changes such as that of attribute order (so verification will succeed), while no-transformation will reflect such a change (verification will fail).

Signature placement

The signature can be placed within the XML file or be created as a separate file. The following options are available:

- *Enveloped:* The signature element is created as the last child element of the root (document) element.
- *Enveloping:* The signature element is created as the root (document) element and the XML document is inserted as a child element.
- *Detached:* The XML signature is created as a separate file. In this case, you can specify the file extension of the signature file and whether the file name is created with: (i) the extension appended to the name of the XML file (for example, `test.xml.xsig`), or (ii) the extension replacing the XML extension of the XML file (for example, `test.xsig`). You can also specify whether, in the signature file, the reference to the XML file is a relative or an absolute path.

Note: XML signatures for XML Schema (`.xsd`) files and for XBRL files can only be created as external signature files. For WSDL files, signatures can be created as external files and can be "enveloped" in the WSDL file.

Note: If the XML signature is created as a separate file, then the XML file and signature file are associated with each other via a reference in the signature file. Consequently, signature verification in cases where the signature is in an external fie must be done with the signature file active—not with the XML file active.

Append key information

The *Append Keyinfo* option is available when the signature is certificate-based. It is unavailable if the signature is password-based.

If the option is selected, public-key information is placed inside the signature, otherwise key information is not included in the signature. The advantage of including key information is that the certificate itself (specifically the public-key information in it) will not be required for the verification process (since the key information is present in the signature).

26.4.17 Verify XML Signature

An XML signature will be correctly verified if the XML file has not been changed since having been signed. Otherwise the verification will fail. The **Verify XML Signature** command executes the verification process and displays the results of the verification in the Messages windows. The various verification scenarios in XMLSpy are described below:

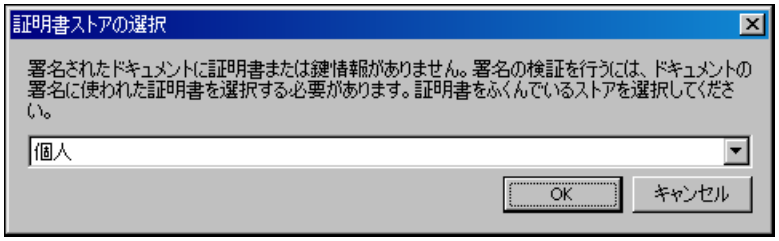
- XML file contains certificate-based signature, key information included in signature
- XML file contains certificate-based signature, key information not contained in signature
- Certificate-based signature in external file, key information contained in signature
- Certificate-based signature in external file, key information not contained in signature
- XML file contains password-based signature
- Password-based signature in external file

XML file contains certificate-based signature, key information included in signature

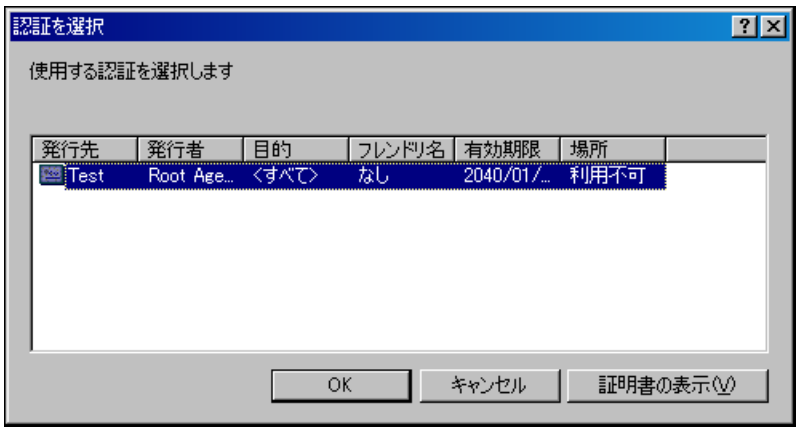
To verify the XML signature in this scenario, make the XML file active in XMLSpy. On clicking the **XML | Verify XML Signature** command, the verification process will be executed and the result will be displayed in the Messages window (verification succeeded or failed).

XML file contains certificate-based signature, key information not contained in signature

If no key information is contained in the certificate-based signature, XMLSpy will prompt you for the certificate from which public-key information for the verification can be read. Verification is done with the XML file active in XMLSpy. On clicking the **XML | Verify XML Signature** command, you will be prompted to select the certificate store in which the certificate is stored (*screenshot below*).



On selecting a certificate store and clicking **OK**, a dialog displaying the certificates in that store pops up (*screenshot below*). Select the certificate required for the verification and click **OK**.



The signature is verified and the result is displayed in the Messages window.

Certificate-based signature in external file, key information contained in signature

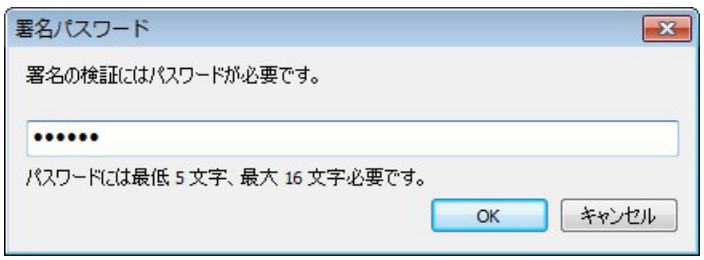
If a certificate-based XML signature is in an external file, the signature is verified with the signature file active in XMLSpy. On clicking the **XML | Verify XML Signature** command, the verification process will be executed and the result will be displayed in the Messages window (verification succeeded or failed).

Certificate-based signature in external file, key information not contained in signature

If a certificate-based XML signature is in an external file, the signature is verified with the signature file active in XMLSpy. On clicking the **XML | Verify XML Signature** command, XMLSpy will prompt you for the certificate from which public-key information for the verification can be read. Select the certificate as described in the section: *XML file contains certificate-based signature, key information not contained in signature*. The verification process will be executed and the result will be displayed in the Messages window (verification succeeded or failed).

XML file contains password-based signature

If the XML file contains a password-based XML signature, the signature is verified with the XML file active in XMLSpy. On clicking the **XML | Verify XML Signature** command, a dialog pops up prompting you for the password (*screenshot below*).

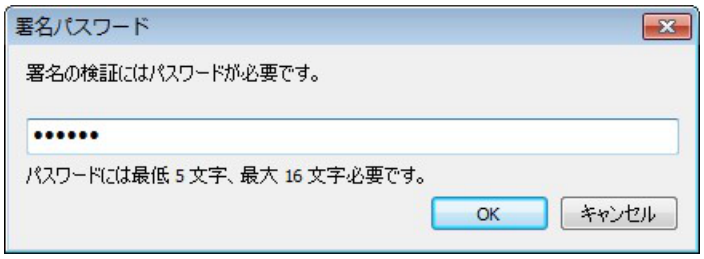


Enter the password, which must be five to sixteen characters long, and then click **OK**. The verification process will be executed and the result will be displayed in the Messages window

(verification succeeded or failed).

Password-based signature in external file

If a password-based XML signature is in an external file, the signature is verified with the signature file active in XMLSpy. On clicking the **XML | Verify XML Signature** command, a dialog pops up prompting you for the password (*screenshot below*).



Enter the password, which must be five to sixteen characters long, and then click **OK**. The verification process will be executed and the result will be displayed in the Messages window (verification succeeded or failed).

26.5 DTD/ Schema Menu

The **DTD/Schema** menu (*screenshot below*) contains commands to work with DTDs and XML Schemas.

26.5.1 Assign DTD

The **DTD/Schema | Assign DTD...** command is enabled when an XML file is active. It assigns a DTD to an XML document, thus allowing the document to be validated and enabling intelligent editing for the document. The command opens the Assign File dialog to let you specify the DTD file you wish to assign. You can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button). Note that you can make the path of the assigned DTD file relative by clicking the Make Path Relative To... check box. When you are done, your XML document will contain a DOCTYPE declaration that references the assigned DTD. The DOCTYPE declaration will look something like this:

```
<!DOCTYPE main SYSTEM "http://altova.com/spyweb.dtd">
```

Please note: A DTD can be assigned to a new XML file at the time the file is created.

26.5.2 Assign Schema

The **DTD/Schema | Assign Schema...** command is enabled when an XML document is active. It assigns an XML Schema to an XML document, thus allowing the document to be validated and enabling intelligent editing for the document. The command opens the Assign File dialog to let you specify the XML Schema file you wish to assign. You can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button). Note that you can make the path of the assigned file relative by clicking the Make Path Relative To... check box. When you are done, your XML document will contain an XML Schema assignment with the required namespaces. The schema assignment will look something like this:

```
xmlns="http://www.xmlspy.com/schemas/icon/orgchart"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xsi:schemaLocation="http://www.xmlspy.com/schemas/icon/orgchart http://
    schema.xmlspy.com/schemas/icon/orgchart.xsd"
```

26.5.3 Include Another DTD

The **DTD/Schema | Include another DTD...** command allows you to include another Document Type Definition (DTD) or external parsed entity into the internal subset of a document type definition, or in any DTD document. This is done by defining a corresponding external parsed entity declaration and using that entity in the following line:

```
<!ENTITY % navigation.dtd SYSTEM "S:\xml\navigation.dtd">
%navigation.dtd;
```

The command opens the Assign File dialog to let you specify the DTD file you want to include in your DTD.

Please note: This command is enabled in Grid View only.

26.5.4 Go to DTD

The **DTD/Schema | Go to DTD** command opens the DTD on which the active XML document is based. If no DTD is assigned, then an error message is displayed.

26.5.5 Go to Schema

The **DTD/Schema | Go to Schema** command opens the XML Schema on which the active XML document is based. If no XML Schema is assigned, then an error message is displayed.

26.5.6 Go to Definition

The **DTD/Schema | Go to Definition** command displays the exact definition of an element or attribute in the corresponding Document Type Definition or Schema document.

To see the item definition in Grid View

1. Click left on the item.
2. Select the menu item **DTD/Schema | Go to Definition,** or click on the icon.

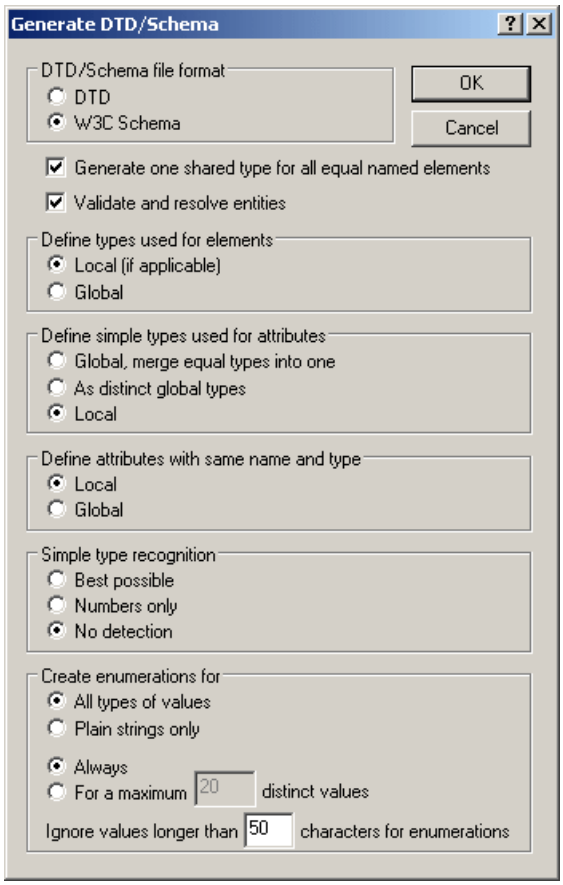
To see the item definition in Schema View

- Use CTRL + Double click on the item you want to see the definition of, or
- Click the item and select menu option **DTD/Schema | Go to Definition,** or click on the icon.

  In both cases, the corresponding DTD or Schema file is opened, and the item definition is highlighted.

26.5.7 Generate DTD/ Schema

The **DTD/Schema | Generate DTD/Schema** command generates a new DTD or W3C XML Schema from an XML document (or from a set of XML documents contained in a folder in the project window). This command is useful when you want to generate a DTD or XML Schema from XML documents.



If you generate an XML Schema, the following options are available:

- ***Elements***: The type of elements can be defined locally or globally (*Define types for elements*). If elements have the same name, a common type can be declared for use in the definition of these elements (*Generate one shared type*).
- ***Atributes***: The simple types of attributes (*Define simple types for attributes*) can be defined as (i) common global types; (ii) distinct global types; (iii) local types. Attributes with the same name and type can be defined either locally or globally.
- ***Simple type recognition***: The recognition of types (*Simple type recognition*) can be set to: (i) best possible; (ii) recognition of number datatypes only; (iii) no datatype recognition, in which case all datatypes are set to `xs:string`.
- ***Entity resolution***: In the XML document, entities may appear in element content and attribute values. Whether they are resolved or not (*Validate and resolve entities*) is therefore significant for enumeration values. Furthermore, some entities (especially parsed entities that contain markup) can affect the content model differently depending on whether they are resolved or not. Note that the XML document will be validated for being correct XML before the schema is generated. If the document is invalid, the schema generation process will be discontinued.
- ***Enumerations***: All types of values, or string values only, can be enumerated.

If you generate a DTD, the entity resolution and enumeration options are available.

The Generate DTD/Schema command normally operates on the active main window, but you can also use the **Generate DTD/Schema** command on any file, folder, or group of files in the active project window.

26.5.8 Flatten DTD

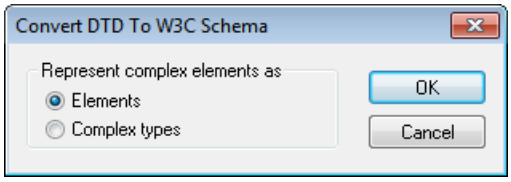
The **Flatten DTD** command is enabled when a DTD is the active document. It creates a new flat DTD, removing parameter entities and producing a single DTD from a collection of modules. It also suppresses sections marked `IGNORE` and deletes unused parameter entities.

The command pops up a Save dialog, in which you select a location at which to save the generated DTD file. Click **Save** to carry out the conversion. The flattened DTD file is generated and opened in XMLSpy.

26.5.9 Convert DTD to Schema

The **Convert DTD to Schema** command is enabled when a DTD is the active document. It converts a DTD into an XML Schema document (XSD).

The command pops up the Convert DTD to W3C Schema dialog (*screenshot below*), in which you can select whether complex elements should be converted into elements or complex types. On clicking **OK**, you are prompted to select a location at which to save the generated XSD file. Click **Save** to carry out the conversion. The XSD file is generated and opened in XMLSpy.



When you convert a DTD to XML Schema, XMLSpy makes a few assumptions because of the limited information available. Most notably, the values of certain DTD components are treated

literally rather than having their semantics parsed. This is because the program cannot know which of several possible usages is intended. In these cases, you should modify the generated conversion.

In any case, you should carefully examine the generated conversion to see if you can enhance it. A few areas in which improvements may be required are listed below.

Attribute Datatyping
DTDs allow for only 10 attribute datatypes, whereas XML Schemas, for instance, allow for more than 40 datatypes plus derived datatypes. You may wish to enhance a generated XML Schema, for example, by using a more restrictive datatype. Note that when an XML Schema is converted to DTD datatype information will be lost.

Namespaces
DTDs are not namespace-aware. As a result, if namespaces are to be specified in a DTD they must be hard-coded into element and attribute names. This could pose challenging problems when converting from one schema to another.

Entities
XML Schema does not have equivalents for the general entity declarations of DTDs. When XMLSpy converts a DTD to an XML Schema, it ignores entity declarations.

Unparsed data declarations
DTDs and XML Schemas use different mechanisms for handling unparsed data. This is explained in more detail below.

DTDs use the following mechanism:

- A notation is declared consisting of a name and an identifier, for example:
  ```
  <!NOTATION gif SYSTEM "image/gif">
  ```
- You declare the entity, for example:
  ```
  <!ENTITY cover_img SYSTEM "graphics/cover_img.gif" NDATA gif>
  ```
- Typically, you specify an attribute type of `ENTITY` on the relevant attribute, for example:
  ```
  <!ELEMENT img EMPTY>
    <!ATTLIST img format ENTITY #REQUIRED>
  ```

In XML Schema, the corresponding mechanism is as follows:

- Declare a notation. This functions in the same way as for the DTD.
  ```
  <xs:notation name="gif" public="image/gif"/>
  ```
  Note that the `public` attribute is mandatory and holds the identifier. An optional `system` attribute holds the system identifier and is usually an executable that can deal with resources of the notation type.
- You associate the `notation` declaration with a given attribute value using the `NOTATION` datatype. You cannot, however, use the `NOTATION` datatype directly, but must derive another datatype from the `NOTATION` datatype.
  ```
  <xs:simpleType name="formatType">
     <xs:restriction base="xs:NOTATION">
        <xs:enumeration value="gif"/>
        <xs:enumeration value="jpeg"/>
     </xs:restriction>
  </xs:simpleType>
  ```
- You associate the attribute with the datatype derived from the `NOTATION` datatype, e.g.

```
<xs:complexType name="imgType">
  <xs:attribute name="height"/>
  <xs:attribute name="width"/>
  <xs:attribute name="location"/>
  <xs:attribute name="format" type="formatType" use="required"/>
</xs:complexType>
<xs:element name="img" type="imgType"/>
```

When you convert a DTD to an XML Schema, XMLSpy does the following:

- Something like
  `<!ATTLIST image format ENTITY #REQUIRED ...>`
  is converted to
  `<xs:attribute name="format" type="xs:ENTITY" use="required"/>`

- And something like
  `<!NOTATION gif SYSTEM "image/gif">`
  is converted to
  `<xs:notation name="gif" system="image/gif"/>`

You should therefore make the following modifications:

1. In notations like `<xs:notation name="gif" system="image/gif"/>` replace `system` with `public`, and add an optional system identifier if required.
2. Derive a datatype from the `NOTATION` datatype as described above for `formatType`.
3. Associate the derived datatype with the relevant attribute.

Note: According to the XML Schema specification, you do not need to—or cannot, depending on your viewpoint—declare an external entity.

26.5.10 Flatten Schema

The **Flatten Schema** command is enabled when an XML Schema is the active document. It generates a new flat XSD by (i) adding the components of all included schemas as global components of the active schema, and (ii) deleting the included schemas.

The command redirects to the Flatten Schema command of the Schema Design menu. Since the Flatten Schema command is available in Schema View only, you will be prompted about whether you wish to switch to Schema View or not. For more information, see the Flatten Schema command.

26.5.11 Convert Schema to DTD

The **Convert Schema to DTD** command is enabled when an XML Schema is the active document. It converts an XML Schema document (XSD) into a DTD.

The command pops up a Save dialog, in which you select a location at which to save the generated DTD file. Click **Save** to carry out the conversion. The DTD file is generated and opened in XMLSpy.

Note the following points:

1. When you convert an XML Schema to a DTD, the namespace prefixes used in the XML Schema—not the namespace URIs or the namespace declarations—are carried through to the names of the corresponding elements and attributes in the DTD.
2. Since XML parsers ignore namespaces when validating an XML document against a DTD,

the namespace declarations themselves are not converted.

3. The `elementFormDefault` and `attributeFormDefault` attributes of the `xs:schema` element determine what elements and attributes have their prefixes included in the conversion process. If set to unqualified, then only globally declared elements and attributes, respectively, include prefixes in the conversion. If set to qualified, all element and attribute names have their prefixes included in the conversion.
4. Prefixes are converted to their corresponding string value plus a colon. Elements and attributes in default namespaces are converted to elements and attributes with names that begin with the string: `default_NS_X`, where X is an integer (starting with `1` and having a maximum value equal to the number of default namespaces used in the XML Schema).
5. In the DTD, element names are composed of parameter entities. This enables you to easily change the prefix in the DTD should the prefix in the XML document ever need to change. Parameter entity definitions can be changed either in the DTD document itself or by overriding the parameter entity definitions in the XML document's internal DTD subset.

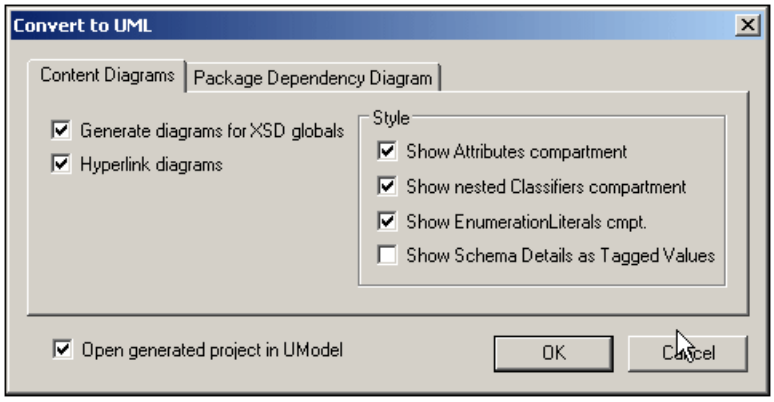
Note: Namespaces have no semantic value in DTDs, and namespace prefixes carried over from the XML Schema become merely a lexical part of the name of the element or attribute defined in the DTD.

26.5.12 Convert to UML

The **DTD/Schema | Convert to UML** command converts a W3C XML Schema to an Altova UModel Project (`.ump`) document (hereafter UModel project). UMP is the native format of Altova UModel, Altova's UML modeling application. UMP files can then be viewed and edited in Altova UModel.

To convert a schema to UML, do the following:

1. With the schema open, click the **Convert to UML** command. This pops up the Convert to UML dialog (*screenshot below*).



2. In the Content Diagrams tab, select the option Generate Diagrams for XSD Globals. This will generate, in the UModel project, a content model diagram for each global component.
3. Select the required options from those available in the dialog. These options are explained below.
4. If you wish to view the created project in UModel immediately, select the option to open the project in UModel. Otherwise leave this option unselected.
5. Click **OK**.
6. In the Save As dialog that appears, browse for the destination folder, then enter the name of the UMP file, and click **Save**.

Convert to UML options

The following options are available in the Convert to UML dialog.

In the ***Content Diagrams*** tab:

- *Hyperlink diagrams* creates in each diagram a link to the entry of that global component in the Model Tree view, thus enabling the component to be quickly located in the schema hierarchy.
- In the *Style* pane, the show compartments options enables various compartments to be either shown or hidden.

In the ***Package Dependency Diagram*** tab:

- The *Generate Diagram* option determines whether a package dependency diagram is generated. A package dependency diagram provides an overview of the entire package, showing the relationships of package components to one another. Note that the other options in this tab will be enabled only if the Generate Diagram option is selected.
- Selecting the *Hyperlink Package to Diagram* option creates a link from the package diagram to the Model Tree View.
- Four options are available for the layout of the package dependencies diagram: (i) unorganized layout (Autolayout option unselected); (ii) hierarchical layout (Autolayout and Hierarchical options selected); iii) block (Autolayout and Block options selected); and (iv) evenly spaced (Autolayout and Force Directed options selected). The layout can be modified by editing the diagram in UModel.

Note: The Convert to UML feature supports W3C XML Schemas only.

26.5.13 Generate XML from DB, Excel, EDI with MapForce

The **DTD/Schema | Generate XML from DB, Excel, EDI with MapForce** command launches Altova's MapForce if the application is installed. MapForce enables you to map a schema to another DTD, XML Schema, or database and to generate XML.

26.5.14 Design HTML/ PDF/ Word Output with StyleVision...

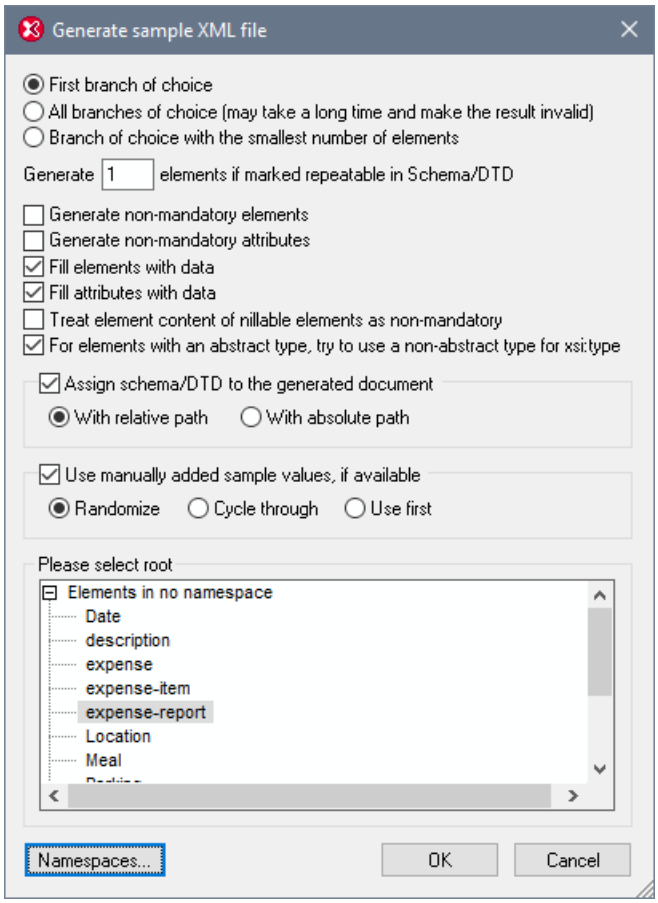
The **DTD/Schema | Design HTML/PDF Output in StyleVision...** command launches Altova's StyleVision if the application is installed. StyleVision enables you to design stylesheets for HTML, PDF, and RTF output.

26.5.15 Generate Sample XML/ JSON File

The **Generate Sample XML/JSON File** command is enabled in Text View, Grid View, and Schema View, and generates an XML or JSON instance based on the currently active schema file. If the currently active file is a DTD or XML Schema, then an XML instance file is generated. If the currently active file is a JSON schema, then a JSON instance file is generated.

Generating sample XML files

With a DTD or XML Schema active, you can generate a sample XML instance based on the schema. On clicking the **Generate Sample XML/JSON File** command, the Generate Sample XML File dialog (*screenshot below*) appears, in which you can specify the options for the sample generation.



Elements of choice groups

A choice group is a group of elements from which one may be used. For example, if an element called `items` is defined as having a `choice` group consisting of the three elements: `cd`, `dvd`, `book`, then `items` can validly have any one of these three elements as a child element (with a maximum number of occurrences as specified in that element's `maxOccurs` attribute).

In the Generate Sample XML File dialog, you can select whether (i) the first branch (element) of the `choice` group, (ii) all branches, or (iii) the branch with the smallest number of descendant elements is generated. Note that the *All branches* selection could generate an invalid document since only one branch from a `choice` group is allowed.

If any of the `choice` groups's branches are repeatable (that is, it has a `maxOccurs` value of greater than `1`), then you specify, in the first text box of the dialog, how many of the repeatable elements to generate, up to a maximum of `99`. If the `maxOccurs` attribute of the `choice` group is defined as `unbounded` or as a large number and *All branches* is selected in the Generate Sample XML File dialog, then the `maxOccurs` of the `choice` group is also limited by the number of repeatable elements you specify in the first text box of the dialog.

Generate non-mandatory elements

Activating this option generates both the mandatory and non-mandatory elements defined in the schema.

Generate non-mandatory attributes
Activating this option generates both the mandatory and non-mandatory attributes defined in the schema.

Generate X elements if marked repeatable in Schema/DTD
Activating this option generates the number of repeatable elements you enter in the text box. This applies to all elements, including those in `choice` groups.

Fill elements and attributes with data
Activating this option inserts the datatype values of the respective elements and attributes. For example if an element is defined as being pf datatype `string`, then the element is given a dummy value of `string`.

Nillable elements and abstract types
The contents of nillable elements can be treated as non-mandatory, and elements with an abstract type can use a non-abstract type for its `xsi:type` attribute.

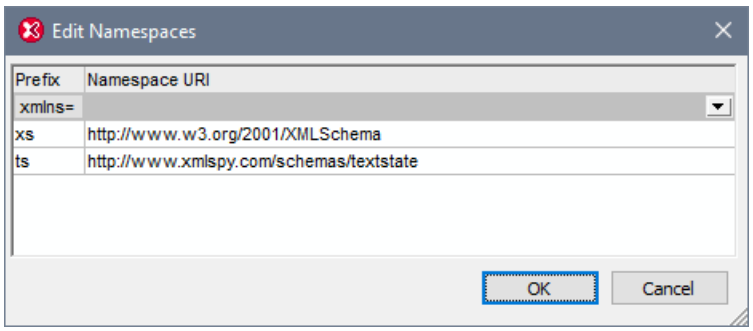
Schema assignment for the generated XML file
The schema used to generate the XML file can be assigned to the generated XML file with a relative or absolute path.

Use manually added sample values if available
If the schema component has sample values assigned to it, then these will be used as the value or content of that component. For individual components, sample values are assigned in the Facets Entry Helper, in the Samples tab. Which value from the available sample values is selected for a single file generation can be specified:

- A random selection.
- Each sample value in turn for each instance of the component. For each file generation, the cycle starts anew.
- The first value always.

Root element
If the schema contains more than one global element, these are listed, and the root element required for the sample XML file can be selected from the list.

Namespaces
Click the **Namespaces** button to open the Edit Namespaces dialog (*screenshot below*), The namespaces that are defined in the schema, plus any standard XML Schema namespaces that are required in the sample XML file, will appear in this dialog.

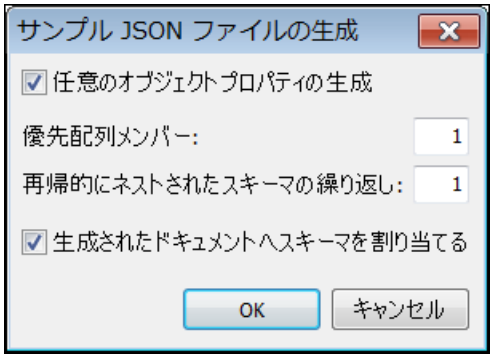


You can edit the following:

- The namespace prefix that is bound to any of the document's namespaces. The namespace prefixes that are set in this dialog will be used (in the generated XML file) to prefix nodes that are in the corresponding namespace. For example, the screenshot indicates that nodes in the `http://www.xmlspy.com/schemas/textstate` namespace will be prefixed with `ts:` in the sample file .
- You can set one of the document's namespaces to be the default namespace (`xmlns=`) by selecting, in the `xmlns=` combo box, the namespace that you want. Nodes in the namespace that is selected as the default namespace will then be generated without a namespace prefix.

Generating sample JSON files

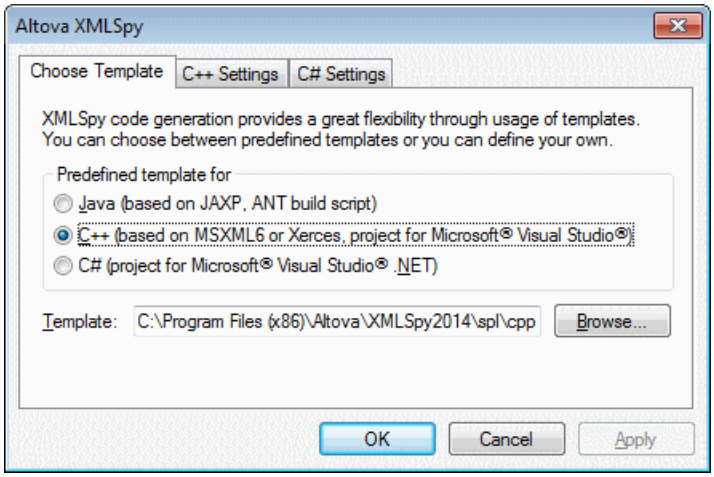
With a JSON schema active, you can generate a sample JSON instance based on the JSON schema. On clicking the command, the Generate Sample JSON File dialog (*screenshot below*) appears, in which you can specify the options for the sample generation.



You can choose whether to generate non-mandatory object properties, the length of arrays, and repetitions of recursive definitions. You can also specify whether the active JSON schema should be automatically assigned to the generated JSON sample file or not.

26.5.16 Generate Program Code

The **DTD/Schema | Generate Program Code...** command automatically generates Java, C++, or C# class files from definitions in the active schema (DTD or XML Schema) document.



Please see the Code Generator section of this documentation for details about code generation.

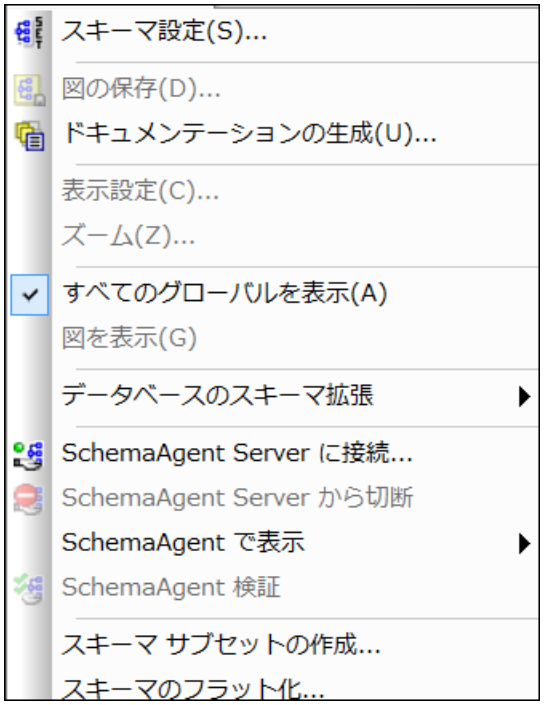
26.5.17 Flush Memory Cache

The **DTD/Schema | Flush Memory Cache** command flushes all cached schema (DTD and XML Schema) documents from memory. To speed up validation and intelligent editing, XMLSpy caches recently used schema documents and external parsed entities in memory. Information from these cached documents is also displayed when the Go to Definition command is invoked.

Flush the memory cache if memory is tight on your system, or if you have used documents based on different schemas recently.

26.6 Schema Design Menu

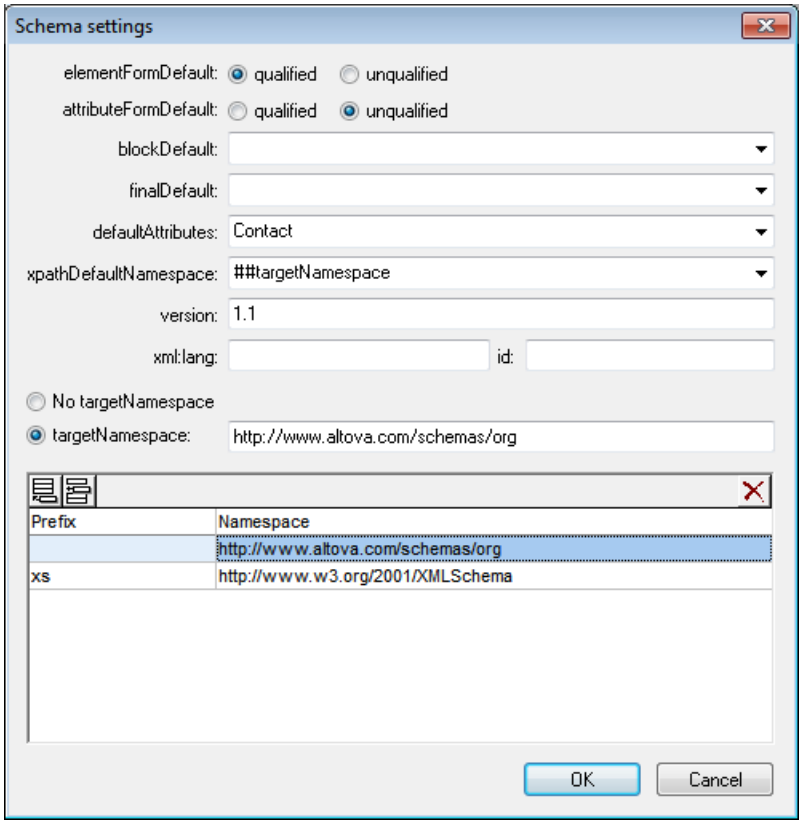
The **Schema Design** menu enables you to configure the Schema View of XMLSpy. This view enables you to design XML Schemas in a GUI. It is available when an XML Schema document is active in Schema View.



The commands available in this menu are described in this section.

26.6.1 Schema Settings

The **Schema Design** | **Schema Settings** command is enabled in Schema View and lets you define global settings for the active schema. These settings are the attributes of the `xs:schema` element.



The settings defined in the Schema Settings dialog above (when XSD mode is set to 1.1) will create the following `xs:schema` element.

```
<xs:schema xmlns="http://www.altova.com/schemas/org"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.altova.com/schemas/org"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    xpathDefaultNamespace="##targetNamespace"
    version="1.1"
    defaultAttributes="Contact">
```

Note: What's in the Schema Settings dialog will differ according to the active XSD mode. When XSD 1.0 is the active mode, XSD 1.1 attributes are not present in the dialog.

The `defaultAttributes` and `xpathDefaultNamespace` attributes are XML Schema 1.1 features and will be available only in XSD 1.1 mode. The other attributes are available in both XSD 1.0 and XSD 1.1.

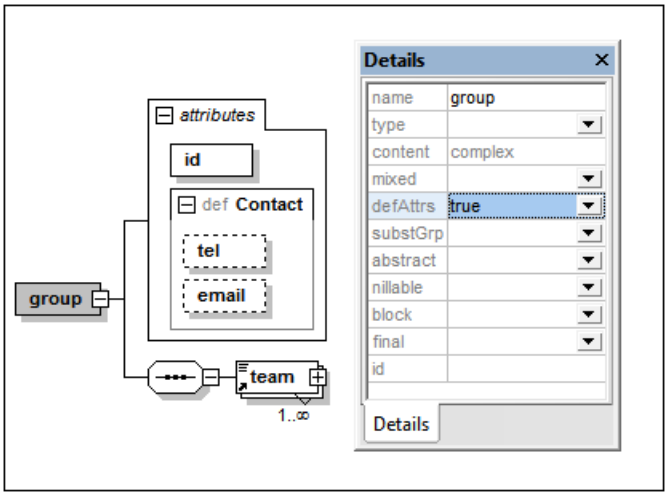
The `version` attribute

The version attribute is the document version. It is not the XSD version of the document.

The `defaultAttributes` attribute

The **`defaultAttributes`** attribute enables you to select an attribute group as the default attribute group of all complex types in the schema. The default attribute group is displayed in the content

model of these complex types. In the screenshot below, the `Contact` attribute group is the default attribute group (*also see screenshot above, where this has been set*), and is automatically available on the `group` element. To disable the attribute group, set the complex type's `defaultAttributesApply` to `false`. You can do this via the *defAttrs* property in the Details entry helper of the complex type (*see screenshot below*).



The `xpathDefaultNamespace` attribute

The `xpathDefaultNamespace` attribute sets the default namespace for elements in XPath expressions used in the schema. If set in the Schema Settings dialog, the attribute is applied to the top-level `xs:schema` element. So the scope of the declaration will be the entire document. You can override the declaration on `xs:schema` with declarations on elements where the attribute is allowed:

- `xs:assert` and `xs:assertion`
- `xs:alternative`
- `xs:selector` and `xs:field` (in identity constraints)

You can change the XPath default namespace in the Details entry helper of the elements listed above.

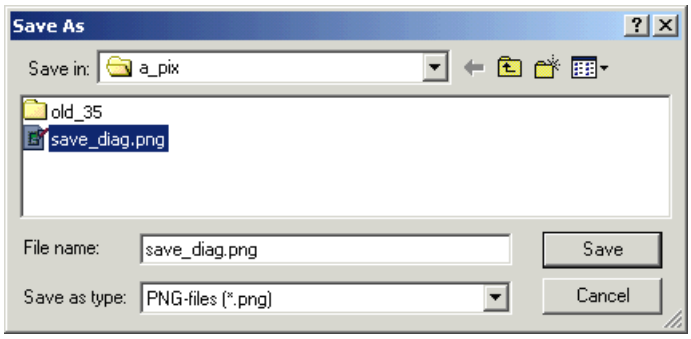
The `xpathDefaultNamespace` attribute can have one of three allowed values:

- `##targetNamespace`: The XPath default namespace will be the same as the target namespace of the schema
- `##defaultNamespace`: The XPath default namespace will be the same as the default namespace of the schema
- `##local`: There is no XPath default namespace

If no XPath default namespace is declared in the document, unprefixed elements in XPath expressions will be in no namespace. The XPath default namespace declaration does not apply to attributes.

26.6.2 Save Diagram

The **Schema Design | Save Diagram...** command saves the diagram of the Content Model currently displayed in the Main Window in PNG format to any desired location.



26.6.3 Generate Documentation

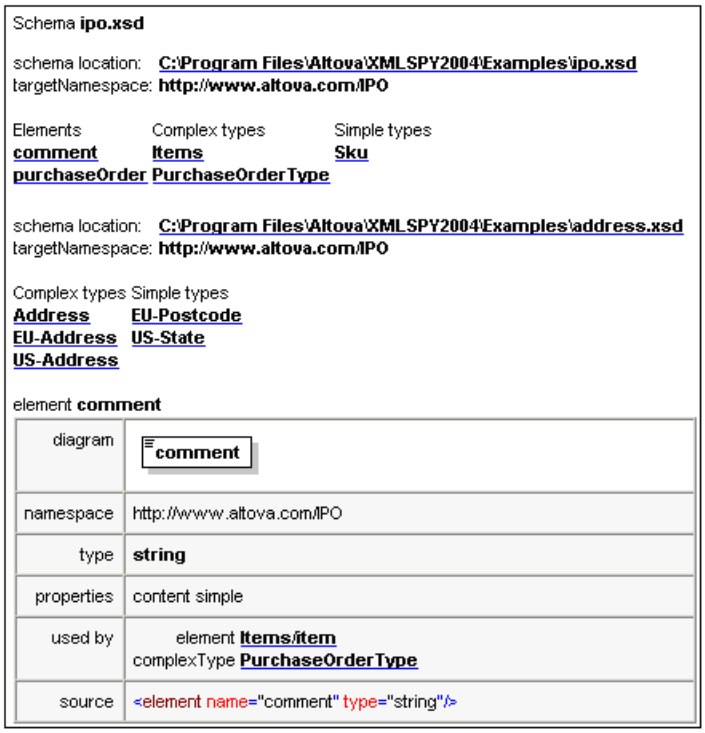
The **Schema Design | Generate Documentation** command generates detailed documentation about your XML or JSON schema (*see screenshot below*) in HTML, MS Word, RTF or PDF. The documentation generated by this command can be freely altered and used; permission from Altova to do so is not required. Documentation is generated for components you select in the (JSON) Schema Documentation dialog (which appears when you select the **Generate Documentation** command). Related elements (child elements, complex types, etc.) are typically hyperlinked in the onscreen output, enabling you to navigate from component to component. Components with a content model also have links to the content model definitions. Note that schema documentation is also generated for **included and imported schema components**. The various documentation-generation options for XML Schema are described in the section Documentation Options. JSON schema documentation options are described in the section Generating JSON Schema Documentation.

Note that the Documentation Options are applied on top of the settings you specify in the Schema Display Configuration dialog.

Note: In order to generate documentation in MS Word format, you must have MS Word (version 2000 or later) installed.

You can either use XMLSpy's fixed standard design for the generated document, or you can use a StyleVision SPS for the design. Using a StyleVision SPS enables you to customize the design of the generated documentation as well as to generate PDF as an additional output format. How to work with an SPS is explained in the section, User-Defined Design.

Note: In order to use an SPS to generate schema documentation, you must have StyleVision installed on your machine.



Schema **ipo.xsd**

schema location: **C:\Program Files\Altova\XMLSPY2004\Examples\ipo.xsd**
targetNamespace: **http://www.altova.com/IPO**

Elements	Complex types	Simple types
comment	**Items**	**Sku**
purchaseOrder	**PurchaseOrderType**	

schema location: **C:\Program Files\Altova\XMLSPY2004\Examples\address.xsd**
targetNamespace: **http://www.altova.com/IPO**

Complex types	Simple types
Address	**EU-Postcode**
EU-Address	**US-State**
US-Address	

element **comment**

diagram	comment
namespace	http://www.altova.com/IPO
type	**string**
properties	content simple
used by	element **Items/item** complexType **PurchaseOrderType**
source	<element name="comment" type="string"/>

The screenshot above shows generated schema documentation with an index (all related schemas with their global components organized by component type) at the top of the document.

Note: When generating documentation for W3C schema documents, XMLSpy uses application-internal versions of these documents. Consequently, other locations of these documents are not considered, and redefinitions and other schema modifications will not be reflected in the documentation.

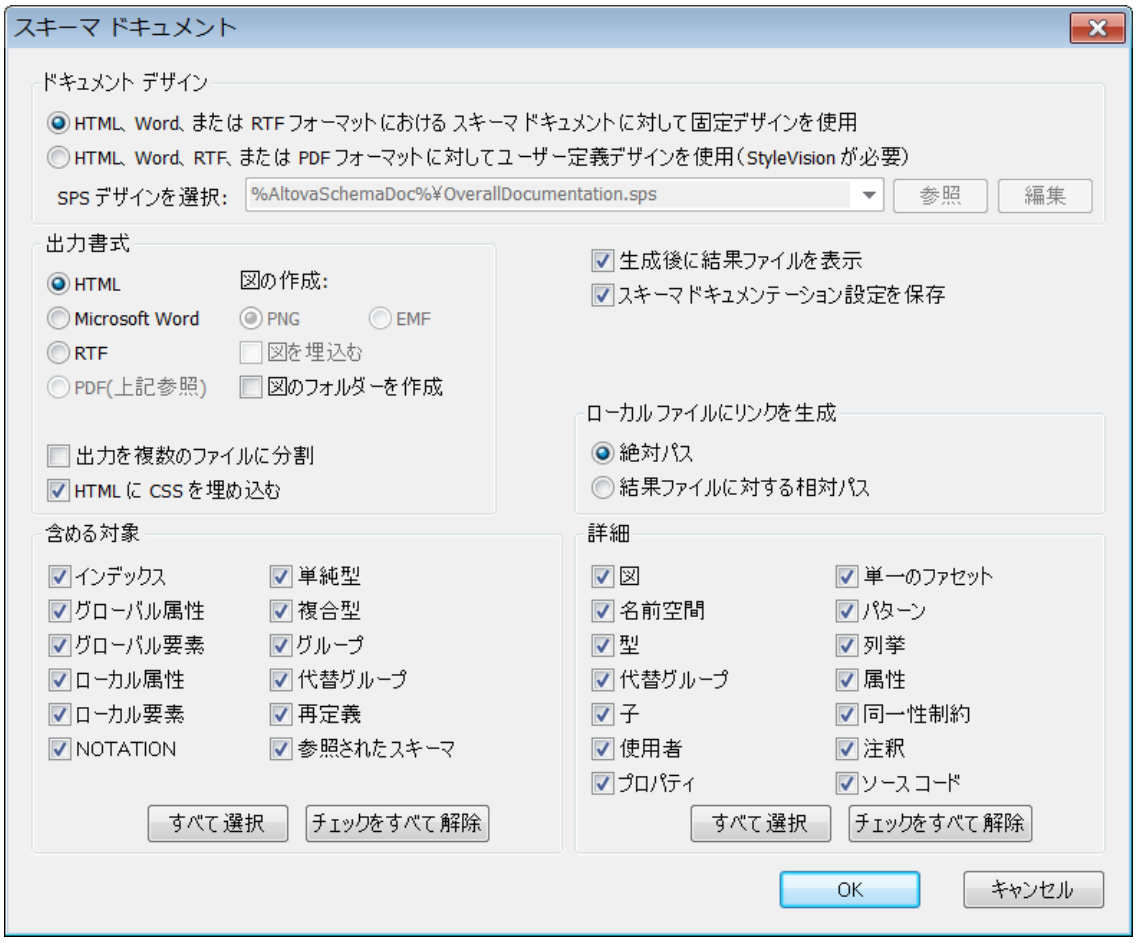
Documentation Options

The **Schema Design | Generate Documentation** command generates detailed documentation of the active schema: XML Schema or JSON schema. This section describes the generation of XML Schema documentation. The procedure for generating JSON schema documentation is similar. For details about generating JSON schema documentation and a description of documentation generation settings, see the section Generating JSON Schema Documentation.

Generating XML Schema documentation

If an XML Schema document is active and you click the **Generate Documentation** command, the Schema Documentation dialog (*screenshot below*) is displayed. In this dialog, you can select options for the documentation.

In the Documentation Design pane of the dialog you can select whether to use the fixed XMLSpy design for the generated documentation or whether to use a customized design created in a StyleVision SPS. Select the option you want. Note that PDF output is available only for documentation generated with a StyleVision SPS, not for documentation generated using a fixed design. How to work with a user-defined design is described in the section, User-Defined Design.



The other options in the Schema Documentation dialog are explained below:

- The required format is specified in the Output Format pane: either HTML, Microsoft Word, RTF, or PDF. (The PDF output format is only available if you use a StyleVision SPS to generate the documentation.) On clicking **OK**, you will be prompted for the name of the output file and the location to which it should be saved.
- Microsoft Word documents are created with the `.doc` file extension when generated using a fixed design, and with a `.docx` file extension when generated using a StyleVision SPS.
- The documentation can be generated either as a single file or be split into multiple files. When multiple files are generated, each file corresponds to a component. What components are included in the output is specified using the check boxes in the Include pane. In fixed designs, links between multiple documents are created automatically.
- For HTML output, the CSS style definitions can be either saved in a separate CSS file or embedded in the HTML file (in the `<head>` element). If a separate CSS file is created, it will be given the same name as the HTML file, but will have a `.css` extension. Check or uncheck the *Embed CSS in HTML* check box to set the required option.
- The *Embed Diagrams* option is enabled for the MS Word, RTF, and PDF output options. When this option is checked, diagrams are embedded in the result file, either in PNG or EMF format. Otherwise diagrams are created as PNG or EMF files, which are displayed in the result file via object links.
- When the output is HTML, all diagrams are created aers document-external PNG files. If the *Create folder for diagrams* check box is checked, then a folder will be created in the same folder as the HTML file, and the PNG files will be saved inside it. This folder will have

a name of the format *HTMLFilename*_diagrams. If the *Create folder for diagrams* check box is unchecked, the PNG files will be saved in the same folder as the HTML file.

- Links to local files (such as diagram image files and external CSS file) can be relative or absolute. In the *Generate links to local files* pane, select the appropriate radio button according to the option you prefer.
- In the Include pane, you select which item types you want to include in the documentation. Each item of the selected types will be displayed in the generated documentation. For example, if *Local Attributes* is checked, then the description of each local attribute is displayed as a separate entry. The Index option lists all related schemas at the top of the file, with their global components organized by component type. The **Check All** and **Uncheck All** buttons enable you to quickly select or deselect all the options in the pane. Note that the *Include* option does not affect the display of an item type within the graphical definitions. That display is controlled by the settings you make in the Schema Display Configuration dialog. So if you wish to disable the display of attributes within the graphical representation of a schema item, then unchcek the Attributes option in the Schema Display Configuration dialog.
- The Details pane lists the details that may be included for each component. Select the details you wish to include in the documentation. The **Check All** and **Uncheck All** buttons enable you to quickly select or deselect all the options in the pane.
- The *Show Result File* option is enabled for all output options. When this option is checked, the result files are displayed in Browser View (HTML output), MS Word (MS Word output), and the default applications for .rtf files (RTF output) and .pdf files (PDF output).

Parameter values

If the StyleVision SPS contains one or more parameter definitions, then on clicking **OK**, a dialog pops up listing all the parameters defined in the SPS. You can enter parameter values in this dialog to override the default parameter values that were assigned in the SPS.

User- Defined Design

Instead of the fixed standard XMLSpy design, you can create a customized design for schema documentation. The customized design is created in a StyleVision SPS, which is a design template for the output document.

Creating the SPS

A StyleVision Power Stylesheet (or SPS) is created using Altova's StyleVision product. An SPS for generating schema documentation must be based on an XML Schema that specifies the structure of the schema documentation. This schema is called SchemaDocumentation.xsd, and it is delivered with your XMLSpy package. It is stored in the folder: C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Documentation\Schema.

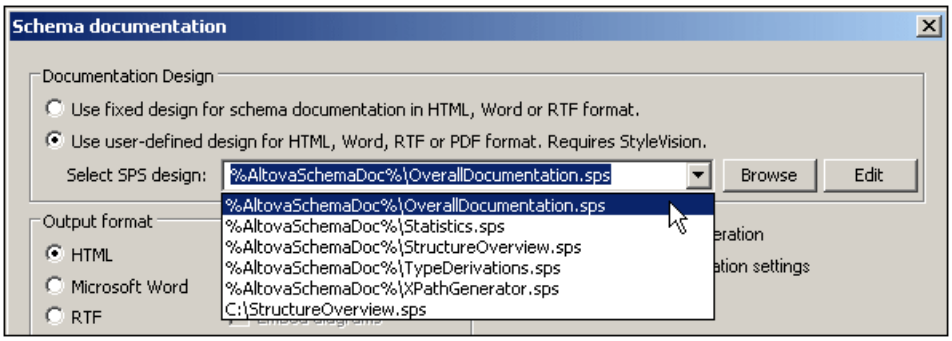
When creating the SPS design in StyleVision, nodes from the SchemaDocumentation.xsd schema are placed in the design template and assigned styles and properties. Additional components, like links, tables and images, can also be added to the SPS design. In this way, the entire output document can be designed in the SPS. How to create an SPS design in StyleVision is described in detail in the StyleVision user manual.

The advantage of using an SPS for generating schema documentation is that you have complete control over the schema documentation design. Note also that PDF output of the schema documentation is available only if a user-defined SPS is used; PDF output is not available if the fixed XMLSpy design is used.

Specifying the SPS to use for schema documentation

After an SPS has been created, it can be used to generate schema documentation. The SPS you wish to use for generating the schema documentation is selected in the Schema Documentation dialog (accessed via the **Schema Design | Generate Documentation** command). In the Documentation Design pane of this dialog (*see screenshot below*), select the *Use User-Defined Design* radio button. You can then click the **Browse** button and browse for the SPS you want. Click the dialog's **OK** button, and, in the Save dialog that pops up, select the folder for, and enter the name of, the output file.

Note: The SPS file must correctly locate the schema on which it is based: `SchemaDocumentation.xsd` (*see above*).



The following editable SPS designs for schema documentation generation are delivered with XMLSpy. They are in the (My Documents) folder: `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2011\Documentation\Schema\`. They are:

- `OverallDocumentation.sps`, which generates full documentation about the schema
- `Statistics.sps`, which lists the number of global and local elements, attributes and attribute groups, and simple and complex types for the main schema and for each schema file independently
- `StructureOverview.sps`, which outputs a structure of global elements and complex types up to a configurable depth
- `TypeDerivations.sps`, which lists simple and complex types and all their directly and indirectly derived types in the form of a tree
- `XPathGenerator.sps`, which generates all possible XPath statements up to a configurable depth

These files, together with other SPS files you have recently browsed for, will be available in the combo box of the *Use User-Defined* option (*see screenshot above*).

Clicking the **Edit** button in the Documentation Design pane launches StyleVision and opens the selected SPS in a StyleVision window. In order to preview the result document in StyleVision, you will need a Working XML file. A sample XML file for this purpose, called `OrgChart.xml`, is supplied with your application and is located in the (My Documents) folder:

`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Documentation\Schema\SampleData`

Note: In order to use an SPS to generate schema documentation, you must have StyleVision installed on your machine.

26.6.4 Configure View

The **Schema Design | Configure view** command is active in Content Model View and allows you to configure the Content Model View. Clicking the command opens the Schema Display Configuration dialog at the bottom right of the XMLSpy window, enabling you to see the effect of your settings as you enter them in the dialog. The settings take effect when you click the **OK** button of the dialog, and apply to the Content Model View of all XML Schema files that are opened subsequently. These settings also apply to the schema documentation output and printer output. For example, if you wish to disable the display of attributes within the graphical representation of a schema item in the schema documentation output, then uncheck the *Attributes* option in the Schema Display Configuration dialog (*screenshot below*).

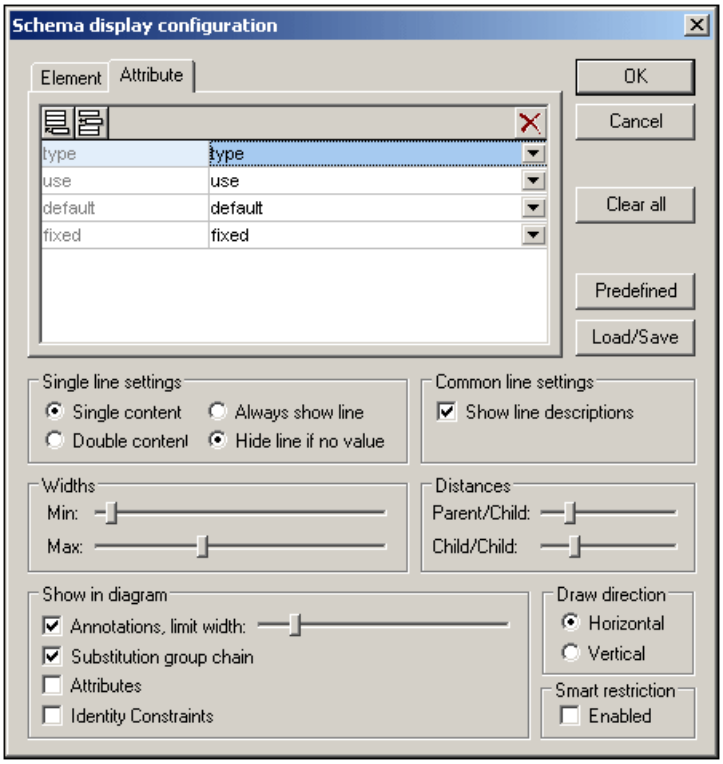
Note: For a description of how to configure JSON Schema Design View, see the section Configuring JSON Schema Design View.

Defining property descriptor lines for the content model

You can define what properties of elements and attributes are displayed in the Content Model View. These properties appear as grid lines in component boxes.

To define property descriptor lines:

1. Select **Schema Design | Configure view**. The Schema display configuration dialog appears.
2. In the **Element** or **Attribute** tab, click the Append or Insert icon to add a property descriptor line. The line is added in the dialog and to element boxes in the Content Model View.
3. From the combo box, select the property you want to display. *See screenshot.*
4. Repeat steps 2 and 3 for as many properties as required.



The Content Model View is updated, showing the defined property descriptor lines for all elements for which they exist.

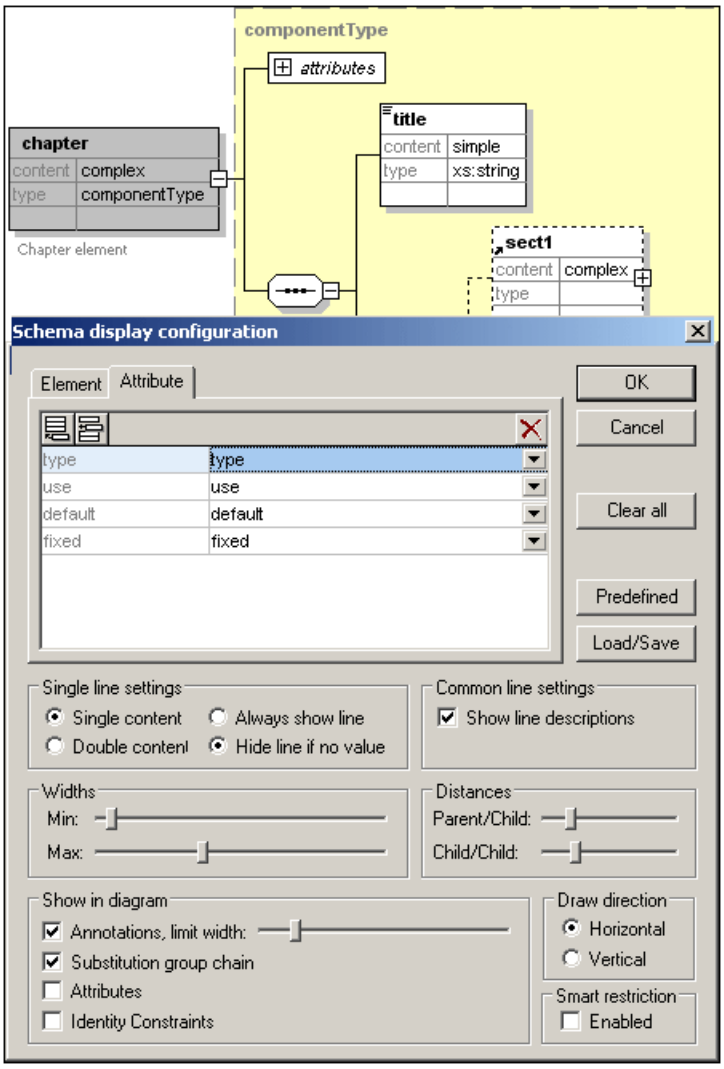
Note: For attributes, the configuration you define appears only when attributes are displayed in the diagram (as opposed to them being displayed in a pane below the Content Model View). The configured view applies to all Content Model Views opened after the configuration is defined.

Deleting a property descriptor line from the Content Model View

To delete individual property descriptor lines, in the Schema Display Configuration dialog, select the property descriptor line you want to delete, and click the Delete icon .

Settings for configuring the Content Model View

The Content Model View can be configured using settings in the Schema Display Configuration dialog. How to define what property descriptor lines are displayed in Content Model View has been described above. The other settings are described below.



Single line settings
You can define whether a property descriptor line is to contain single or double content, and whether individual lines must appear for every element or only for elements that contain that property. Use the appropriate radio buttons to define your settings. Note that these two settings can be set for individual lines separately (select the required line and make the setting).

Common line settings
This option toggles the line descriptions (i.e. the name of the property) on and off.

Widths
These sliders enable you to set the minimum and maximum size of the element rectangles in Content Model View. Change the sizes if line descriptor text is not fully visible or if you want to standardize your display.

Distances
These sliders let you define the horizontal and vertical distances between various elements onscreen.

Show in diagram
The Annotations check box toggles the display of annotation text on or off, as well as the

annotation text width with the slider. You can also toggle the display of the substitution groups on or off. The Attributes and Identity Constraints appear in the Content Model diagram if their check boxes are selected; otherwise they appear as tabs in a pane at the bottom of the Content Model window.

Draw direction
These options define the orientation of the element tree on screen, horizontal or vertical.

Editing the content model in the diagram itself
You can change element properties directly in the content model diagram. To do this, double-click the property you wish to edit and start entering data. If a selection is available, a drop-down list appears, from which you can select an option. Otherwise, enter a value and confirm with **Enter**.

Buttons in the Schema display configuration dialog
This dialog has the following buttons:

- The **Load/Save** button allows you to load and save the settings you make here.
- The **Predefined** button, resets the display configuration to default values.
- The **Clear all** button empties the list box of all entries.

Enabling smart restrictions

To enable smart restrictions, check the Enable Schema Restrictions check box.

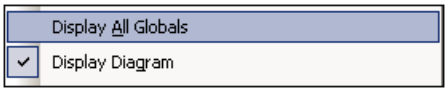
26.6.5 Zoom

The **Schema Design | Zoom** command controls the zoom factor of the Content Model View. This feature is useful if you have a large content model and wish to zoom out so that the entire content model fits in the Main Window. You can zoom between 10% and 200% of actual size.

To zoom in and out, either drag the slider or click in the entry box and enter a percentage value.

26.6.6 Display All Globals

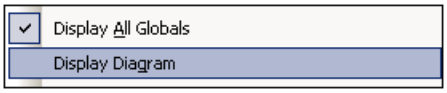
The **Schema Design | Display All Globals** command switches from Content Model View to Schema Overview to display all global components in the schema. It is a toggle with the Display Diagram command. The currently selected toggle is indicated with a check mark to its left (*see screenshot*).



Alternatively, you could use the **Display All Globals** icon at the top of the Content Model View to switch to the Schema Overview.

26.6.7 Display Diagram

The **Schema Design | Display Diagram** command switches to the Content Model View of the selected global component—if the selected component has a content model. Global components that have a content model (complex types, elements, and element groups) are indicated with the icon to its left. The Display Diagram command is a toggle with the Display All Globals command. The currently selected toggle is indicated with a check mark to its left (*screenshot below*).



Alternatively, you could use the following methods to switch to Content Model View:

- Click the icon next to the component, the content model of which you want to display.
- Double-click a component name in the Component Navigator Entry Helper (at top right).

26.6.8 Schema Extensions for Databases

This menu item pops out a sub-menu containing commands for Oracle and MS SQL Server schema extensions.

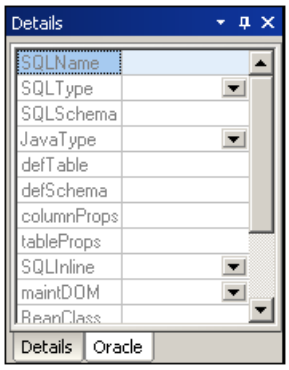
- Enable Oracle Schema Extensions
- Oracle Schema Settings
- Enable Microsoft SQL Server Schema Extensions
- Named Schema Relationships
- Unnamed Element Relationships

Enable Oracle Schema Extensions

XMLSpy provides support for Oracle schema extensions for use with Oracle 9i Project XDB. Using these schema extensions allows you to configure and customize how Oracle 9i Project XDB stores XML documents. These XML documents are then accessible through SQL queries and legacy tools. Please see the Oracle Website for more information.

When you select the **Enable Oracle Schema Extensions** command, the following occurs:

- The XDB namespace is declared on the `schema` element: `xmlns:xdb="http://xmlns.oracle.com/xdb"`.
- An Oracle tab is created in the Details Entry Helper, enabling you to add attributes—including XDB-specific attributes—to schema elements such as `xsd:complexType` and `xsd:element`.

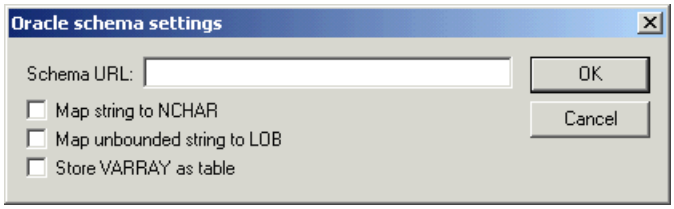


Oracle extensions can be defined for complex types, elements, and attributes. Use the Entry Helper as you normally would in XMLSpy.

Please note: This menu command can be toggled on and off, that is, extensions can be enabled or disabled. When Oracle extensions are enabled, the command is displayed with a check mark to its left. Disabling Oracle extensions (by clicking the enabled command) deletes the XDB namespace declaration and all XDB extensions in the file. A warning message appears since this action cannot be undone.

Oracle Schema Settings

The **Oracle Schema Settings** command allows you to define global settings for Oracle schema extensions.



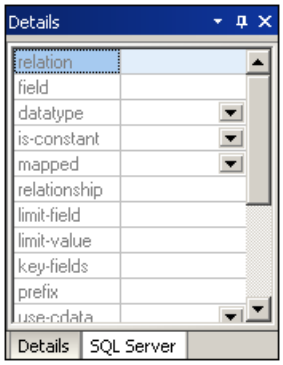
In order to access this dialog, Oracle schema extensions must be enabled (using the Enable Oracle Schema Extensions command).

Enable Microsoft SQL Server Schema Extensions

XMLSpy provides support for Microsoft SQL Server 2000 schema extensions for use with Microsoft SQL Server. Using these schema extensions allows you to configure and customize how Microsoft SQL Server stores XML documents. These XML documents are then accessible through SQL queries and legacy tools. Please see the Microsoft Website for more information.

When you select the **Enable Microsoft SQL Server Schema Extensions** command, the following occurs:

- The SQL Server namespace is declared on the `schema` element: `xmlns:sql="urn:schemas-microsoft-com:mapping-schema"`.
- An SQL Server tab is created in the Details Entry Helper, enabling you to add attributes to schema elements such as `xsd:element`.



Where SQL Server extensions can be defined for a schema component, the SQL Server tab is available in the Details Entry Helper when the component is selected. Use the Entry Helper as you normally would in XMLSpy.

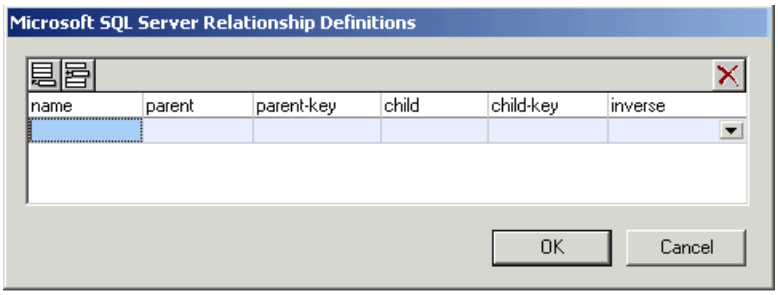
Please note: This menu command can be toggled on and off, that is, extensions can be enabled or disabled. When SQL Server extensions are enabled, the command is displayed with a check mark to its left. Disabling SQL Server extensions (by clicking the enabled command) deletes the SQL Server namespace declaration and all SQL extensions in the file. A warning message appears since this action cannot be undone.

Named Schema Relationships

The **Named Schema Relationships** command allows the definition of named relationships to provide the information needed to create the document hierarchy. You have to have previously enabled the SQL Server schema extensions, using the menu option "Enable SQL Server Schema Extensions", to be able to access this menu option.

To create a named schema relationship:

1. Click the insert or append icon, to add a new row to the dialog box.
2. Click the field and enter the corresponding relationship name.
3. Click **OK** to confirm.



This generates a SQL relationship element, placing it just after the namespace declaration.

Please note: Click the delete icon, to delete a row from the dialog box.

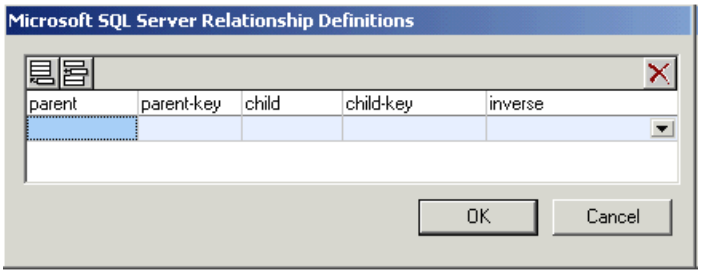
Unnamed Element Relationships

The **Unnamed Element Relationships** command allows the definition of unnamed relationships to provide the information needed to create the document hierarchy. You have to have previously enabled the SQL Server schema extensions, using the menu option **Enable Microsoft SQL**

Server Schema Extensions, to be able to access this menu option.

To create an unnamed schema relationship:

1. Click the insert or append icon, to add a new row to the dialog box.
2. Click the field and enter the corresponding name.
3. Click **OK** to confirm.

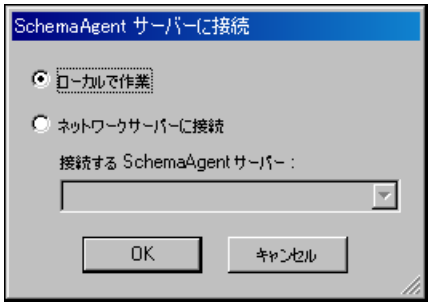


This generates a SQL relationship element for the currently selected schema element.

Please note: Click the delete icon, to delete a row from the dialog box.

26.6.9 Connect to SchemaAgent Server

The **Schema Design | Connect to SchemaAgent Server** command is enabled when an XML Schema document is active and it enables you to connect to a SchemaAgent Server. You are able to connect to a SchemaAgent server only if a licensed Altova SchemaAgent product is installed on your machine. When you click this command, the Connect to SchemaAgent Server dialog (*screenshot below*) opens:



You can use either the local server (the SchemaAgent server that is packaged with Altova SchemaAgent) or a network server (the Altova SchemaAgent Server product, which is available free of charge). If you select **Work Locally**, the local server of SchemaAgent will be started when you click **OK** and a connection with it will be established. If you select **Connect to Network Server**, the selected SchemaAgent Server must be running in order for a connection to be made.

When connected to SchemaAgent Server, XMLSpy acts as a SchemaAgent client, and provides powerful and enhanced schema editing and management functionality. For details about SchemaAgent, the installation of SchemaAgent Server, and how to connect to SchemaAgent Server, see SchemaAgent in the DTD and XML Schema section of this user manual. For more information about installing and working with these two products, see the SchemaAgent user manual that is delivered with these products.

After you connect to SchemaAgent Server, a message appears in the bar at the top of the Main Window with information about the connection. You now have full access to all schemas and schema components in the search path/s (folder/s) defined for the SchemaAgent server to which XMLSpy is connected.

Please note: In order for the connection to succeed, you must have Altova's SchemaAgent Client product installed with a valid license on the same machine as that on which XMLSpy is installed.

26.6.10 Disconnect from SchemaAgent Server

The **Disconnect from SchemaAgent Server** command is enabled when a connection to a SchemaAgent Server has been made successfully. Selecting this command disconnects XMLSpy from the SchemaAgent Server.

26.6.11 Show in SchemaAgent

The **Show in SchemaAgent** menu item causes the active schema and, optionally, linked schemas to be displayed in the Altova product SchemaAgent. (This product must be installed on the same machine as XMLSpy if you wish to use SchemaAgent functionality). The schema/s are opened in a new SchemaAgent Design in SchemaAgent.

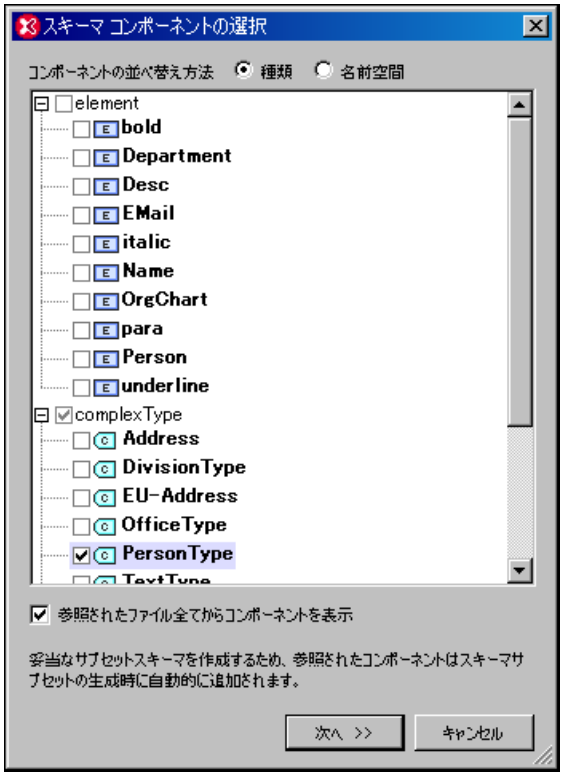
Mousing over the **Show in SchemaAgent** menu item pops out a submenu with options about what schemas to show in SchemaAgent. These options are described in SchemaAgent in the DTD and XML Schema section of this user manual.

26.6.12 SchemaAgent Validation

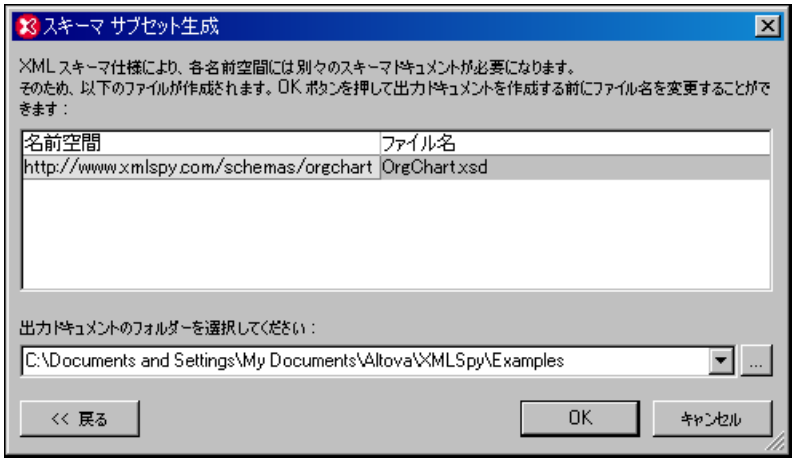
The **SchemaAgent Validation** command enables you to validate the currently active schema as well as schemas related to the currently active schema. This feature is described in detail in the SchemaAgent Validation section in the Schema View section of this user manual.

26.6.13 Create Schema Subset

The **Create Schema Subset**. command pops up the Select Schema Components dialog (*screenshot below*). In this dialog, you check the component or components you wish to create as a single schema subset, then click **Next**. (Note that a check box below the pane enables components from all referenced files to also be listed for selection.)



In the Schema Subset Generation dialog that now appears (*screenshot below*), enter the name/s you want the file/s of the schema subset package to have. You must also specify the folder in which the new schema subset files are to be saved. A schema subset package could have multiple files if one or more of the components being created is an imported component in the original schema. A separate schema file is created for each namespace in the schema subset. The filenames displayed in the dialog are, by default, the names of the original files. But since you are not allowed to overwrite the original files, use new filenames if you wish to save the files in the same folder as the original files.

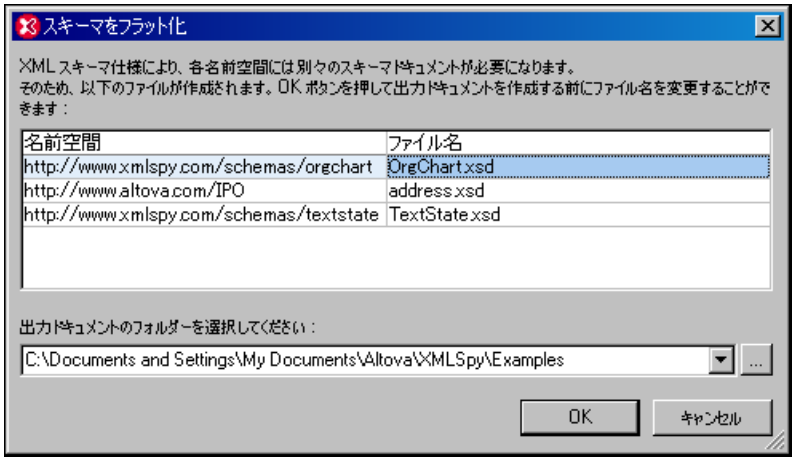


On clicking **OK**, the schema subset file with the namespace corresponding to that of the active file is opened in Schema View. Any other files in the package are created but not opened in Schema View.

26.6.14 Flatten Schema

Flattening the active schema in Schema View is the process of: (i) adding the components of all included schemas as global components of the active schema, and (ii) deleting the included schemas.

To flatten the active schema, select the command **Schema Design | Flatten Schema**. This pops up the Flatten Schema dialog (*screenshot below*), which contains the names of separate files, one for each namespace that will be in the flattened schema. These default names are the same as the original filenames. But since you are not allowed to overwrite the original files, the filenames must be changed if you wish to save in the same folder as the active file. You can browse for a folder in which the flattened schema and its associated files will be saved.



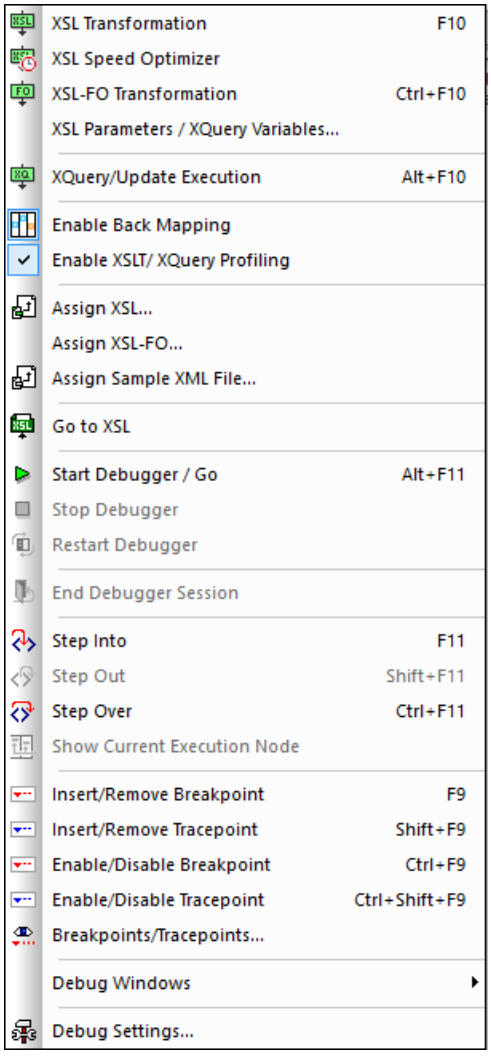
On clicking **OK**, the flattened schema file will be opened in Schema View.

26.7 XSL/ XQuery Menu

The XSL Transformation language lets you specify how an XML document should be converted into other XML documents or text files. One kind of XML document that is generated with an XSLT document is an FO document, which can then be further processed to generate PDF output. XMLSpy contains built-in XSLT processors (for XSLT 1.0, XSLT 2.0, and XSLT 3.0) and can link to an FO processor on your system to transform XML files and generate various kinds of outputs. The location of the FO processor must be specified in the XSL tab of the Options dialog (Tools | Options) in order to be able to use it directly from within the XMLSpy interface.

XMLSpy also has a built-in XQuery engine, which can be used to execute XQuery documents (with or without reference to an XML document).

Commands to deal with all the above transformations are accessible in the **XSL/XQuery** menu. In addition, this menu also contains commands to work with the Altova XSLT/XQuery Debugger.



26.7.1 XSL Transformation

F10

The **XSL/XQuery | XSL Transformation** command transforms an XML document using an assigned XSLT stylesheet. The transformation can be carried out using the appropriate built-in Altova XSLT Engine (Altova XSLT 1.0 Engine for XSLT 1.0 stylesheets; Altova XSLT 2.0 Engine for XSLT 2.0 stylesheets; Altova XSLT 3.0 Engine for XSLT 3.0 stylesheets), the Microsoft-supplied MSXML module, or an external XSLT processor. The processor that is used in conjunction with this command is specified in the XSL tab of the Options dialog (**Tools | Options**).

If your XML document contains a reference to an XSLT stylesheet, then this stylesheet is used for the transformation. (An XSLT stylesheet can be assigned to an XML document using the Assign XSL command. If the XML document is part of a project, an XSLT stylesheet can be specified on a per-folder basis in the Project Properties dialog. Right-click the project folder/s or file/s you wish to transform and select XSL Transformation.) If an XSLT stylesheet has not been assigned to an XML file, you are prompted for the XSLT stylesheet to use. You can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button).

Automating validation with RaptorXML 2017

RaptorXML is Altova's standalone application for XML validation, XSLT transformation, and XQuery transformation. It can be used from the command line, via a COM interface, in Java programs, and in .NET applications. XSLT transformation tasks can therefore be automated with the use of RaptorXML. For example, you can create a batch file that calls RaptorXML to run XSLT transformations on a set of documents and sends the output to a text file. See the RaptorXML documentation for details.

Transformations to ZIP files

In order to enforce output to a ZIP file, including Open Office XML (OOXML) files such as `.docx`, one must specify the ZIP protocol in the file path of the output file. For example:

```
filename.zip|zip/filename.xxx

filename.docx|zip/filename.xxx
```

Note: The directory structure might need to be created before running the transformation. If you are generating files for an Open Office XML archive, you would need to zip the archive files in order to create the top-level OOXML file (for example, `.docx`).

26.7.2 XSL Speed Optimizer

The **XSL Speed Optimizer** command is enabled when an XSLT or XML document is active. It starts the XSL Speed Optimizer, which analyzes the possibility of carrying out faster transformations using the XSLT stylesheet being analyzed. The Optimizer works by running the XSLT stylesheet to be optmized over an XML dataset (one or more XML documents), and analyzing the stylesheet's performance. An optimization strategy is derived from this analysis and can be saved with the XSLT stylesheet (as a processing instruction at the end of the stylesheet). The optimized stylesheet can be used subsequently to produce faster transformations.

On clicking the command, you will be prompted to select, depending on whether an XSLT or XML document is active, respectively, an XML document or XSLT stylesheet. On clicking **OK**, the analysis starts. If the XSLT or XML document already has, respectively, an XML assignment or XSLT assignment in the document, this step is skipped, and the analysis starts straightaway. For details of how to us e the Optimizer, see the section XSL Speed Optimizer. The settings of the Optimizer can be made in the XSL Speed Optimizer tab of the Options dialog (**Tools | Options**).

26.7.3 XSL- FO Transformation

Ctrl+F10

FO is an XML format that describes paged documents. An FO processor, such as the Apache XML Project's FOP, takes an FO file as input and generates PDF as output. The production of a PDF document from an XML document is, therefore, a two-step process.

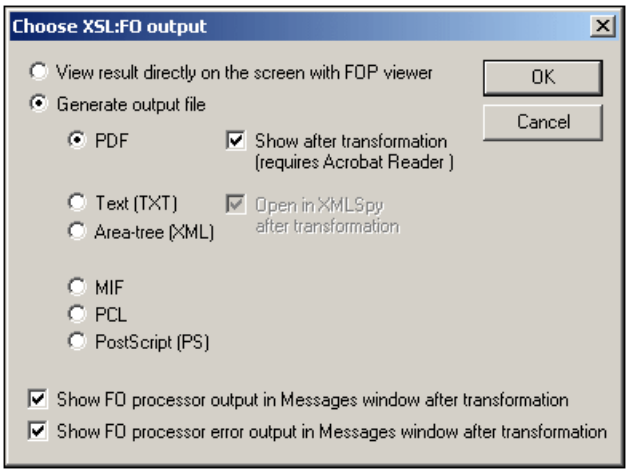
1. The XML document is transformed to an FO document using an XSLT stylesheet.
2. The FO document is processed by an FO processor to generate PDF (or some alternative output).

The **XSL/XQuery | XSL:FO Transformation** command transforms an XML document or an FO document to PDF.

- If the **XSL:FO Transformation** command is executed on a source XML document, then both of the steps listed above are executed, in sequence, one after the other. If the XSLT stylesheet required to transform to FO is not referenced in the XML document, you are prompted to assign one for the transformation. Note that you can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button). The transformation from XML to XSL-FO is carried out by the XSLT processor specified in the XSL tab of the Options dialog (**Tools | Options**). By default the selected XSLT processor is XMLSpy's built-in XSLT processor. The resultant FO document is directly processed with the FO processor specified in the XSL tab of the Options dialog (**Tools | Options**).
- If the **XSL:FO Transformation** command is executed on an FO document, then the document is processed with the FO processor specified in the XSL tab of the Options dialog (**Tools | Options**).

XSL:FO Transformation output

The **XSL:FO Transformation** command pops up the Choose XSL:FO Output dialog (*screenshot below*). (If the active document is an XML document without an XSLT assignment, you are first prompted for an XSLT file.)



You can view the output of the FO processor directly on screen using FOP viewer or you can generate an output file in any one of the following formats: PDF, text, an XML area tree, MIF PCL, or PostScript. You can also switch on messages from the FO processor to show (i) the processor's standard output message in the Messages window; and (ii) the processor's error messages in the Messages window. To switch on either these two options, check the appropriate check box at the bottom of the dialog.

Note:

- Unless you deselected the option to install the FOP processor of the Apache XML Project, it will have been installed in the folder `C:\ProgramData\Altova\SharedBetweenVersions`. If installed, the path to it wil automatically have been entered in the XSL tab of the Options dialog (**Tools | Options**) as the FO processor to use. You can set the path to any FO processor you wish to use.
- The XSL:FO Transformation command can not only be used on the active file in the Main Window but also on any file or folder you select in the active project. To do this, right-click and select **XSL:FO Transformation**. The XSLT stylesheet assigned to the selected project folder is used.

26.7.4 XSL Parameters / XQuery Variables

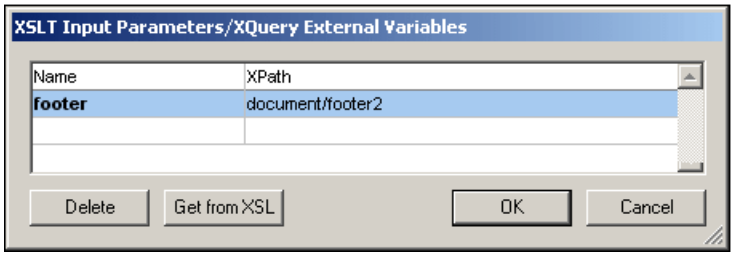
The **XSL/XQuery | XSL Parameters/XQuery Variables** command opens the XSLT Input Parameters/XQuery External Variables dialog (*see screenshot*). You can enter the name of one or more parameters you wish to pass to the XSLT stylesheet, or one or more external XQuery variables you wish to pass to the XQuery document, and their respective values. These parameters are used as follows in XMLSpy:

- When the **XSL Transformation** command in the XSL/XQuery menu is used to transform an XML document, the parameter values currently saved in the dialog are passed to the selected XSLT document and used for the transformation.
- When the **XQuery Execution** command in the XSL/XQuery menu is used to process an XQuery document, the XQuery external variable values currently saved in the dialog are passed to the XQuery document for the execution.

Please note: Parameters or variables that you enter in the XSLT Input Parameters/XQuery External Variables dialog are only passed on to the built-in Altova XSLT engine. Therefore, if you are using MSXML or another external engine that you have configured, these parameters are not passed to this engine.

Using XSLT Parameters

The value you enter for the parameter can be an XPath expression without quotes or a text string delimited by quotes. If the active document is an XSLT document, the **Get from XSL** button will be enabled. Clicking this button inserts parameters declared in the XSLT into the dialog together with their default values. This enables you to quickly include declared parameters and then change their default values as required.



Please note: Once a set of parameter-values is entered in the XSLT Input Parameters/XQuery External Variables dialog, it is used for all subsequent transformations until it is explicitly deleted or the application is restarted. Parameters entered in the XSLT Input Parameters/XQuery External Variables dialog are specified at the application-level, and will be passed to the respective XSLT document for every transformation that is carried out via the IDE from that point onward. This means that:

- parameters are not associated with any particular document
- any parameter entered in the XSLT Input Parameters/XQuery External Variables dialog is erased once XMLSpy has been closed.

Usage example for XSLT parameters

In the following example, we select the required document footer from among three possibilities in the XML document (`footer1`, `footer2`, `footer3`).

```
<?xml version="1.0" encoding="UTF-8"?>
<document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="C:\workarea\footers\footers.xsd">
   <footer1>Footer 1</footer1>
   <footer2>Footer 2</footer2>
   <footer3>Footer 3</footer3>
   <title>Document Title</title>
   <para>Paragraph text.</para>
   <para>Paragraph text.</para>
</document>
```

The XSLT file contains a local parameter called `footer` in the template for the root element. This parameter has a default value of `footer1`. The parameter value is instantiated subsequently in the template with a `$footer` value in the definition of the footer block.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:fo="http://www.w3.org/1999/XSL/Format">
   ...
   <xsl:param name="footer" select="document/footer1" />
   ...
   <xsl:template match="/">
   <fo:root>
      <xsl:copy-of select="$fo:layout-master-set" />
         <fo:page-sequence master-reference="default-page"
            initial-page-number="1" format="1">
```

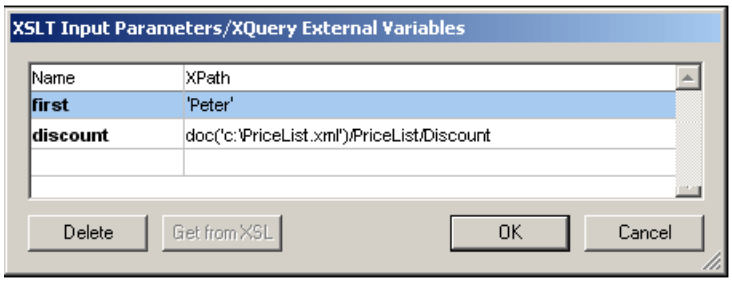
```
            <fo:static-content flow-name="xsl-region-after"
               display-align="after">
               ...
               <fo:inline color="#800000" font-size="10pt" font-weight="bold">
                  <xsl:value-of select="$footer"/>
               </fo:inline>
               ...
            </fo:static-content>
         </fo:page-sequence>
      </fo:root>
   </xsl:template>
</xsl:stylesheet>
```

In the XSLT Input Parameters dialog, a new value for the `footer` parameter can be entered, such as the XPath: `document/footer2` (*see screenshot above*) or a text string. During transformation, this value is passed to the `footer` parameter in the template for the root element and is the value used when the footer block is instantiated.

Note:

- If you use the **XSL:FO Transformation** command (**XSL/XQuery | XSL:FO Transformation**), parameters entered in the XSLT Input Parameters/XQuery External Variables dialog are **not** passed to the stylesheet. In order for these parameters to be used in PDF output, first transform from XML to FO using the XSLT Transformation command (**XSL/XQuery | XSL Transformation**), and then transform the FO to PDF using the **XSL:FO Transformation** command (**XSL/XQuery | XSL:FO Transformation**).
- If you use an XSLT processor other than the built-in Altova XSLT Engines, parameters you enter using the Input Parameters dialog will not be passed to the external processor.

Using external XQuery variables
The value you enter for an external XQuery variable could be an XPath expression without quotes or a text string delimited by quotes. The datatype of the external variable is specified in the variable declaration in the XQuery document.



Note: Once a set of external XQuery variables are entered in the XSLT Input Parameters/ XQuery External Variables dialog, they are used for all subsequent executions until they are explicitly deleted or the application is restarted. Variables entered in the XSLT Input Parameters/ XQuery External Variables dialog are specified at the application-level, and will be passed to the respective XQuery document for every execution that is carried out via the IDE from that point onward. This means that:

- Variables are not associated with any particular document
- Any variable entered in the XSLT Input Parameters/XQuery External Variables dialog is erased once the application (XMLSpy) has been closed down.

Usage example for external XQuery variables
In the following example, a variable `$first` is declared in the XQuery document and is then used in the return clause of the FLWOR expression:

```
xquery version "1.0";
declare variable $first as xs:string external;
let $last := "Jones"
return concat($first," ",$last)
```

This XQuery returns `Peter Jones`, if the value of the external variable (entered in the XSLT Input Parameters/XQuery External Variables dialog) is `Peter`. Note the following:

- The `external` keyword in the variable declaration in the XQuery document indicates that this variable is an external variable.
- Defining the static type of the variable is optional. If a datatype for the variable is not specified in the variable declaration, then the variable value is assigned the type `xs:untypedAtomic`.
- If an external variable is declared in the XQuery document, but no external variable of that name is passed to the XQuery document, then an error is reported.
- If an external variable is declared and is entered in the XSLT Input Parameters/XQuery External Variables dialog, then it is considered to be in scope for the XQuery document being executed. If a new variable with that name is declared within the XQuery document, the new variable temporarily overrides the in-scope external variable. For example, the XQuery document below returns `Paul Jones` even though the in-scope external variable `$first` has a value of `Peter`.

  ```
  xquery version "1.0";
  declare variable $first as xs:string external;
  let $first := "Paul"
  let $last := "Jones"
  return concat($first," ",$last)
  ```

Note: It is not an error if an external XQuery variable (or XSLT parameter) is defined in the XSLT Input Parameters/XQuery External Variables dialog but is not used in the XQuery document. Neither is it an error if an XSLT parameter (or external XQuery variable) is defined in the XSLT Input Parameters/XQuery External Variables dialog but is not used in an XSLT transformation.

26.7.5 XQuery/ Update Execution

The **XSL/XQuery | XQuery/ Update Execution** command executes an XQuery (1.0/3.1) or XQuery Update (1.0/3.0) document. Depending on whether the selected file is an XQuery or XQuery Update file, either an XQuery execution or an XQuery update is carried out. XMLSpy recognizes the type of document (XQuery or XQuery Update) on the basis of the document's file type association (defined in the File types tab of the Options dialog).

The XQuery Engine to use (1.0 or 3.1) is selected automatically on the basis of the version declaration in the document. If there is no version declaration in the document, then the default version specified in the XQuery tab of the Options dialog is used. The **XQuery/ Update Execution** command can be invoked when an XQuery, XQuery Update, or XML file is active. When invoked from an XML file, it opens a dialog asking for an XQuery file to associate with the XML file. You can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button).

Note: The command is also available in the context menu of Project window items.

Automating validation with RaptorXML 2017

RaptorXML is Altova's standalone application for XML validation, XSLT transformation, and XQuery transformation. It can be used from the command line, via a COM interface, in Java programs, and in .NET applications. XQuery execution tasks can therefore be automated with the use of RaptorXML. For example, you can create a batch file that calls RaptorXML to run XQuery executions on a set of documents and sends the output to a text file. See the RaptorXML documentation for details.

26.7.6 Enable Back-Mapping

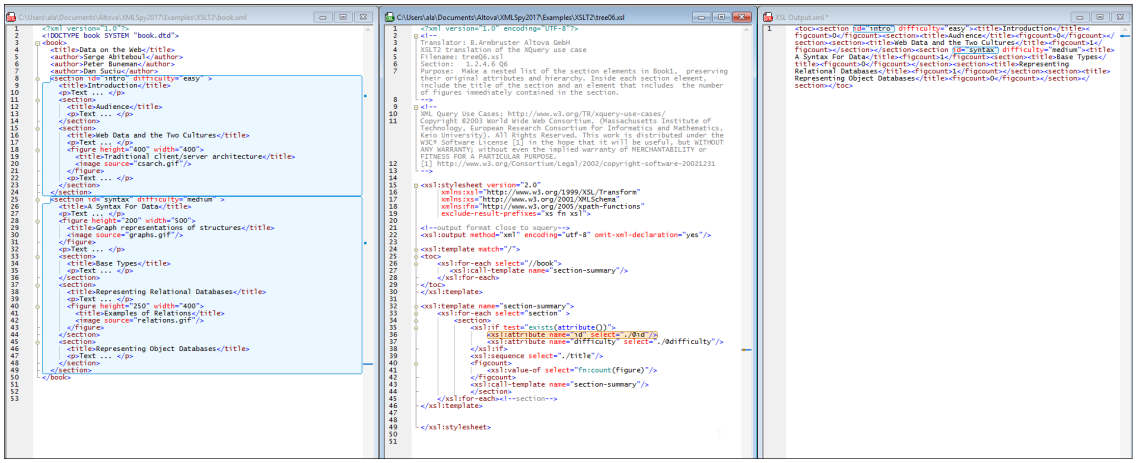
This command, which is also available in the Main toolbar, switches on the Back-Mapping feature.

After Back-Mapping has been enabled (via this command), XSLT transformations and XQuery executions will be carried out so that the result document can be mapped back on to the originating XSLT+XML or XQuery+XML documents. This means that if you click on a node in the result document, then the **XSLT instruction** *and* the **XML source data** that generated that particular result node will be highlighted *(see screenshot below)*. This is useful for checking how exactly the XSLT transformation or XQuery execution creates the different parts of the result document. You can also click a node in either the XSLT/XQuery document or XML document to view the corresponding parts in the other two documents.

Note: Result documents of all types except HTML are opened in Text View. HTML result documents are opened in Browser View, but you can switch to Text View. If the result document is opened in Browser View, then back-mapping is available only by selecting in the result document; back-mapping is not available by selecting in either the XML or XSLT/ XQuery document.

Note: Back-mapping is not available for transformations from Authentic View or those that are run as project transformations.

The screenshot below shows the back-mapping of an XSLT transformation. All three documents —`XML+XSLT+Result`—are tiled vertically, in that order, next to each other. The XSLT instruction that generates the **`section/@id`** attribute in the result document has been clicked. As a result, all the result nodes generated from this instruction are highlighted, as well as the XML source data from which the result node was generated. You can also click nodes in the result document or XML document to highlight the corresponding nodes in the other two documents.



When you click the **Enable XSLT/XQuery Back-Mapping** command, a dialog appears that asks whether you wish to **tile the document windows** after transformation. If you choose to do this, then the three documents will be tiled vertically side-by-side as in the screenshot above.

Note the following points:

- Only XML documents that are loaded from a disk location are displayed; temporary trees are not displayed.
- In some cases, such as XQuery executions, the result document is created without obtaining data from any XML source. In these cases, no XML file is involved in the back-mapping; as a result, none is displayed.
- If multiple XML files are used as data sources, then the first one to be encountered in the transformation or execution process is displayed.
- Back-mapping is slower and more memory-intensive than transformations/executions that are not back-mapped. Be aware of this especially when working with large files.
- The context menu of the result document (when tiled and not tiled) contains commands (**Go to Context Node** and **Go to Source Instruction**) to take you to the corresponding nodes in the XML and XSLT/XQuery document, respectively.

The Back-Mapping toolbar

The Back-Mapping toolbar (*screenshot below*) contains the following icons:

- *Highlight HTML in Browser View on mouseover:* If the result document is displayed in Browser View, then back-mapping is available only by selecting content in the result document; it is not available by selecting in either the XML or XSLT/XQuery document. In Browser View, you can select content for back-mapping in one of two ways: (i) by clicking content in Browser View, or (ii) by mousing over content. Use this toggle command to choose between the two selection methods. Mousing over is useful in cases where clicking content in Browser View might cause a change in the result document (for example, by clicking a radio button or combo box).
- *End back-mapping session:* Ends the back-mapping session.

Ending the back-mapping session

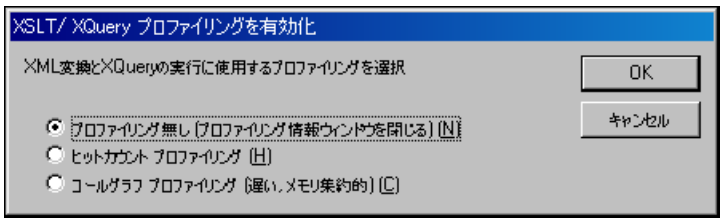
While a back-mapping session is running, an **End Back-Mapping Session** icon is displayed in the Back-Mapping toolbar (*see above*). Click it to end the back-mapping session. You should use the back-mapping session for diagnostics only. If you wish to edit any of the documents, it is best to end the back-mapping session before editing.

Text highlighting colors

The colors of the active back-mapping (back-mapped content that is currently selected) and inactive back-mapping (back-mapped content that is not selected) can be set in the *Miscellaneous* category of the Text Font settings (**Tools | Options | Text Fonts**).

26.7.7 Enable XSLT/ XQuery Profiling

The **Enable XSLT/XQuery profiling** command opens the Enable XSLT/XQuery profiling dialog. This dialog allows you to activate the Profiler, which is a tool that analyzes the time it takes for instructions to execute during an XSLT transformation or XQuery execution.



26.7.8 Assign XSL

The **XSL/XQuery | Assign XSL...** command assigns an XSLT stylesheet to an XML document. Clicking the command opens a dialog to let you specify the XSLT file you want to assign. You can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button).

An `xml-stylesheet` processing instruction is inserted in the XML document:

```
<?xml-stylesheet type="text/xsl" href="C:\workarea\recursion\recursion.xsl"?>
```

Please note: You can make the path of the assigned file relative by clicking the **Make Path Relative To...** check box.

26.7.9 Assign XSL- FO

The **XSL/ XQuery | Assign XSL:FO** command assigns an XSLT stylesheet for transformation to FO to an XML document. The command opens a dialog to let you specify the XSL or XSLT file you want to assign and inserts the required processing instruction into your XML document.

You can make the path of the assigned file relative by clicking the *Make Path Relative To* check box. You can also select a file via a global resource or a URL (click the **Browse** button) or a file

in one of the open windows in XMLSpy (click the **Window** button).

Please note: An XML document may have two XSLT files assigned to it: one for standard XSLT transformations, a second for an XSLT transformation to FO.

26.7.10 Assign Sample XML File

The **XSL/XQuery | Assign Sample XML File** command assigns an XML file to an XSLT document. The command inserts a processing instruction naming an XML file to be processed with this XSLT file when the XSL Transformation is executed on the XSLT file:

```
<?altova_samplexml C:\workarea\html2xml\article.xml?>
```

Please note: You can make the path of the assigned file relative by clicking the Make Path Relative To... check box. You can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button).

26.7.11 Go to XSL

The **XSL/XQuery | Go to XSL** command opens the associated XSLT document. If your XML document contains a stylesheet processing instruction (i.e. an XSLT assignment) such as this:

```
<?xml-stylesheet type="text/xsl" href="Company.xsl"?>
```

then the **Go to XSL** command opens the XSLT document in XMLSpy.

26.7.12 Start Debugger / Go

Alt+F11

The **XSL/XQuery | Start Debugger/Go** command starts or continues processing the XSLT/ XQuery document till the end. If breakpoints have been set, then processing will pause at that point. If tracepoints have been set, output for these statements will be displayed in the Trace window when the closing node of the statement with the tracepoint has been reached. If the debugger session has not been started, then this button will start the session and stop at the first node to be processed. If the session is running, then the XSLT/XQuery document will be processed to the end, or until the next breakpoint is encountered.

26.7.13 Stop Debugger

The **XSL/XQuery | Stop Debugger** command stops the debugger. This is not the same as stopping the debugger **session** in which the debugger is running. This is convenient if you wish to edit a document in the middle of a debugging session or to use alternative files within the same debugging session. After stopping the debugger, you must restart the debugger to start from the beginning of the XSLT/XQuery document.

26.7.14 Restart Debugger

The **XSL/XQuery | Restart Debugger** command clears the output window and restarts the debugging session with the currently selected files.

26.7.15 End Debugger Session

The **XSL/XQuery | End Debugger Session** command ends the debugging session and returns you to the normal XMLSpy view that was active before you started the debugging session. Whether the output documents that were opened for the debugging session stay open depends on a setting you make in the XSLT/ XQuery Debugger Settings dialog.

26.7.16 Step Into

F11

The **XSL/XQuery | Step Into** command proceeds in single steps through all nodes and XPath expressions in the stylesheet. This command is also used to re-start the debugger after it has been stopped.

26.7.17 Step Out

Shift+F11

The **XSL/XQuery | Step Out** command steps out of the current node to the next sibling of the parent node, or to the next node at the next higher level from that of the parent node.

26.7.18 Step Over

Ctrl+F11

The **XSL/XQuery | Step Over** command steps over the current node to the next node at the same level, or to the next node at the next higher level from that of the current node. This command is also used to re-start the debugger after it has been stopped.

26.7.19 Show Current Execution Node

The **XSL/XQuery | Show Current Execution Node** command displays/selects the current execution node in the XSLT/XQuery document and the corresponding context node in the XML document. This is useful when you have clicked in other tabs which show or mark specific code in the XSLT stylesheet or XML file, and you want to return to where you were before you did this.

26.7.20 Insert/ Remove Breakpoint

F9

The **XSL/XQuery | Insert/Remove Breakpoint** command inserts or removes a breakpoint at the current cursor position. Inline breakpoints can be defined for nodes in both the XSLT/XQuery and XML documents, and determine where the processing should pause. A dashed red line appears above the node when you set a breakpoint. Breakpoints cannot be defined on closing nodes, and breakpoints on attributes in XSLT documents will be ignored. This command is also available by right-clicking at the breakpoint location.

26.7.21 Insert/ Remove Tracepoint

Shift+F9

The **XSL/XQuery | Insert/Remove Tracepoint** command inserts or removes a tracepoint at the current cursor position in an XSLT/XQuery document. For statements with a tracepoint, during debugging, the value of the statement is displayed in the Trace window when the closing node of that statement is reached. A dashed blue line appears above the node when you set a tracepoint. Tracepoints cannot be defined on closing nodes. This command is also available by right-clicking at the tracepoint location.

26.7.22 Enable/ Disable Breakpoint

Ctrl+F9

The **XSL/XQuery | Enable/Disable Breakpoint** command enables or disables already defined breakpoints. The red breakpoint highlight turns to gray when the breakpoint is disabled. The debugger does not stop at disabled breakpoints. To disable/enable a breakpoint, place the cursor in that node name and click the **Enable/Disable Breakpoint** command. This command is also available by right-clicking at the location where you want to enable/disable the breakpoint.

26.7.23 Enable/ Disable Tracepoint

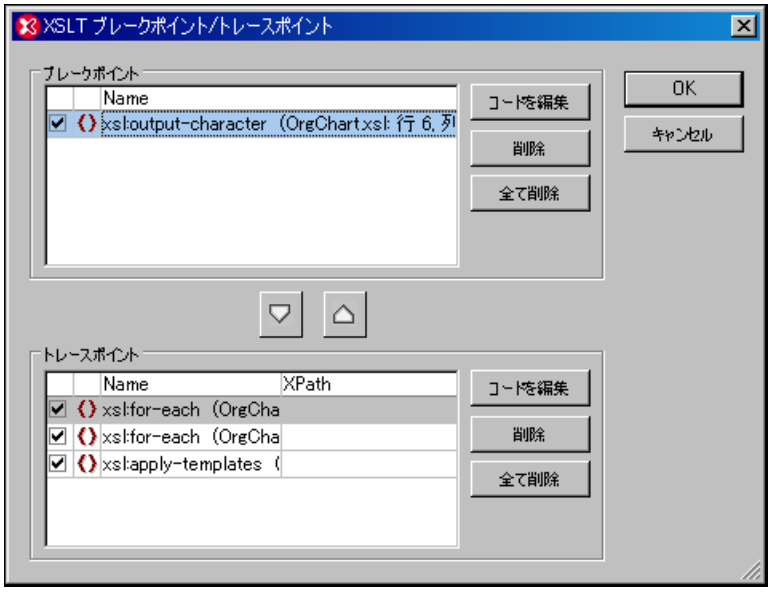
Ctrl+Shift+F9

The **XSL/XQuery | Enable/Disable Tracepoint** command enables or disables already defined tracepoints. The blue tracepoint highlight turns to gray when the tracepoint is disabled. No output is displayed for statements with disabled tracepoints. To disable/enable a tracepoint, place the cursor in that node name and click the **Enable/Disable Tracepoint** command. This command is also available by right-clicking at the location where you want to enable/disable the tracepoint.

26.7.24 Breakpoints/ Tracepoints

The **XSL/XQuery | Breakpoints/Tracepoints...** command opens the XSLT Breakpoints / Tracepoints dialog, which displays a list of all currently defined breakpoints and tracepoints

(including disabled breakpoints and tracepoints) in all files in the current debugging session.



The check boxes indicate whether a breakpoint or tracepoint is enabled (checked) or disabled. You can remove the highlighted breakpoint or tracepoint by clicking the corresponding **Remove** button, and remove all breakpoints by clicking the corresponding **Remove All** button. The **Edit Code** button takes you directly to that breakpoint/tracepoint in the file.

Use the down arrow to move the highlighted breakpoint to the **Tracepoints** pane and the up arrow to move the highlighted tracepoint to the **Breakpoints** pane.

In the **XPath** column in the Tracepoints pane, you can set an XPath for each tracepoint.

26.7.25 Debug Windows

Placing the cursor over the **XSL/XQuery | Debug Windows** command pops out a submenu with a list of the various Information Windows of the XSLT/XQuery Debugger. Selecting an Information Window from this list shows/hides that Information Window in the XSLT/XQuery Debugger interface. This command can be used to effect only when a debugging session is in progress.

26.7.26 Debug Settings

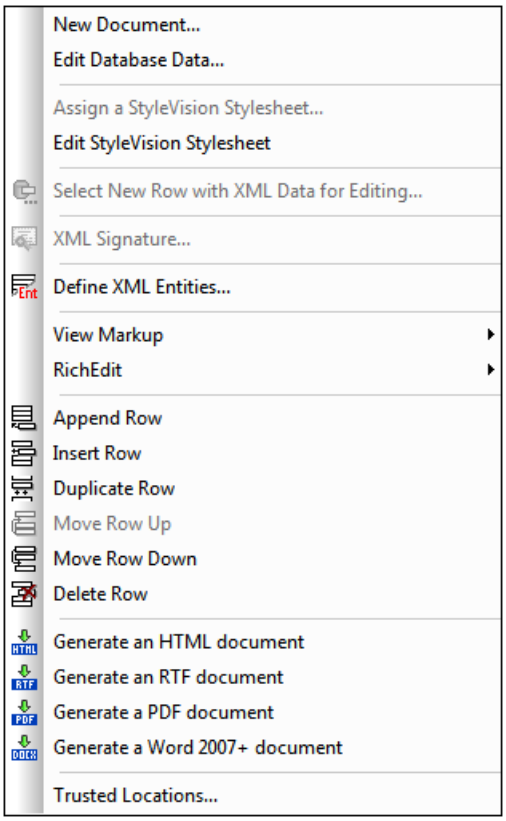
The **XSL/XQuery | Debug Settings** command opens the Debug Settings dialog, which enables you to set user options for the Debugger. See the XSLT/XQuery Debugger section for details.

26.8 Authentic Menu

Authentic View enables you to edit XML documents **based on StyleVision Power Stylesheets (.sps files) created in Altova's StyleVision product!** These stylesheets contain information that enables an XML file to be displayed graphically in Authentic View. In addition to containing display information, StyleVision Power Stylesheets also allow you to write data to the XML file. This data is dynamically processed using all the capability available to XSLT stylesheets and instantly produces the output in Authentic View.

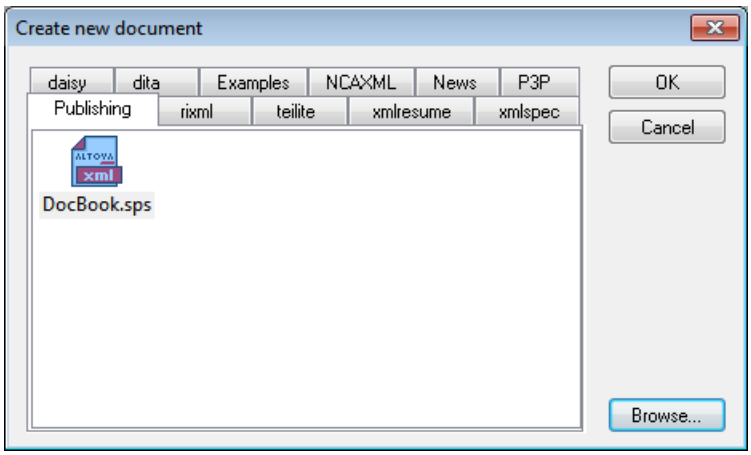
Additionally, StyleVision Power Stylesheets can be created to display an editable XML view of a database. The StyleVision Power Stylesheet contains information for connecting to the database, displaying the data from the database in Authentic View, and writing back to the database.

The **Authentic** menu contains commands relevant to editing XML documents in Authentic View. For a tutorial on Authentic View, see the Authentic View Tutorials section.



26.8.1 New Document

This command enables you to open a new XML document template in Authentic View. The XML document template is based on a StyleVision Power Stylesheet (.sps file), and is opened by selecting the StyleVision Power Stylesheet (SPS file) in the Create New Document dialog (*screenshot below*). On selecting an SPS and clicking **OK**, the XML document template defined for that SPS file is opened in Authentic View.



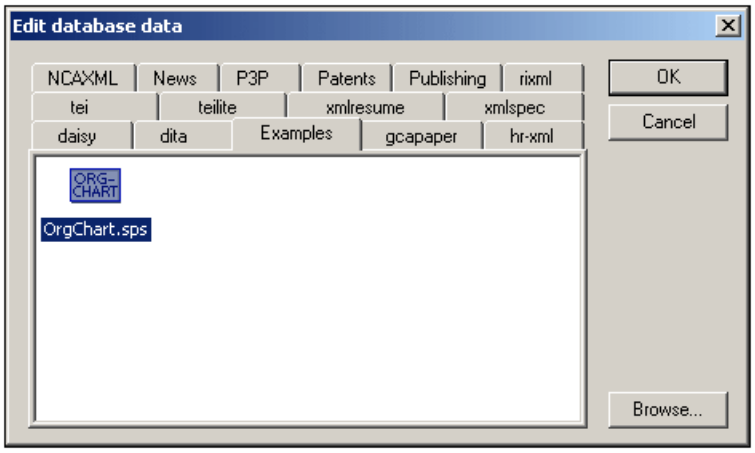
The Create New Document dialog offers a choice of XML document templates that are based on popular DTDs or schemas. Alternatively, you can browse for a custom-made SPS file that has a Template XML File assigned to it. SPS files are created using Altova StyleVision, an application that enables you to design XML document templates based on a DTD or XML Schema. After designing the required SPS in StyleVision, an XML file is assigned (in StyleVision) as a Template XML File to the SPS. The data in this XML file provides the starting data of the new document template that is opened in the Authentic View of XMLSpy.

The new XML document template will therefore have the documentation presentation properties defined in the SPS and the data of the XML file that was selected as the Template XML File. The Authentic View user can now edit the XML document template in a graphical WYSIWYG interface, and save it as an XML document.

26.8.2 Edit Database Data

The **Authentic** | **Edit Database Data...** command enables you to open an editable view of a database (DB) in Authentic View. All the information about connecting to the DB and how to display the DB and accept changes to it in Authentic View is contained in a StyleVision Power Stylesheet. It is such a DB-based StyleVision Power Stylesheet that you open with the **Edit Database Data...** command. This sets up a connection to the DB and displays the DB data (through an XML lens) in Authentic View.

Clicking the **Edit Database Data...** command opens the Edit Database Data dialog.



Browse for the required SPS file, and select it. This connects to the DB and opens an editable view of the DB in Authentic View. The design of the DB view displayed in Authentic View is contained in the StyleVision Power Stylesheet.

Please note: If, with the **Edit Database Data...** command, you attempt to open a StyleVision Power Stylesheet that is not based on a DB or to open a DB-based StyleVision Power Stylesheet that was created in a version of StyleVision prior to the StyleVision 2005 release, you will receive an error.

Please note: StyleVision Power Stylesheets are created using Altova StyleVision.

26.8.3 Assign a StyleVision Stylesheet

The **Assign a StyleVision Stylesheet** command assigns a StyleVision Power Stylesheet (SPS) to an **XML document** to enable the viewing and editing of that XML document in Authentic View. The StyleVision Power Stylesheet that is to be assigned to the XML file must be based on the same schema as that on which the XML file is based.

To assign a StyleVision Power Stylesheet to an XML file:

1. Make the XML file the active file and select the **Authentic** | **Assign a StyleVision Stylesheet...** command.
2. The command opens a dialog box in which you specify the StyleVision Power Stylesheet file you wish to assign to the XML.
3. Click **OK** to insert the required SPS statement into your XML document. Note that you can make the path to the assigned file relative by clicking the **Make path relative to ...** check box. You can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button).

```
<?xml version="1.0" encoding="UTF-8"?>
<?altova_sps HTML-Orgchart.sps?>
```

In the example above, the StyleVision Power Stylesheet is called `HTML_Orgchart.sps`, and it is located in the same directory as the XML file.

Note: Previous versions of Altova products used a processing instruction with a target or name of `xmlspysps`, so a processing instruction would look something like `<?xmlspysps HTML-Orgchart.sps?>`. These older processing instructions are still valid with Authentic View in current versions of Altova products.

26.8.4 Edit StyleVision Stylesheet

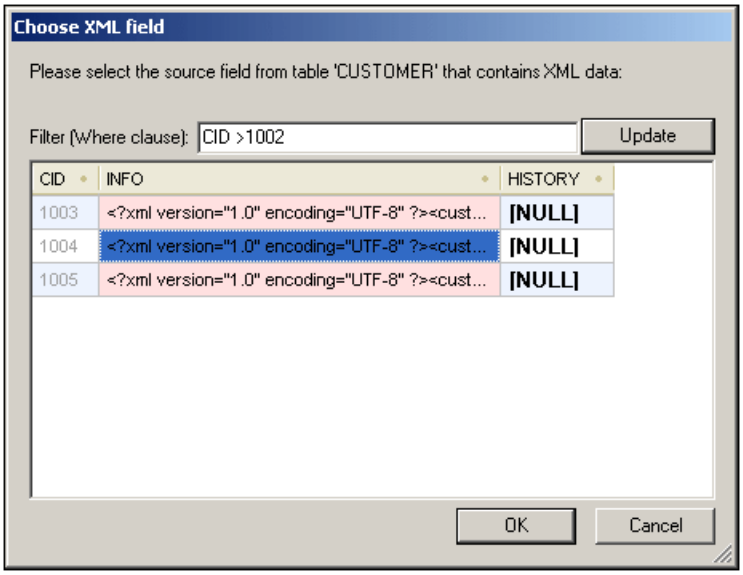
The **Authentic** | **Edit StyleVision Stylesheet** command starts StyleVision and allows you to edit the StyleVision Power Stylesheet immediately in StyleVision. The command is enabled only if a StyleVision Power Stylesheet has been assigned to the XML document.

26.8.5 Select New Row with XML Data for Editing

The **Select New Row with XML Data for Editing** command enables you to select a new row from the relevant table in an XML DB, such as IBM DB2. This row appears in Authentic View, can be edited there, and then saved back to the DB.

When an XML DB is used as the XML data source, the XML data that is displayed in Authentic View is the XML document contained in one of the cells of the XML data column. The **Select New Row with XML Data for Editing** command enables you to select an XML document from

another cell (or row) of that XML column. Selecting the **Select New Row...** command pops up the Choose XML Field dialog (*screenshot below*), which displays the table containing the XML column.



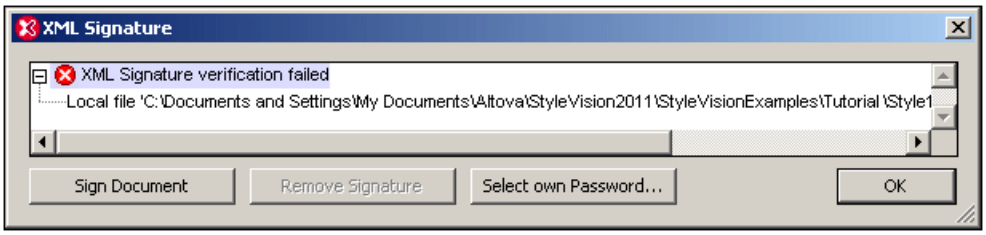
You can enter a filter for this table. The filter should be an SQL `WHERE` clause (just the condition, without the `WHERE` keyword, for example: `CID>1002`). Click **Update** to refresh the dialog. In the screenshot above, you can see the result of a filtered view. Next, select the cell containing the required XML document and click **OK**. The XML document in the selected cell (row) is loaded into Authentic View.

26.8.6 XML Signature

The **XML Signature** command is available in Authentic View when the associated SPS has XML Signatures enabled. The **XML Signature** command is also available as the XML Signature toolbar icon in the Authentic toolbar.

Verification and own certificate/password

Clicking the **XML Signature** command starts the signature verification process. If no signature is present in the document, a message to that effect is displayed in the XML Signature dialog (*see screenshot below*), and the dialog will have a button that enables the Authentic View user to sign the document.

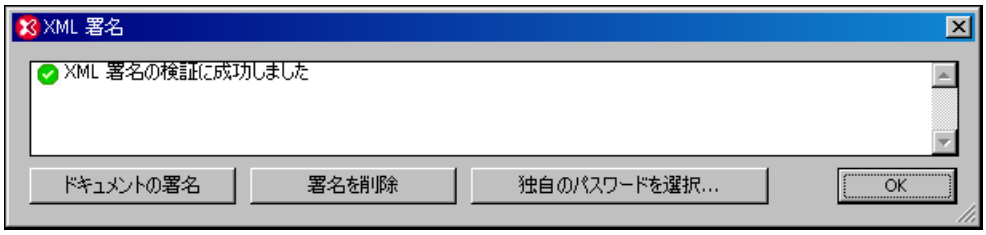


If the **Select Own Certificate** or **Select Own Password** button is present in this dialog, it means that the Authentic View has been given the option of selecting an own certificate/password. (Whether a certificate or password is to be chosen has been decided by the SPS designer at the time the signature was configured. The signature will be either certificate-based or password-

based.) Clicking either of these buttons, if present in the dialog, enables the Authentic View user to browse for a certificate or to enter a password. The Authentic View user's selection is stored in memory and is valid for the current session only. If, after selecting a certificate or password, the document or application is closed, the certificate/password setting reverts to the setting originally saved with the SPS.

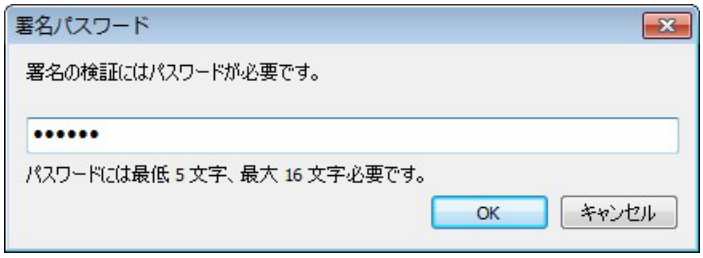
Verification and authentication information

If the verification process is run on a signed document, two general situations are possible. First: If the authentication information is available (in the signature or the SPS), then the verification process is executed directly and the result is displayed (*screenshot below*).



Authentication information is either the signing certificate's key information or the signing password. The SPS designer will have specified whether the certificate's key information is saved in the signature when the XML document is signed, or, in the case of a password-based signature, whether the password is saved in the SPS. In either of these cases, the authentication is available. Consequently the verification process will be run directly, without requiring any input from the Authentic View user.

The second possible general situation occurs when authentication information is not available in the signature (certificate's key information) or SPS file (password). In this situation, the Authentic View user will be asked to supply the authentication information: a password (*see screenshot below*) or the location of a certificate.



26.8.7 Define XML Entities

You can define entities for use in Authentic View, whether your document is based on a DTD or an XML Schema. Once defined, these entities are displayed in the Entities Entry Helper and in the **Insert Entity** submenu of the context menu. When you double-click on an entity in the Entities Entry Helper, that entity is inserted at the cursor insertion point.

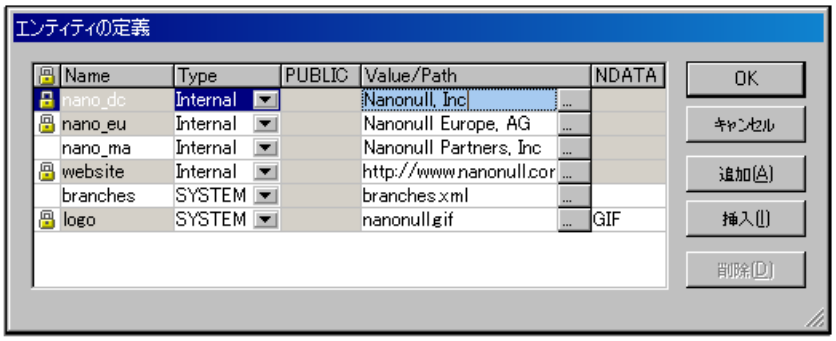
An entity is useful if you will be using a text string, XML fragment, or some other external resource in multiple locations in your document. You define the entity, which is basically a short name that stands in for the required data, in the Define Entities dialog. After defining an entity you can use it at multiple locations in your document. This helps you save time and greatly enhances maintenance.

There are two broad types of entities you can use in your document: a **parsed entity**, which is

XML data (either a text string or a fragment of an XML document), or an **unparsed entity**, which is non-XML data such as a binary file (usually a graphic, sound, or multimedia object). Each entity has a name and a value. In the case of parsed entities the entity is a placeholder for the XML data. The value of the entity is either the XML data itself or a URI that points to a `xml`file that contains the XML data. In the case of unparsed entities, the value of the entity is a URI that points to the non-XML data file.

To define an entity:

1. Click **Authentic | Define XML Entities...**. This opens the Define Entities dialog.



2. Enter the name of your entity in the **Name** field. This is the name that will appear in the Entities Entry Helper.
3. Enter the type of entity from the drop-down list in the **Type** field. Three types are possible. An **Internal** entity is one for which the text to be used is stored in the XML document itself. Selecting **PUBLIC** or **SYSTEM** specifies that the resource is located outside the XML file, and will be located with the use of a public identifier or a system identifier, respectively. A system identifier is a URI that gives the location of the resource. A public identifier is a location-independent identifier, which enables some processors to identify the resource. If you specify both a public and system identifier, the public identifier resolves to the system identifier, and the system identifier is used.
4. If you have selected PUBLIC as the Type, enter the public identifier of your resource in the PUBLIC field. If you have selected Internal or SYSTEM as your Type, the PUBLIC field is disabled.
5. In the **Value/Path** field, you can enter any one of the following:
   - If the entity type is Internal, enter the text string you want as the value of your entity. Do not enter quotes to delimit the entry. Any quotes that you enter will be treated as part of the text string.
   - If the entity type is SYSTEM, enter the URI of the resource or select a resource on your local network by using the **Browse** button. If the resource contains parsed data, it must be an XML file (i.e. it must have a `xml`extension). Alternatively, the resource can be a binary file, such as a GIF file.
   - If the entity type is PUBLIC, you must additionally enter a system identifier in this field.
6. The NDATA entry tells the processor that this entity is not to be parsed but to be sent to the appropriate processor. The NDATA field should therefore be used with unparsed entities only.

Dialog features
You can append, insert, and delete entities by clicking the appropriate buttons. You can also sort entities on the alphabetical value of any column by clicking the column header; clicking once sorts in ascending order, twice in descending order. You can also resize the dialog box and the width of columns.

Once an entity is used in the XML document, it is locked and cannot be edited in the Define

Entities dialog. Locked entities are indicated by a lock symbol in the first column. Locking an entity ensures that the XML document valid with respect to entities. (The document would be invalid if an entity is referenced but not defined.)

Duplicate entities are flagged.

Limitations

- An entity contained within another entity is not resolved, either in the dialog, Authentic View, or XSLT output, and the ampersand character of such an entity is displayed in its escaped form, i.e. `&`
- External entities are not resolved in Authentic View, except in the case where an entity is an image file and it is entered as the value of an attribute which has been defined in the schema as being of type `ENTITY` or `ENTITIES`. Such entities are resolved when the document is processed with an XSLT generated from the SPS.

26.8.8 View Markup

The **View Markup** command has a submenu with options to control markup in the Authentic XML document. These options are described below.

The **Hide Markup** command hides markup symbols in Authentic View.

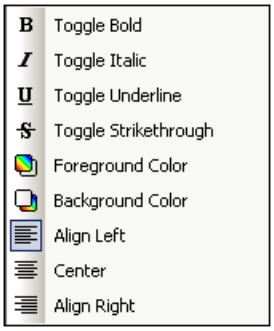
The **Show Small Markup** command shows small markup symbols in Authentic View.

The **Show Large Markup** command shows large markup symbols in Authentic View.

The **Show Mixed Markup** command shows mixed markup symbols in Authentic View. The person who designs the StyleVision Power Stylesheet can specify either large markup, small markup, or no markup for individual elements/attributes in the document. The Authentic View user sees this customized markup in mixed markup viewing mode.

26.8.9 RichEdit

Mousing over the **RichEdit** command pops out a submenu containing the RichEdit markup commands (*screenshot below*). The menu commands in this submenu are enabled only in Authentic View and when the cursor is placed inside an element that has been created as a RichEdit component in the SPS design.



The text-styling properties of the RichEdit menu will be applied to the selected text when a RichEdit markup command is clicked. The Authentic View user can, in addition to the font and font-size specified in the Authentic toolbar, additionally specify the font-weight, font-style, font-decoration, color, background color and alignment of the selected text.

26.8.10 Append/ Insert/ Duplicate/ Delete Row

The **Append Row** command appends a row to the current table in Authentic View.

The **Insert Row** command inserts a row into the current table in Authentic View.

The **Duplicate Row** command duplicates the current table row in Authentic View.

The **Delete Row** command deletes the current table row in Authentic View.

26.8.11 Move Row Up/ Down

The **Move Row Up** command moves the current table row up by one row in Authentic View.

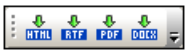
The **Move Row Down** command moves the current table row down by one row in Authentic View.

26.8.12 Generate HTML, RTF, PDF, Word 2007+ Document

These four commands generate output documents from the Authentic View XML document stored in a PXF file:

- **Generate an HTML Document**
- **Generate an RTF Document**
- **Generate a PDF Document**
- **Generate a Word 2007+ Document**

They are also available in the Portable XML Form (PXF) toolbar (*screenshot below*).

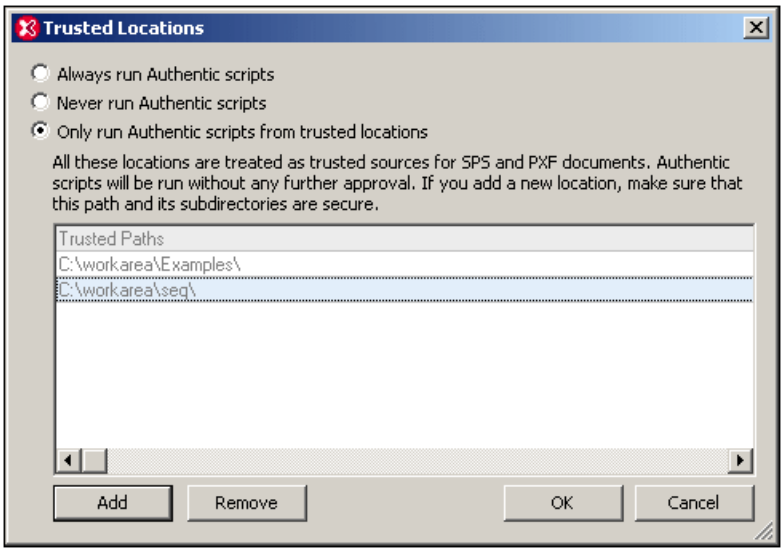


Clicking the individual command or buttons generates HTML, RTF, PDF, or DocX output, respectively.

These buttons are enabled when a PXF file is opened in Authentic View. Individual commands and buttons are enabled if the PXF file was configured to contain the XSLT stylesheet for that specific output format. For example, if the PXF file was configured to contain the XSLT stylesheets for HTML and RTF, then only the commands and toolbar buttons for HTML and RTF output will be enabled while those for PDF and DocX (Word 2007+) output will be disabled.

26.8.13 Trusted Locations

The Trusted Locations command pops up the Trusted Locations dialog (*screenshot below*), in which you can specify the security settings for scripts in an SPS. When an XML file based on a script-containing SPS is switched to Authentic View, the script will be allowed to run or not depending on the settings you make in this dialog.



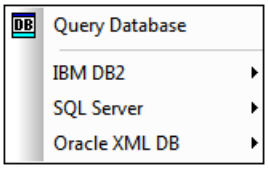
The three available options are:

- Authentic scripts are always run when a file is opened in Authentic View.
- Authentic scripts are never run when a file is opened in Authentic View.
- Only Authentic scripts in trusted locations are run. The list of trusted (folder) locations is shown in the bottom pane. Use the **Add** button to browse for a folder and add it to the list.

To remove an entry from the list, select an entry in the Trusted Locations list and click **Remove**.

26.9 DB Menu

The **DB** menu is the menu for database (DB) operations. It is shown in the screenshot below and contains the menu items listed below. Descriptions of commands in the sub-menus of the DB menu are in the sub-sections of this section.

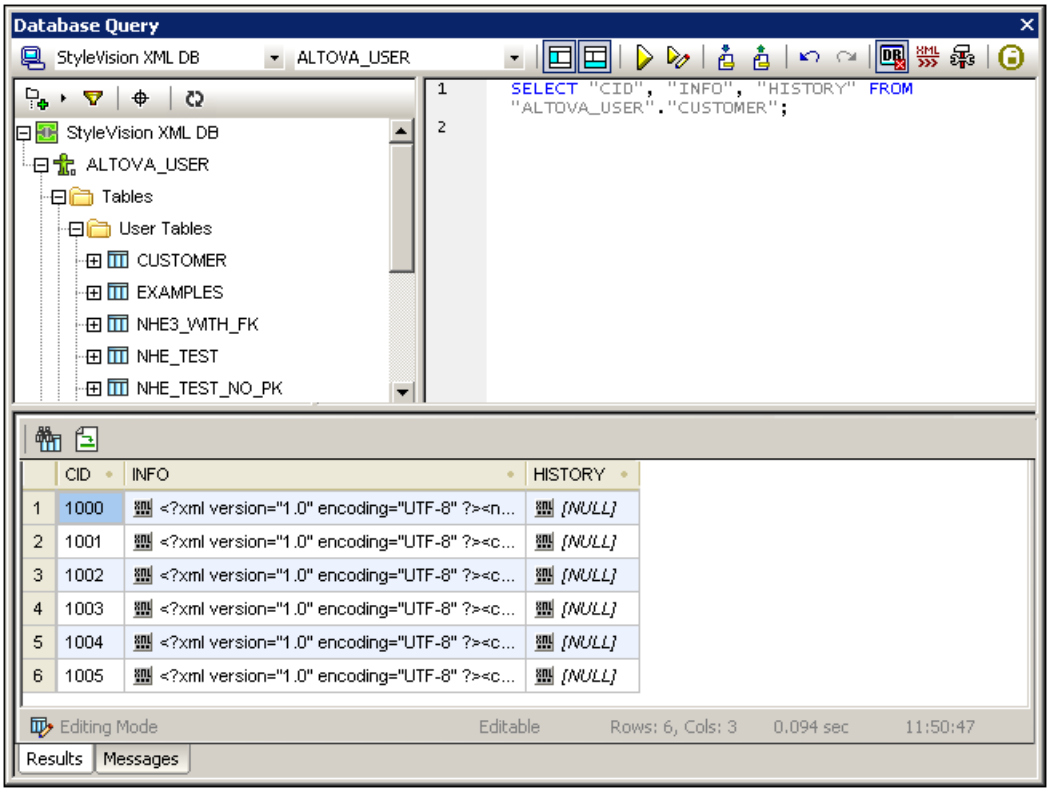


- Query Database, which enables you to query a variety of databases.
- IBM DB2, which contains commands that provide support for IBM DB2-specific functionality.
- SQL Server, which contains commands for managing SQL Server databases.
- Oracle databases, which contains command for working with Oracle databases.

The operations described in this section require a connection to a database (for instructions, see Connecting to a Database).

26.9.1 Query Database

The **Query Database** command opens the Database Query window (*screenshot below*). Once the Query Window is open, its display can be toggled on and off by clicking either the **DB | Query Database** command or the Query Database toolbar icon .



Overview of the Database Query window

The Database Query window consists of three parts:

- A **Browser pane** at top left, which displays connection info and database tables.
- A **Query pane** at top right, in which the query is entered.
- A tabbed **Results/Messages pane**. The Results pane displays the query results in what we call the Result Grid. The Messages pane displays messages about the query execution, including warnings and errors.

The Database Query window has a toolbar at the top. At this point, take note of the two toolbar icons below. The other toolbar icons are described in the section, Query Pane: Description and Features.

Toggles the Browser pane on and off.

Toggles the Results/Messages pane on and off.

Overview of the Query Database mechanism

The Query Database mechanism is as follows. It is described in detail in the sub-sections of this section

1. A connection to the database is established via the Database Query window. Supported databases include: MS Access 2000 and 2003; Microsoft SQL Server; Oracle; MySQL; Sybase; and IBM DB2.
2. The connected database or parts of it are displayed in the Browser pane, which can be configured to suit viewing requirements.
3. A query written in a syntax appropriate to the database to be queried is entered in the Query pane, and the query is executed.
4. The results of the query can be viewed through various filters, edited, and saved back to the DB.

Data Sources

In order to query a database, you have to first connect to the required database This section describes how to:

- Connect to a database, and
- Select the required data source and root object from among multiple existing connections.

Connecting to a database

When you click the **Query Database** command for the first time in a session (or when no database connection exists), the Quick Connect dialog (*screenshot below*) pops up to enable you to connect to a database. To make connections subsequently, click the Quick Connect icon in the Database Query window.

How to connect to a database via the Quick Connect dialog is described in the section Connecting to a Data Source.

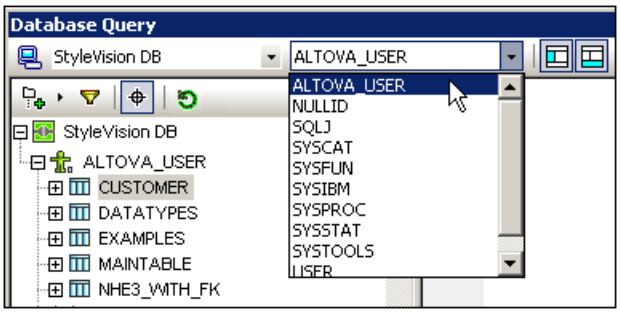
The following databases are supported. The available root object for each database is also listed. While Altova endeavors to support other databases, successful connection and data processing have only been tested with the databases listed below. If your Altova application is a 64-bit version, ensure that you have access to the 64-bit database drivers needed for the specific database you are connecting to.

Database	Root Object	Notes
Firebird 2.5.4	database	
IBM DB2 8.x, 9.1, 9.5, 9.7, 10.1, 10.5	schema	
IBM DB2 for i 6.1, 7.1	schema	Logical files are supported and shown as views.
IBM Informix 11.70	database	
Microsoft Access 2003, 2007, 2010, 2013	database	
Microsoft Azure SQL Database	database	SQL Server 2016 codebase
Microsoft SQL Server 2005, 2008, 2012, 2014, 2016	database	
MySQL 5.0, 5.1, 5.5, 5.6	database	
Oracle 9i, 10g, 11g, 12c	schema	
PostgreSQL 8.0, 8.1, 8.2,	database	PostgreSQL connections are supported both

8.3, 9.0.10, 9.1.6, 9.2.1, 9.4, 9.5		as native connections and driver-based connections through interfaces (drivers) such as ODBC or JDBC. Native connections do not require any drivers.
Progress OpenEdge 11.6	database	
SQLite 3.x	database	SQLite connections are supported as native, direct connections to the SQLite database file. No separate drivers are required. In Authentic view, data coming from a SQLite database is not editable. When you attempt to save SQLite data from the Authentic view, a message box will inform you of this known limitation.
Sybase ASE15	database	

Selecting the required data source

All the existing connections and the root objects of each are listed, respectively, in two combo boxes in the toolbar of the Database Query window (*screenshot below*).



In the screenshot above, the database with the name `StyleVision DB` has been selected. Of the available root objects for this database, the root object `ALTOVA_USER` has been selected. The database and the root object are then displayed in the Browser pane.

Browser Pane: Viewing the DB Objects

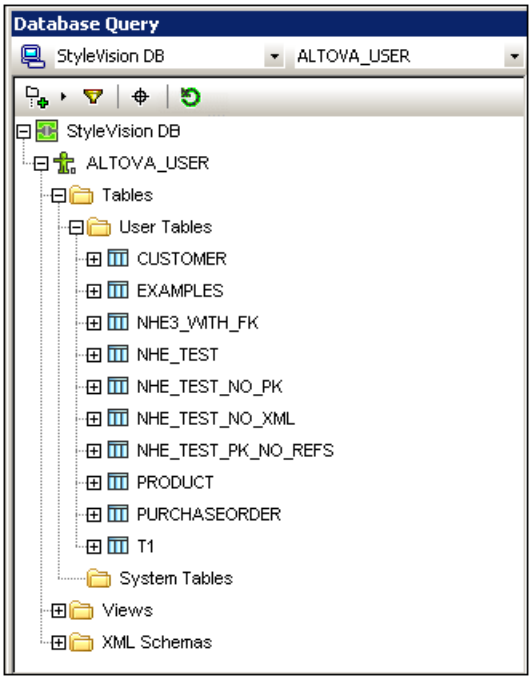
The Browser pane provides an overview of objects in the selected database. This overview includes database constraint information, such as whether a column is a primary or foreign key. In IBM DB2 version 9 databases, the Browser additionally shows registered XML schemas in a separate folder.

This section describes the following:

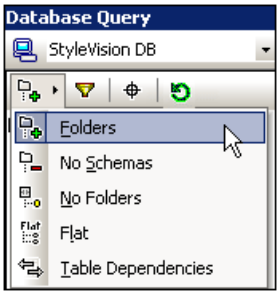
- The layouts available in the Browser pane.
- How to filter database objects.
- How to find database objects.

Browser pane layouts

The default Folders layout displays database objects hierarchically. Depending on the selected object, different context menu options are available when you right-click an item.



To select a layout for the Browser, click the Layout icon in the toolbar of the Browser pane and select the layout from the drop-down list (*screenshot below*). Note that the icon changes with the selected layout.



The available layouts are:

- *Folders:* Organizes database objects into folders based on object type in a hierarchical tree, this is the default setting.
- *No Schemas:* Similar to the Folders layout, except that there are no database schema folders; tables are therefore not categorized by database schema.
- *No Folders:* Displays database objects in a hierarchy without using folders.
- *Flat:* Divides database objects by type in the first hierarchical level. For example, instead of columns being contained in the corresponding table, all columns are displayed in a separate Columns folder.
- *Table Dependencies:* Categorizes tables according to their relationships with other tables. There are categories for tables with foreign keys, tables referenced by foreign keys and tables that have no relationships to other tables.

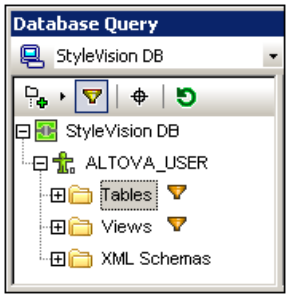
To sort tables into User and System tables, switch to Folders, No Schemas or Flat layout, then right-click the Tables folder and select **Sort into User and System Tables**. The tables are sorted alphabetically in the User Tables and System Tables folders.

Filtering database objects

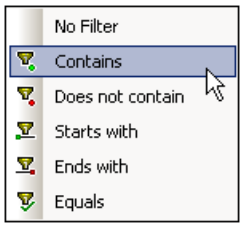
In the Browser pane (in all layouts except No Folders and Table Dependencies), schemas, tables, and views can be filtered by name or part of a name. Objects are filtered as you type in the characters, and filtering is case-insensitive by default.

To filter objects in the Browser, do the following:

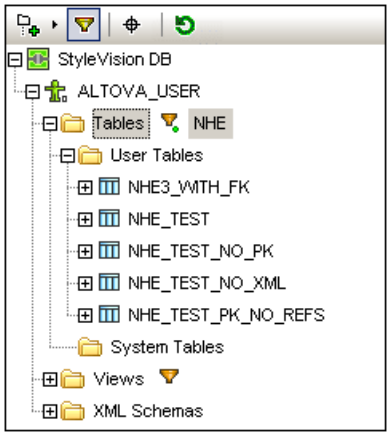
1. Click the Filter Folder Contents icon in the toolbar of the Browser pane. Filter icons appear next to the Tables and Views folders in the currently selected layout (*screenshot below*).



2. Click the filter icon next to the folder you want to filter, and select the filtering option from the popup menu, for example, `Contains`.



3. In the entry field that appears, enter the filter string (in the screenshot below, the filter string on the `Tables` folder is `NHE`). The filter is applied as you type.



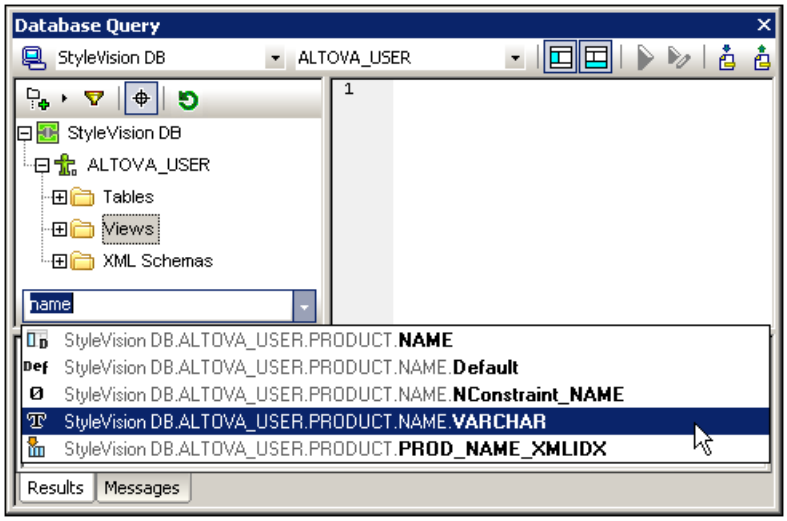
Finding database objects

To find a specific database item by its name, you can use the Browser pane's Object Locator. This works as follows:

1. In the toolbar of the Browser pane, click the Object Locator icon. A drop-down list appears

at the bottom of the Browser.

2. Enter the search string in the entry field of this list, for example `name` (*screenshot below*). Clicking the drop-down arrow displays all objects that contain the search string.



3. Click the object in the list to see it in the Browser.

Query Pane: Description and Features

The Query pane is an intelligent SQL editor for entering queries to the selected database. After entering the query, clicking the Execute command of the Database Query window executes the query and displays the result and execution messages in the Results/Messages pane. How to work with queries is described in the next section, Query Pane: Working with Queries. In this section, we describe the main features of the Query pane:

- SQL Editor icons in the Database Query toolbar
- SQL Editor options
- Auto-completion of SQL statements
- Definition of regions in an SQL script
- Insertion of comments in an SQL script
- Use of bookmarks

SQL Editor icons in the Database Query toolbar

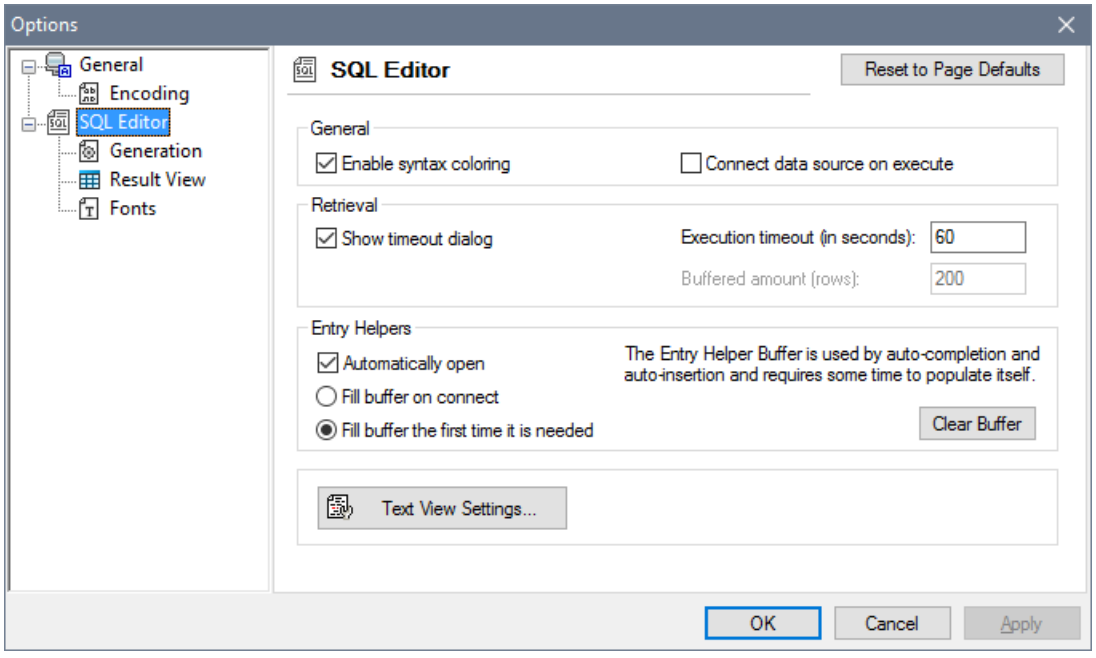
The following icons in the toolbar of the Database Query window are used when working with the SQL Editor:

	Execute	Executes currently selected SQL statement. If script contains multiple statements and none is selected, then all are executed.
	Execute with Data Editing	Same as for Execute command, except that results (in Results tab) are editable.
	Import SQL File	Opens an SQL file in the SQL Editor.
	Export SQL File	Saves SQL queries to an SQL file.

	Undo	Undoes an unlimited number of edits in SQL Editor.
	Redo	Redoes an unlimited number of edits in SQL Editor.
	Hide DB Query on XML Open	Sets whether the DB Query window should be hidden when an XML document is opened for editing.
	Auto-Commit on XML Save	When an edited XML document is saved in XMLSpy, changes are committed to the DB if this toggle is on. Otherwise, changes have to be explicitly committed in the Results Pane.
	Options	Open the Options dialog of SQL Editor.
	Open SQL Script in DatabaseSpy	Opens the SQL script in Altova's DatabaseSpy product.

Options

Clicking the **Options** icon in the Database Query toolbar pops up the Options dialog (*screenshot below*). A page of settings can be selected in the left-hand pane, and the options on that page can be selected. Click the **Reset to Page Defaults** button to reset the options on that page to their original settings.



The key settings are as follows:

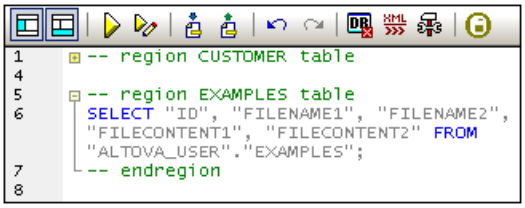
- ***General | Encoding:*** Options for setting the encoding of new SQL files, of existing SQL files for which the encoding cannot be detected, and for setting the Byte Order Mark (BOM). (If the encoding of existing SQL files can be detected, the files are opened and saved without changing the encoding.)
- ***SQL Editor:*** Options for toggling syntax coloring and data source connections on execution on/off. A timeout can be set for query execution, and a dialog to change the

timeout can also be shown if the specified time is exceeded. Entry helpers refer to the entry helpers that appear as part of the auto-completion feature. When you type in an SQL statement, the editor displays a list of context-sensitive auto-completion suggestions. These suggestions can be set to appear automatically. If the automatic display is switched off, then you can ask for an auto-completion suggestion in SQL Editor by pressing **Ctrl +Spacebar**. The buffer for the entry helper information can be filled either on connection to the data source or the first time it is needed. The Text View settings button opens the Text View options window of XMLSpy.

- ***SQL Editor | SQL Generation:*** The application generates SQL statements when you drag objects from the Browser pane into the Query pane. Options for SQL statement generation can be set in the SQL generation tab. Use the *Database* pane to select a database kind and set the statement generation options individually for the different database kinds you are working with. Activating the *Apply to all databases* check box sets the options that are currently selected for all databases. Options include appending semi-colons to statements and surrounding identifiers with escape characters. When the *Append semicolons to statement end* check box is activated, a semicolon is appended when you generate an SQL statement in the SQL Editor. Note that editing of data in Oracle databases and IBM iSeries and DB2 databases via a JDBC connection is possible only if this check box is unchecked.
- ***SQL Editor | Result View:*** Options to configure the Result tab.
- ***SQL Editor | Fonts:*** Options for setting the font style of the text in the Text Editor and in the Result View.

Definition of regions in an SQL script

Regions are sections in SQL scripts that are marked and declared to be a unit. Regions can be collapsed and expanded to hide or display parts of the script. It is also possible to nest regions within other regions. Regions are delimited by `--region` and `--endregion` comments, respectively, before and after the region. Regions can optionally be given a name, which is entered after the `-- region` delimiter (*see screenshot below*).



To insert a region, select the statement/s to be made into a region, right-click, and select **Insert Region**. The expandable/collapsible region is created. Add a name if you wish. In the screenshot above, also notice the line-numbering. To remove a region, delete the two `--region` and `--endregion` delimiters.

Insertion of comments in an SQL script

Text in an SQL script can be commented out. These portions of the script are skipped when the script is executed.

- To comment out a block, mark the block, right-click, and select **Insert/Remove Block Comment**. To remove the block comment, mark the comment, right-click and select **Insert/Remove Block Comment**.
- To comment out a line or part of a line, place the cursor at the point where the line comment should start, right-click, and select **Insert/Remove Line Comment**. To remove the line comment, mark the comment, right-click and select **Insert/Remove Line Comment**.

Use of bookmarks

Bookmarks can be inserted at specific lines, and you can then navigate through the bookmarks in the document. To insert a bookmark, place the cursor in the line to be bookmarked, right-click, and select **Insert/Remove Bookmark**. To go to the next or previous bookmark, right-click, and select **Go to Next Bookmark** or **Go to Previous Bookmark**, respectively. To remove a bookmark, place the cursor in the line for which the bookmark is to be removed, right-click, and select **Insert/Remove Bookmark**. To remove all bookmarks, right-click, and select **Remove All Bookmarks**.

Query Pane: Working with Queries

After connecting to a database, an SQL script can be entered in the SQL Editor and executed. This section describes:

- How an SQL script is entered in the SQL Editor.
- How the script is executed in the Database Query window.

The following icons are referred to in this section:

	Execute Query	Executes currently selected SQL statement. If script contains multiple statements and none is selected, then all are executed.
	Execute for Data Editing	Same as for Execute command, except that results (in Results tab) are editable.
	Import SQL File	Opens an SQL file in the SQL Editor.

Creating SQL statements and scripts in the SQL Editor

The following GUI methods can be used to create SQL statements or scripts:

- *Drag and drop:* Drag an object from the Browser pane into the SQL Editor. An SQL statement is generated to query the database for that object.
- *Context menu:* Right-click an object in the Browser pane and select **Show in SQL Editor | Select**.
- *Manual entry:* Type SQL statements directly in SQL Editor. The Auto-completion feature can help with editing.
- *Import an SQL script:* Click the **Import SQL File** icon in the toolbar of the Database Query window.

Executing SQL statements

If the SQL script in the SQL Editor has more than one SQL statement, select the statement to execute and click either the **Execute** icon or **Execute with Data Editing** icon in the toolbar of the Database Query window. If no statement in the SQL script is selected, then all the statements in the script are executed. The database data is retrieved and displayed as a grid in the Results tab. If **Execute with Data Editing** was selected, then the retrieved data in the Result Grid can be edited. Messages about the execution are displayed in the Messages tab.

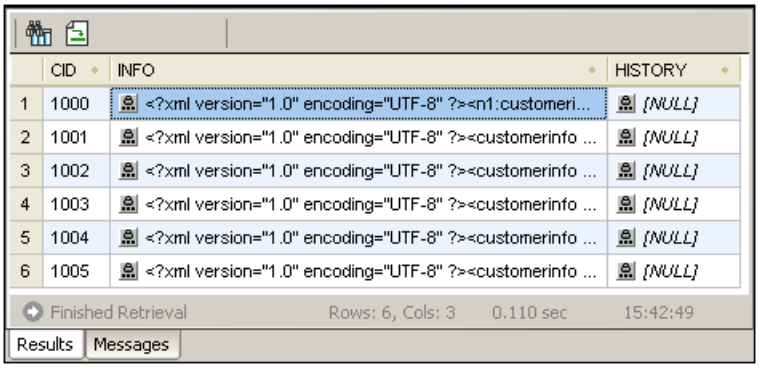
Results and Messages

The Results/Messages pane has two tabs:

- The Results tab shows the data that is retrieved by the query.
- The Messages tab shows messages about the query execution.

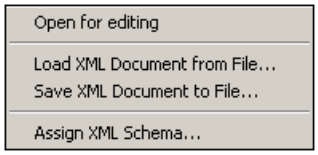
Results tab

The data retrieved by the query is displayed in the form of a grid in the Results tab (*screenshot below*). When the query results have been generated using the **Execute Query** command, the XML documents in the Results tab are indicated with the XML icon (*screenshot below*). If the **Execute for Data Editing** command was used, XML documents are shown with the Editable XML icon.



The following operations can be carried out in the Results tab, via the context menu that pops up when you right-click in the appropriate location in the Results tab:

- *Sorting on a column:* Right-click anywhere in the column on which the records are to be sorted, then select **Sorting | Ascending/Descending/Restore Default**.
- *Copying to the clipboard:* This consists of two steps: (i) selecting the data range; and (ii) copying the selection. Data can be selected in several ways: (i) by clicking a column header or row number to select the column or row, respectively; (ii) selecting individual cells (use the **Shift** and/or **Ctrl** keys to select multiple cells); (iii) right-clicking a cell, and selecting **Selection | Row/Column/All**. After making the selection, right-click, and select **Copy Selected Cells**. This copies the selection to the clipboard, from where it can be pasted into another application.
- *Appending a new row:* If the query was executed for editing, right-click anywhere in the Results pane to access the **Append row** command.
- *Deleting a row:* If the query was executed for editing, right-click anywhere in a row to access the **Delete row** command.
- *Editing records:* If the query was executed for editing, individual fields can be edited. To commit changes, click the **Commit** button in the toolbar of the Results tab.
- *Editing XML records:* This feature is supported for IBM DB2, SQLServer, PostgreSQL (8.3 and higher), and Oracle (9 and higher) databases, and only for those DB tables that have a primary key. If the query was executed for editing and an editable field is an XML field, clicking the Editable XML icon in the Result Grid opens the Edit XML menu (*screenshot below*). An XML field can also be opened for data editing by right-clicking the XML filed in the Folders pane and selecting the command **Edit Data**.



The **Open for Editing** command opens the XML document in an XMLSpy window, and the Editable XML icon changes to , in which the three dots are red. When this document is saved and if the Auto-Commit XML Changes icon in the Query Database toolbar was selected when the document was opened, the changes to the XML document are committed automatically to the database. Otherwise, saved changes will have to be committed using the Commit button of the Results pane. (Note that to toggle between the XML document window and the Database Query window, you must click the **DB | Query Database** command.) The **Load XML Document from File** command loads an external XML document to the selected field in the database. The **Save XML Document to File** saves the XML document in the selected database field to a file location you choose. The Assign XML Schema command pops up the Choose XML Schema dialog, in which you can select an XML Schema to assign to the XML document. This assignment is saved to the database. XML Schema assignment is explained in more detail in the section, IBM DB2 | Assign XML Schema.

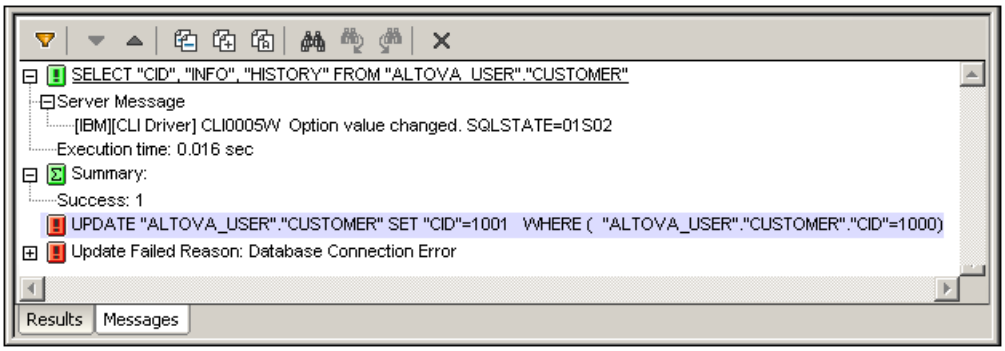
- *Set NULL, Set default, Undo changes for this cell:* If the query was executed for editing, right-clicking in a cell provides access to commands that enable you to set a `NULL` value or, if defined, a column default value for that cell. Changes made to a cell can be undone with the **Undo changes for this cell** command; the current edited value is replaced by the value currently in the DB.

The Results tab has the following toolbar icons:

	Go to Statement	Highlights the statement in the SQL Editor that produced the current result.
	Find	Finds text in the Results pane. XML document content is also searched.
	Add New Line	Adds a new row to the Result Grid.
	Delete Row	Deletes the current row in the Result Grid.
	Undo Changes to Result Grid	Undoes all changes to the Result Grid.
Commit	**Commit**	Commits changes made in the Result Grid to the database.

Messages tab

The Messages tab provides information on the previously executed SQL statement and reports errors or warning messages.



The toolbar of the Messages tab contains icons that enable you to customize the view, navigate it, and copy messages to the clipboard. The **Filter** icon enables the display of particular types of messages to be toggled on or off. The **Next** and **Previous** icons move the selection down and up the list, respectively. Messages can also be copied with or without their child components to the clipboard, enabling them to be pasted in documents. The **Find** function enables you to specify a search term and then search up or down the listing for this term. Finally, the **Clear** icon clears the contents of the Messages pane.

Note: These toolbar icon commands are also available as context menu commands.

26.9.2 IBM DB2

The **IBM DB2** menu item rolls out a submenu containing commands (i) to register and unregister schemas with an IBM DB2 database (**Manage XML Schemas**), and (ii) to assign schemas for XML file validation (**Assign XML Schema**).

Both these mechanisms require that you connect to the required IBM DB2 database. For a connection example, see Connecting to IBM DB2 (ODBC). In this section the focus is on how to manage schemas in an IBM DB2 database and how to assign XML Schemas to a DB XML file.

Note: The Result Grid of the Database Query window provides important functionality for working with XML files in IBM DB2 databases. This functionality includes the ability to open files for editing, loading XML files into a DB XML files, saving DB XML files externally, and assigning XML Schemas to DB XML files.

Manage XML Schemas

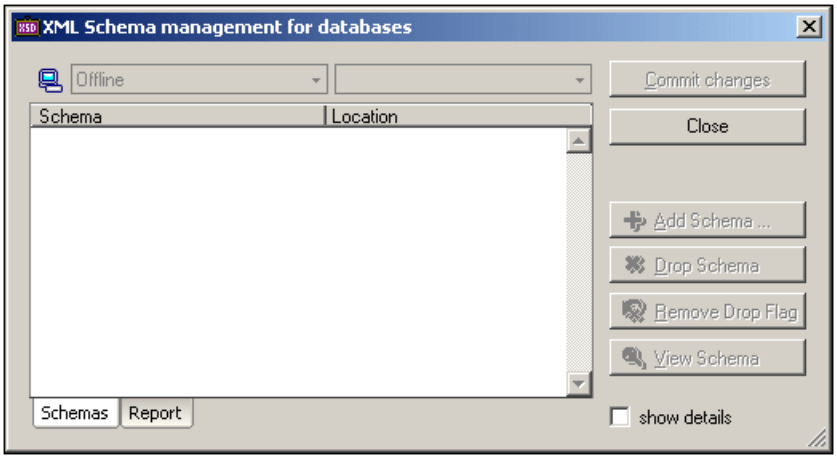
The **Manage XML Schemas** feature enables schemas to be added to and dropped from individual database schemas in an IBM DB2 database. To manage schemas, you have to do the following:

- Connect to the IBM DB2 database
- Select the database schema for which XML Schemas need to be added or dropped
- Carry out the schema management actions.

These steps are described in detail below.

Connecting to the IBM DB2 database

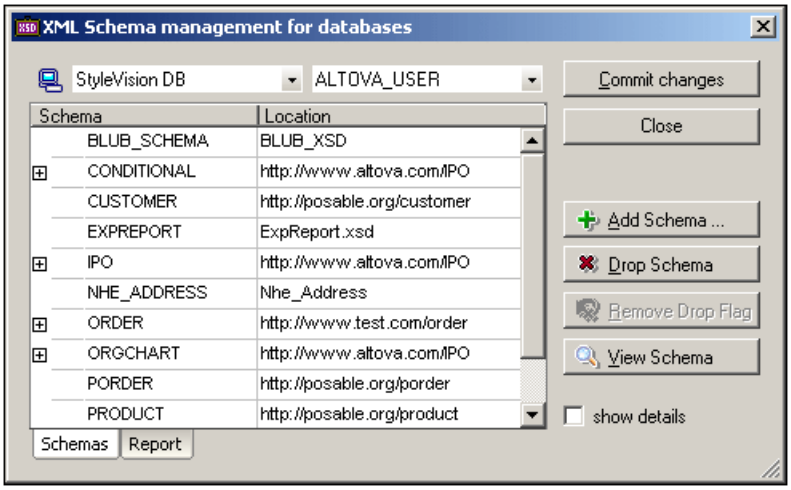
Clicking the **Manage XML Schemas** command pops up the XML Schema Management for Databases dialog (*screenshot below*).



The first thing to do if there is no connection to the required database is to connect to it. If a connection already exists, it appears in the Database combo box. To start the connection process, click the Quick Connect icon in the dialog. This pops up the Quick Connect dialog, through which you can make the connection to the database (for instructions, see Connecting to a Database).

Displaying the list of XML Schemas

After the connection to the IBM DB2 database has been established, the database is listed in the combo box at left (*see screenshot below*). If more than one connection is currently open, you can select the required database in this combo box. In the screenshot below, the StyleVision DB database is selected.



The combo box at right lists all the database schemas of the currently selected IBM DB2 database. When a database schema is selected in this combo box, all the XML Schemas registered for the selected database schema are displayed in the main pane. In the screenshot above, all the XML Schemas registered with the `Altova_User` database schema are listed, together with their locations. Checking the Show Details check box causes additional information columns to be displayed in the main pane.

Managing the XML Schemas

The list of schemas in the main pane represents the schemas registered for the selected database schema. After the list of XML Schemas is displayed, you can add schemas to the list or

drop (delete) schemas from the list.

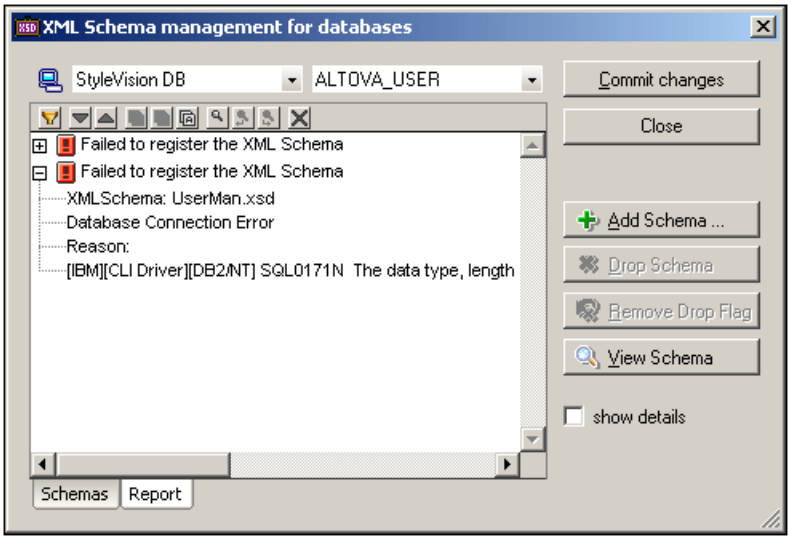
To add a schema, click the **Add** button, browse for the required schema file, and select it. The selected schema file is added to the list in the main pane. Clicking the **Commit Changes** button registers the newly added schema with the database schema.

To drop a schema, select the schema in the main pane and click the **Drop Schema** button. A Drop Flag is assigned to the schema, indicating that it is scheduled for dropping when changes are next committed. The Drop Flag can be removed by selecting the flagged schema and clicking the **Remove Drop Flag** button. When the **Commit Changes** button is clicked, all schemas that have been flagged for dropping will be unregistered from the database schema.

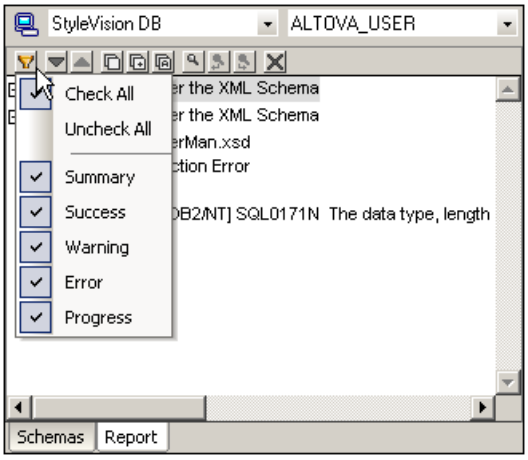
Clicking the View Schema button opens the schema in XMLSpy. To close the XML Schema Management dialog, click the **Close** button.

Reports

When the **Commit Changes** button is clicked, the database is modified according to the changes you have made. A report of the Commit action is displayed in the Report pane (*screenshot below*), enabling you to evaluate whether the success of the action and to debug possible errors. Each subsequent report is displayed below the previous report.



The report pane has a toolbar containing icons that enable you to customize the display of the report listing, navigate the listing, copy report messages, search for text, and clear the pane (*see screenshot below*).



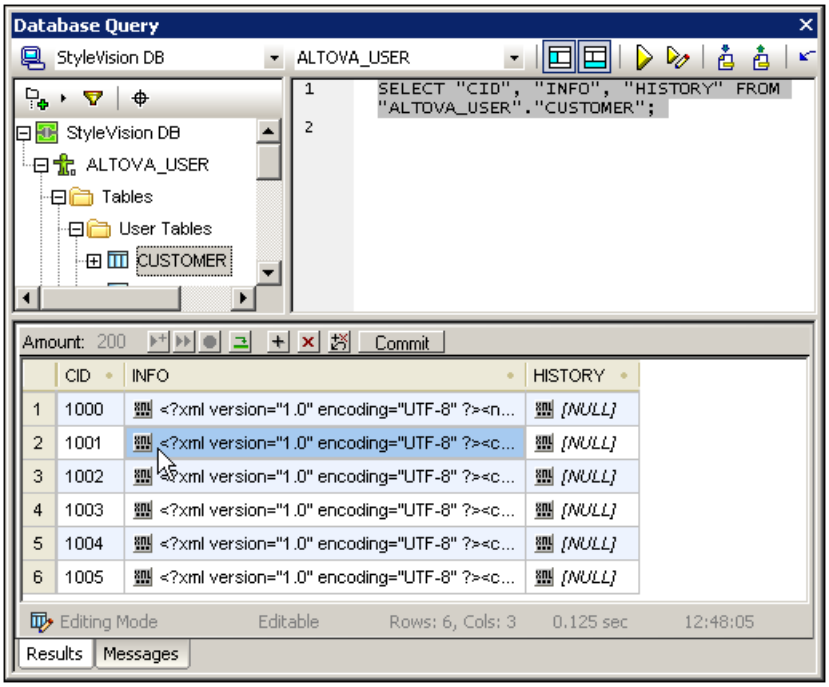
The **Filter** icon enables the display of particular types of messages to be toggled on or off. The **Next** and **Previous** icons move the selection down and up the list, respectively. Messages can also be copied with or without their child components to the clipboard, enabling them to be pasted in documents. The **Find** function enables you to specify a search term and then search up or down the listing for this term. Finally, the **Clear** icon clears the contents of the Report pane.

Assign XML Schema

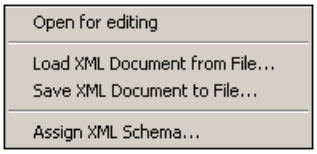
The Assign XML Schema assigns a schema to an XML file opened for editing via the Result Grid of the Database Query window. After the assignment is made, the XML file can be validated against the assigned schema. The assignment is written to the DB when the XML file is saved in XMLSpy.

Opening a DB XML file for editing

In the Database Query window, when a query is addressed to an XML DB and the query is executed for data editing, the Result Grid at the bottom of the Database Query window provides access to the XML files in the database so these can be edited (*see screenshot below*).



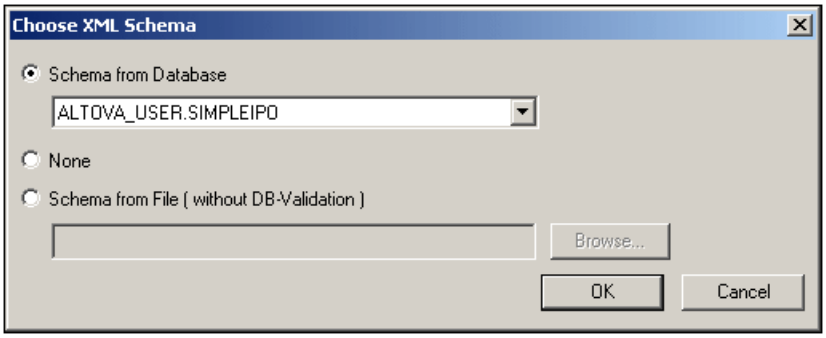
Clicking the XML icon pops up the following menu.



Selecting the **Open for Editing** command opens the XML document in XMLSpy, where it can be edited.

Assigning a schema to the DB XML file

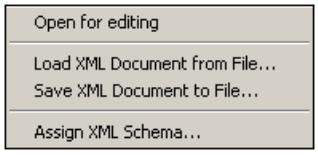
It is when the DB XML file is opened for editing in XMLSpy that the **IBM DB2 | Assign XML Schema** command is enabled. With the XML document active in XMLSpy, clicking the **Assign XML Schema** command pops up the Choose XML Schema dialog (*screenshot below*).



A schema can be selected from among those stored in the database (these are listed in the dropdown list of the Schema from Database combo box), or from among external files that can be browsed. Clicking **OK** assigns the schema to the XML file. Note that the assignment is not written into the XML file. When the XML file is saved in XMLSpy—and if the Auto-Commit XML changes

icon in the Query Database toolbar was selected when the document was opened—then the schema assignment is saved to the database. Note that the schema assignment is written to the database—and not to the XML file.

Note: The Edit XML menu in the Result Grid of the Database Query window also has an **Assign XML Schema** command (*see screenshot below*), which also assigns a schema to the DB XML file.



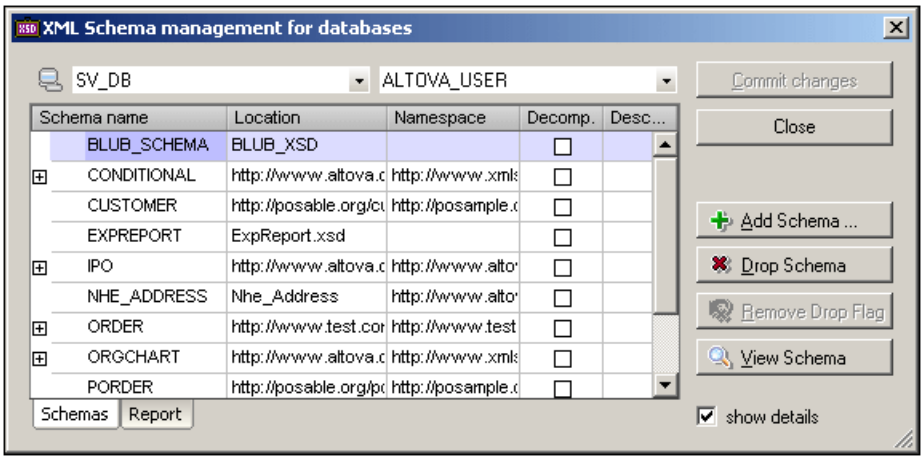
The difference between the two Assign XML Schema commands is that the command in the **DB | IBM DB2** menu enables you to assign an XML Schema while you are editing the XML file thereby allowing you to change schema assignments while editing the XML document and to validate the XML document immediately.

26.9.3 SQL Server

The **SQL Server** menu item rolls out a submenu containing the Manage XML Schemas command.

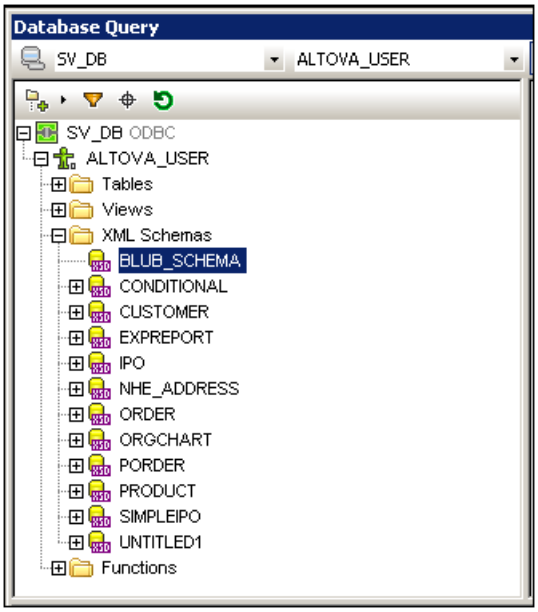
Manage XML Schemas

XML Schema management for databases enables you to add and delete XML Schemas from the schema repository of an XML database. After connecting to the database, XMLSpy provides the XML Schema Management for Databases dialog, in which XML Schemas can be managed.



The dialog box provides a Quick Connect icon which calls the Quick Connect wizard to connect to a data source. If more than one connection currently exists, the required connection can be selected from the combo box on the to left-hand side. The required root object can then be selected from the right-hand side combo box. All the XML Schemas currently in the repository for that root object are displayed in the dialog box. The name, location, and namespace of each schema are listed, as well as the option for decomposition..

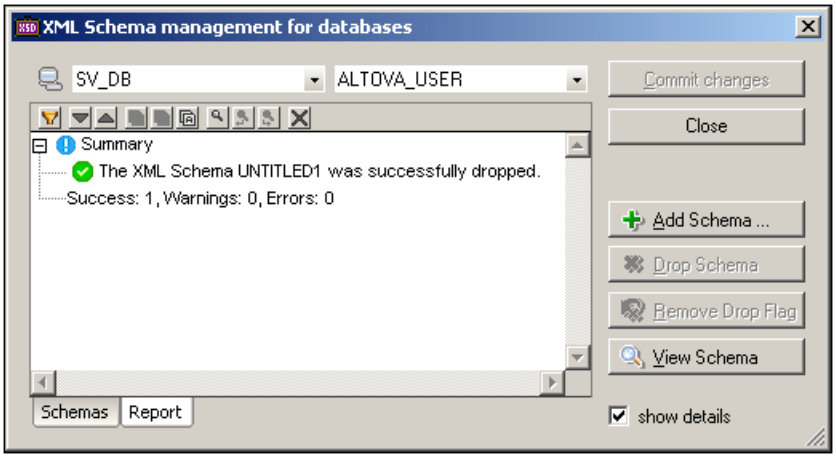
Note that the stored schemas can also be viewed in the Database Query window (screenshot below), but they cannot be managed there. To manage schemas, use the XML Schema Management for Databases dialog.



In the XML Schema Management dialog you can do the following:

- Add a schema using the **Add Schema** button. The selected schema will be appended to the list and marked for addition.
- Mark schemas in the list for deletion with the **Drop Schema** button. The Drop flag can be removed with the **Remove Drop Flag** button.
- Open a selected schema in Schema View by clicking the **View Schema** button.
- Commit the addition and drop (deletion) changes with the **Commit Changes** button.

After changes have been committed, a report of the commit action can be viewed in the Report tab (screenshot below).



26.9.4 Oracle XML DB

XMLSpy allows you to connect to and query Oracle XML Db databases.

The following database functions are supported:

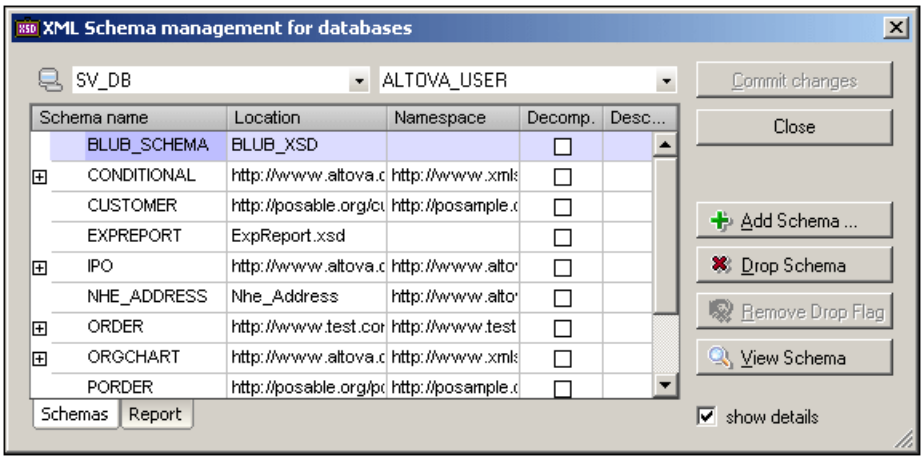
- Add (and register) an XML schema to the Oracle XML Db. The Oracle XML DB client must be installed for you to be able to register XML schemas through XMLSpy.
- Open and delete schemas
- Query the database using XPath statements (DBUri)
- Browse XML documents (using WebDAV)
- Create an XML document based on a schema saved in the database

General installation process:

- Download and install XMLSpy
- Install Oracle server (if necessary)
- Create an Oracle database

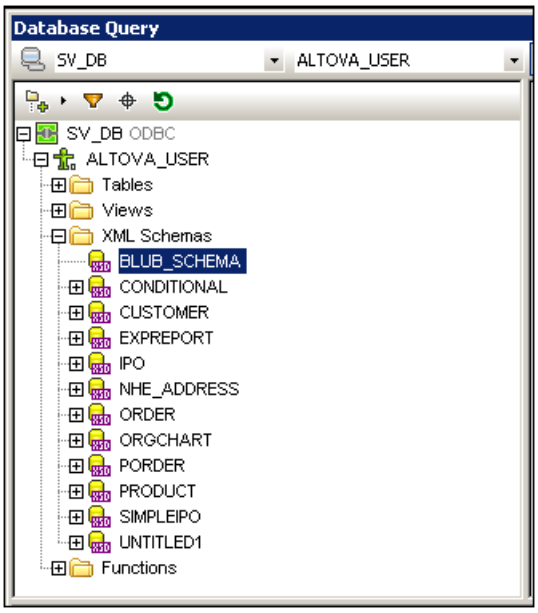
Manage XML Schemas

XML Schema management for databases enables you to add and delete XML Schemas from the schema repository of an XML database. After connecting to the database, XMLSpy provides the XML Schema Management for Databases dialog, in which XML Schemas can be managed.



The dialog box provides a Quick Connect icon which calls the Quick Connect wizard to connect to a data source. If more than one connection currently exists, the required connection can be selected from the combo box on the to left-hand side. The required root object can then be selected from the right-hand side combo box. All the XML Schemas currently in the repository for that root object are displayed in the dialog box. The name, location, and namespace of each schema are listed, as well as the option for decomposition..

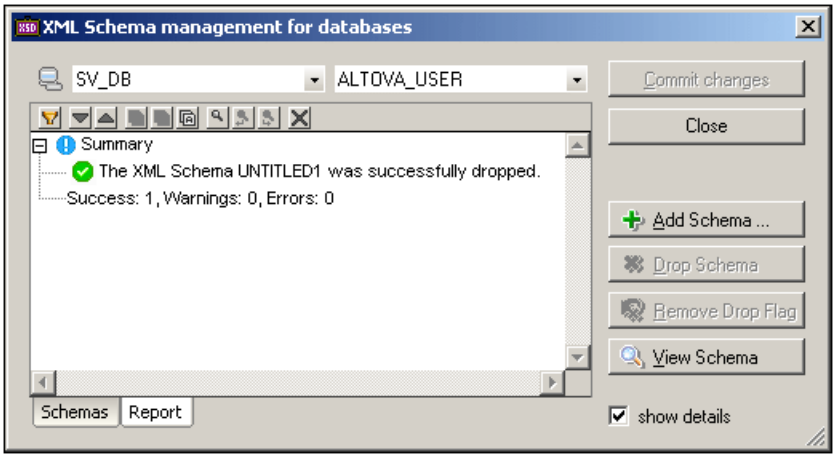
Note that the stored schemas can also be viewed in the Database Query window (screenshot below), but they cannot be managed there. To manage schemas, use the XML Schema Management for Databases dialog.



In the XML Schema Management dialog you can do the following:

- Add a schema using the **Add Schema** button. The selected schema will be appended to the list and marked for addition.
- Mark schemas in the list for deletion with the **Drop Schema** button. The Drop flag can be removed with the **Remove Drop Flag** button.
- Open a selected schema in Schema View by clicking the **View Schema** button.
- Commit the addition and drop (deletion) changes with the **Commit Changes** button.

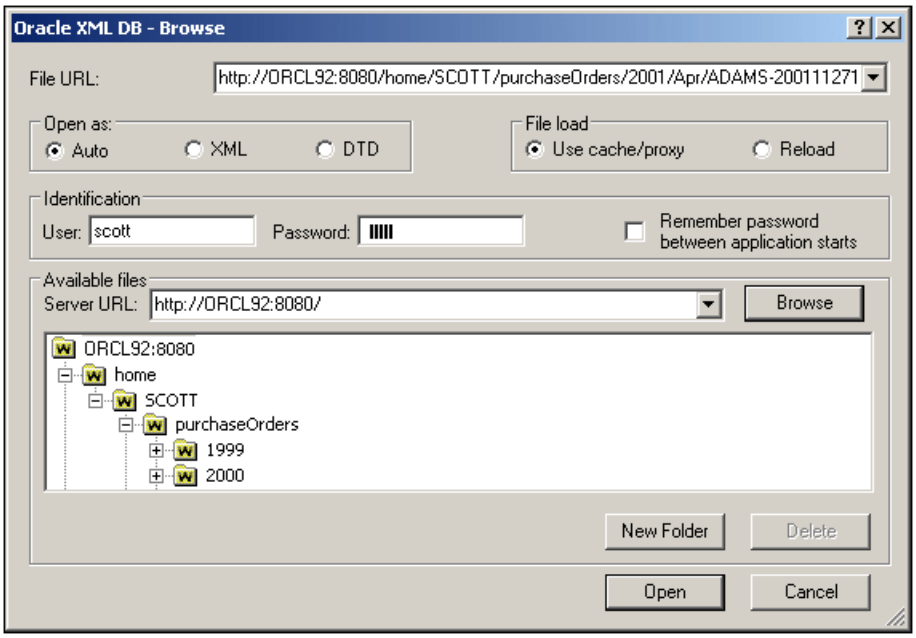
After changes have been committed, a report of the commit action can be viewed in the Report tab (screenshot below).



Browse Oracle XML documents

This command allows you to browse the XML documents available on your server. The server details are automatically filled in if you previously queried the database or listed schemas. If this

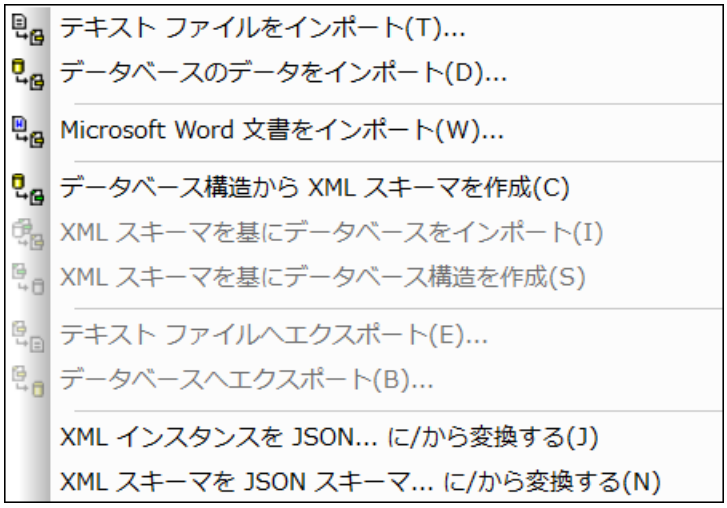
is not the case, then you have to enter them manually.



Use the tree view to find specific XML files. Double clicking a file in the tree view opens it. You can also click a file and click **Open** to achieve the same thing. The **New Folder** button adds a new folder, the **Delete** button deletes the currently selected XML file.

26.10 Convert Menu

The **Convert** menu (*screenshot below*) provides powerful data exchange functionality between data formats:

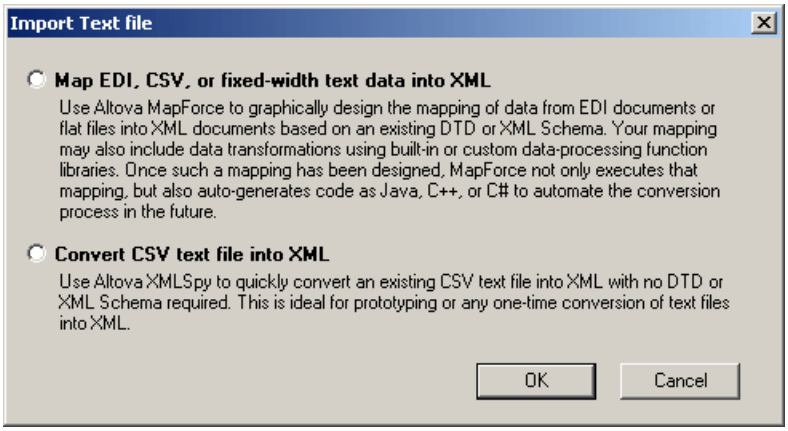


- Import and export text, word processor, database, and XML files.
- Import database data based on an existing XML Schema.
- Create an XML Schema based on the structure of an existing database.
- Create a **database structure**, based on an existing XML schema.
- Convert between XML instances and JSON instances, and between XML Schemas and JSON Schemas.

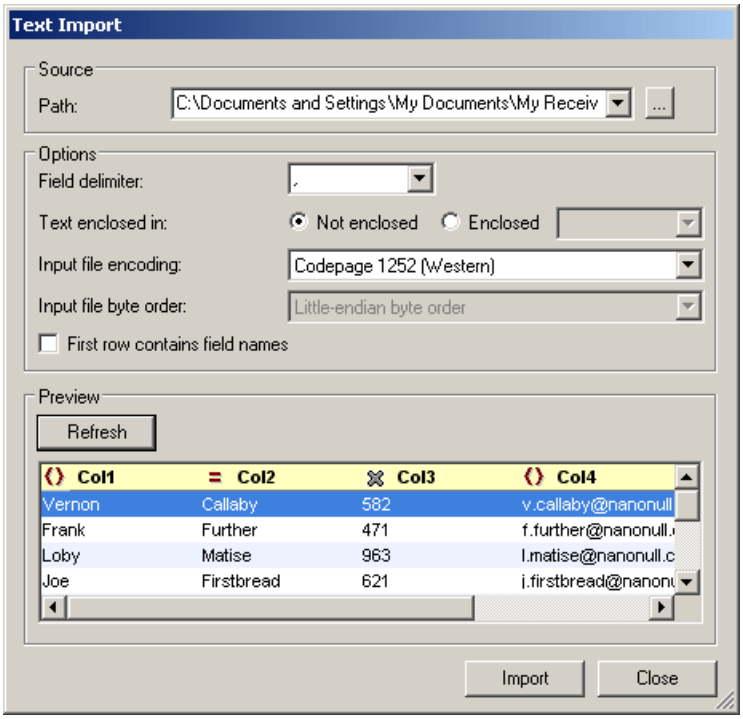
26.10.1 Import Text File

This command lets you **import** any **structured text** file into XMLSpy and convert it to XML format immediately. This is useful when you want to import legacy data from older systems. The steps for importing data in a text file as an XML document are described below.

1. Select the menu item **Convert | Import Text File**. The following dialog appears:



2. Select one of the following:
   - Map EDI, CSV, or fixed-width text data into XML (you must have installed Altova MapForce in order to select this option)
   - Convert CSV text file into XML
3. Click **OK**. The Text import dialog appears (*screenshot below*). How to use this dialog is described below.



Path
Enter the path to the file to import in the Path text box, or select the file using the Browse button to the right of the text box. After the file is selected, a Grid View preview of the XML file is displayed in the Preview pane. Any change in the options selected in this dialog will be reflected in the preview immediately.

Delimiter
To successfully import a text file, you need to specify the field delimiter that is used to separate columns or fields within the file. XMLSpy will auto-detect common row separators

(CR, LF, or CR+LF).

String quotes
Text files exported from legacy systems sometimes enclose textual values in quotes to better distinguish them from numeric values. If this is the case, you can specify what kind of quotes are being used in your file, and remove them automatically when the data is imported.

Encoding
The data is converted into Unicode (the basis of all XML documents), so you need to specify which character-set the file is currently encoded in. For US or Western European Windows systems this will most likely be Codepage 1252, also referred to as the ANSI encoding.

Byte order
If you are importing 16-bit or 32-bit Unicode (UCS-2, UTF-16, or UCS-4) files, you can also switch between little-endian and big-endian byte order.

First row contains column names
It is also very common for text files to contain the field names in the first row within the file. If this is the case, check this check box.

Preview
In the Preview pane you can rename column headers by clicking in a name and editing it. The column headers will be the element or attribute names in the XML document. You can also select whether a column should be an element or an attribute in the XML document, or whether it should not be imported into the XML document. Click the column-type icon in each column header to toggle through these options. In the screenshot above, `Col1` is an element, `Col2` is an attribute, and `Col3` will not be imported.

When you are satisfied with the options, click **Import**. The imported data is converted into an XML document that is displayed in Grid View.

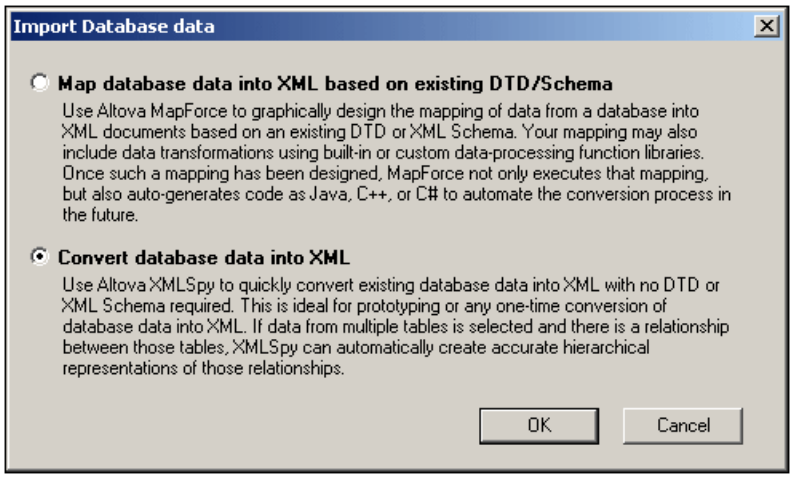
26.10.2 Import Database Data

The **Import Database Data** command enables you to import data from any of a variety of databases into an XML file. The import mechanism involves two steps:

1. A connection to the database is established. For instructions, see Connecting to a Database.
2. The data to be imported is selected.

To import database data, do the following:

1. When you click the Import Database Data command, the Import Database Data dialog (screenshot below) pops up.

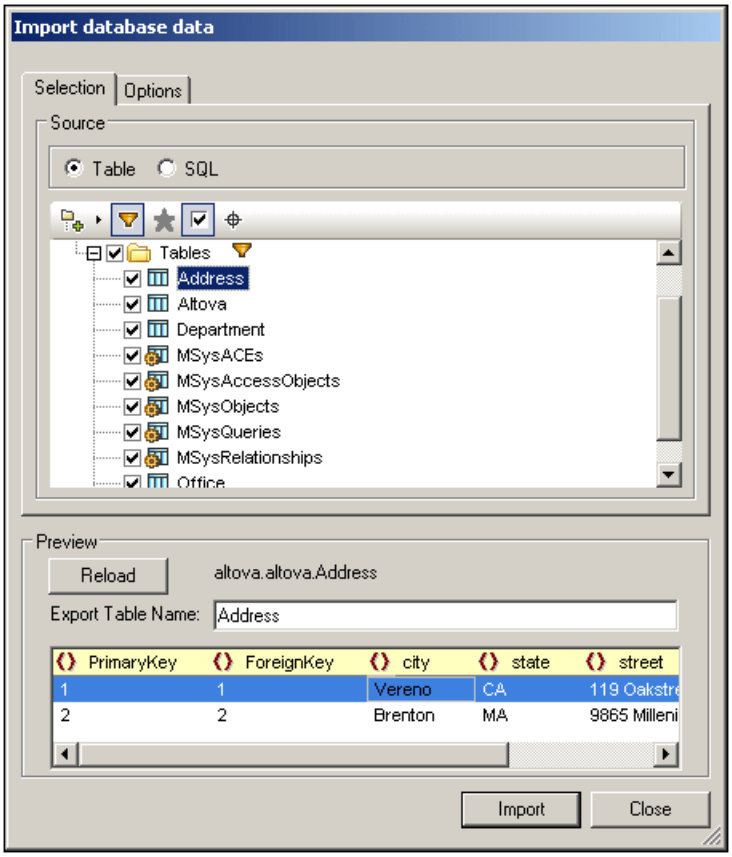


2. Select Convert Database Data into XML and click **OK**. The Connect to a Data Source dialog appears. (The Map Database Data into XML Based on Existing DTD/Schema option requires the use of Altova MapForce to carry out the mapping.)
3. In the Connect to Data Source dialog, you establish a connection to the database. For instructions, see Connecting to a Database.
4. After the connection to the database is established, the Import Database Data dialog displays tabs and windows that enable you to select the database data to import. These options are described below. After finishing, click the **Import** button to import the database data.

Data selection and import options

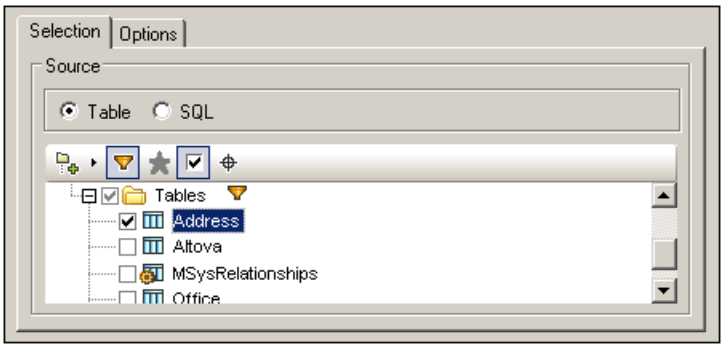
The Import Database Data dialog for setting the selection and import options consists of two parts (*shown separately in the screenshots below*):

- an upper part with two tabs: (i) Selection, and (ii) Options.
- a lower part, which is a Preview window showing the data according to the data selection and import options.



Selection tab

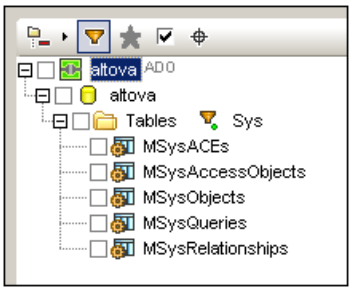
In the Selection tab (*screenshot above*), the Source pane (*screenshot below*) displays either a representation of the tables of the database or an editable SQL statement for selecting the required tables, each view being selected by clicking the respective radio button.



In the Table view, you can select the tables in the database that you want to import by checking the table's check box (*screenshot above*). The contents of the table can then be displayed in the Preview pane. The table selection can be further filtered in the Preview pane (*see below*).

The database structure can be displayed differently and can be filtered. The Layout icon in the Source pane enables you to organize database objects into: (i) folders based on object type; (ii) folders based on object type, but without schema folders; (iii) in a hierarchy, but without folders; and (iv) categories of tables, based on their relationships with other tables.

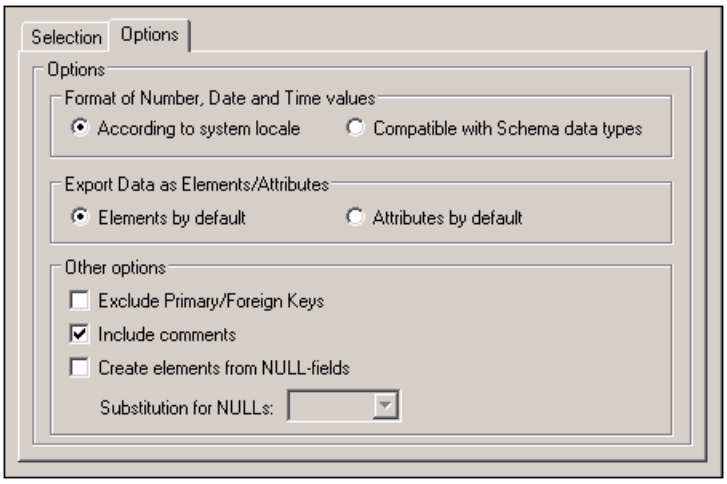
Clicking the Filter Folder Contents icon applies a filter to the selected folder (*in the screenshot below, to the Tables folder*). Clicking the filter icon on the table pops out a menu with a list of filter possibilities. In the screenshot below, the filter has been set to display objects that contain the text `Sys` in its name. The view is filtered accordingly.



Clicking the Object Locator icon pops up a text field, which behaves like a Search entry field. You can enter a text string and the dropdown list will display all the objects whose names contain that text string. Selecting one of these objects from the dropdown list will highlight that object in the tree.

Options tab

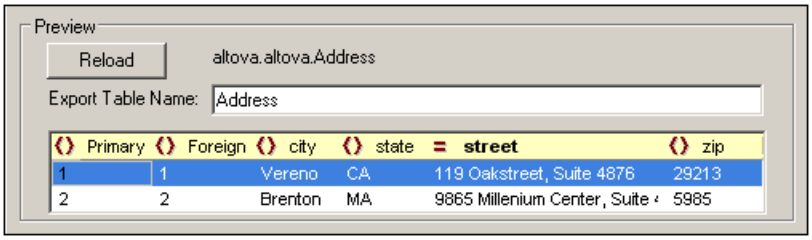
In the Options tab (*screenshot below*), you can specify how number, date, and time values are to be imported; whether data is imported as elements or attributes; and whether comments and `NULL` fields are to be included in the import.



When `NULL` fields are enabled for import, you can enter a substitution XML value for them.

Preview pane

The Preview pane (*screenshot below*) displays the structure of the table currently selected in the Selection tab. When a new table is selected in the Selection tab, click the **Preview** button in the Preview pane to display the table. Click the **Reload** button to refresh the preview.



A field can be specified to be imported as an element or attribute, or not to be imported, by clicking the symbol to the left of the column name. You can click through the element, attribute, and ignore options. In the screenshot above, the `city` field, for example, has been set to be imported as an element while the `street` field has been set to be imported as an attribute.

Datatype conversions

Information about the conversion of database datatypes to XML Schema datatypes is listed in the Appendices.

26.10.3 Import Microsoft Word Document

This command enables the direct **import** of any **Word document** and conversion into XML format if you have been using paragraph styles in Microsoft Word. This option requires Microsoft Word or Microsoft Office (Version 97 or 2000).

When you select this command, the Open dialog box appears. Select the Word document you want to import.

XMLSpy automatically generates an XML document with included CSS stylesheet. Each Word paragraph generates an XML element, whose name is defined as the name of the corresponding paragraph style in Microsoft Word.

26.10.4 Create XML Schema from DB Structure

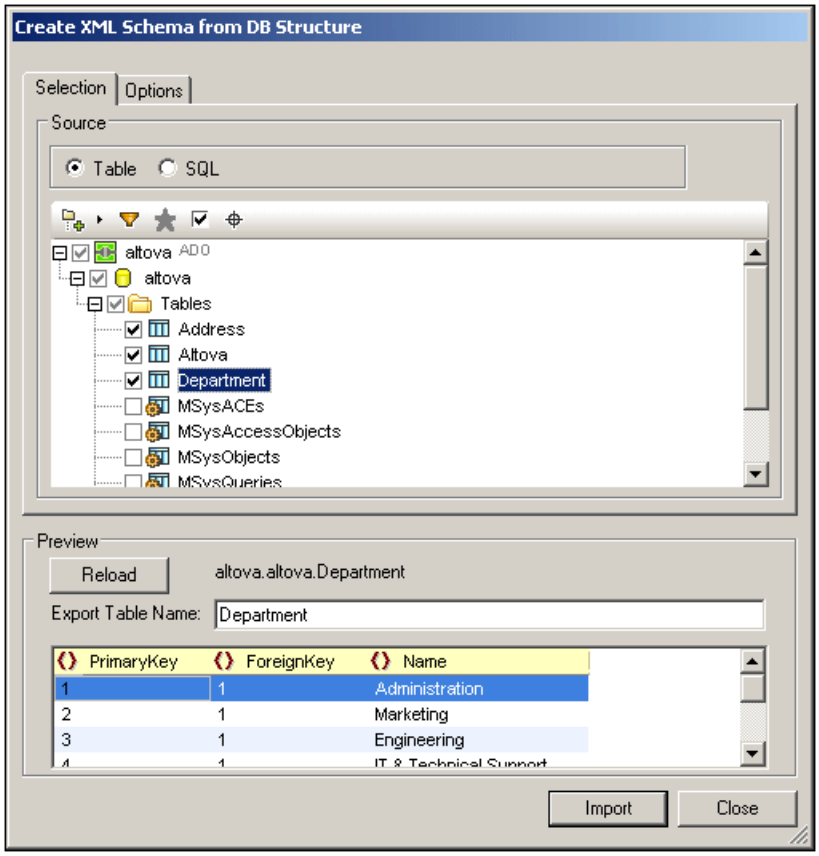
The **Create XML Schema from DB Structure** command enables you to create an XML Schema from the structure of any of a variety of databases. The XML Schema-creation mechanism involves two steps:

1. A connection to the database is established.
2. Options for the database data selection and the XML Schema are specified.

To create an XML Schema from a DB structure you would do the following:

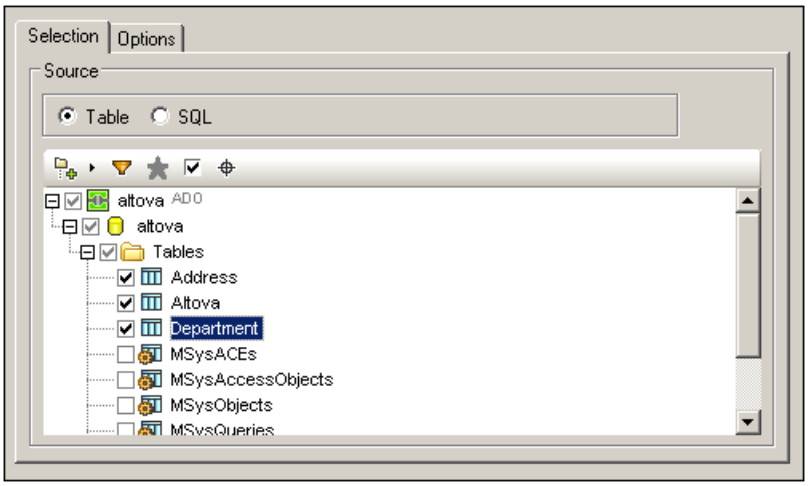
1. When you click the **Create XML Schema from DB Structure** command, the Database Connection Wizard box appears.

2. Using the Database Connection Wizard, you establish a connection to the database (for instructions, see Connecting to a Database).
3. After the connection to the database is established, the Create XML Schema from DB Structure dialog (*screenshot below*) displays tabs and windows that enable you to select the database structure to import. These options are described below. After finishing, click the **Import** button to import the database data.



Selection tab

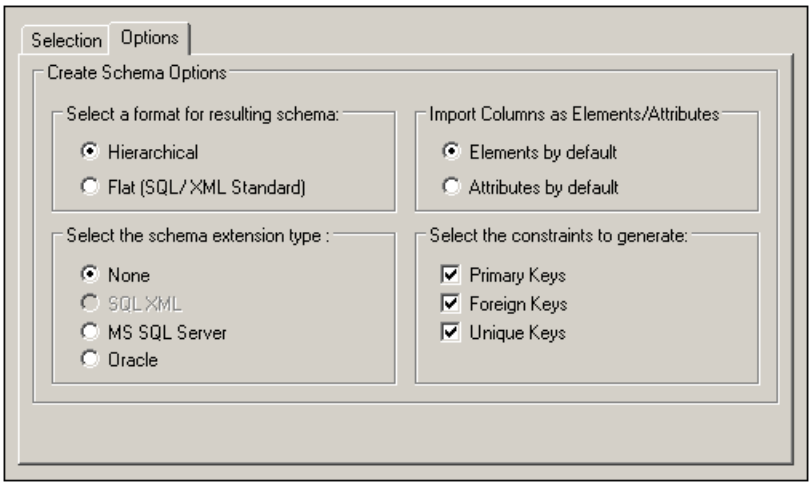
In the Selection tab (*screenshot below*), the data source is listed in the Source Database pane. The Source pane displays either a representation of the tables of the database or an editable SQL statement for selecting the required tables.



In the Table view, you can select the tables in the database that you want to import by checking the table's check box (*screenshot above*). The contents of the table can then be displayed in the Preview pane. The table selection can be further filtered in the Preview pane (*see below*). You can configure the display in the Source pane as described in the section Import Database Data.

Options tab

In the Options tab (*screenshot below*), you can specify the format of the schema, its extension type, whether columns should be imported as elements or attributes, and the database constraints that should be generated in the schema.



Schema format: You can select between a flat (SQL/XML Standard) and a hierarchical schema form

- The **flat** schema model is based on an ISO-ANSI SQL/XML specification INCITS/ISO/IEC 9075-14-2008. The SQL/XML specification defines how to map databases to XML. Relationships are defined in schemas using identity constraints; there are no references to elements. Hence the schema is flat structure which resembles a tree-like view of the database. The specification can be purchased at the ANSI store. For more information, see www.iso.org.
- The **hierarchical** schema model displays the table dependencies visually, in a type of tree view where dependent tables are shown as indented child elements in the content model.

Table dependencies are also displayed in the Identity constraints tab.
Tables are listed as global elements in the schema, and columns are the elements or attributes of these global elements (The user decides whether to map the columns as elements or as attributes). Relationships are created in a hierarchical way so that a foreign key field in one table is actually a reference to the global element that represents that table.

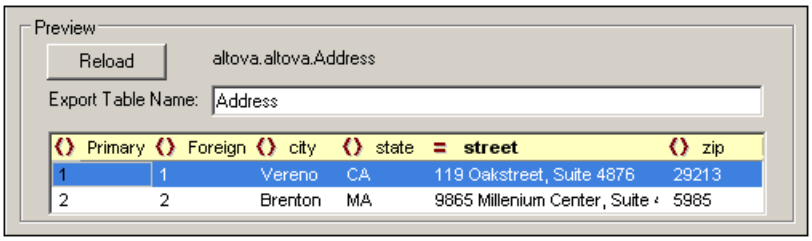
Schema extension type: Schema extension information is additional information read from a database that is then embedded in the schema as either annotation data or attributes. There are four extension type options when generating schemas: (i) no extensions information; (ii) SQL/XML extensions; (iii) MS SQL Server extensions; and (iv) Oracle extensions. These are described below:

- **None**: No additional information is provided by the database.
- **SQL XML:** SQL/XML extensions are only inserted when generating schemas in a flat format. The extension information is stored in annotations and is described in the SQL/XML specification (INCITS/ISO/IEC 9075-14-2008).
- **MS SQL Server:** Selecting Microsoft SQL Server, generates SQL Server extensions. See SQL Server Books Online for resources and MSDN's information about annotating XSD schemas. The following annotation-related elements are generated in the schema: `sql:relation`, `sql:field`, `sql:datatype`, `sql:mapped`.
- **Oracle:** Oracle extensions are selected by default when working with an Oracle database. Additional database information is stored as attributes. Detailed information can be found in Oracle's online documentation. The following subset of attributes is currently generated: `SQLName`, `SQLType`, `SQLSchema`.

Note: Although SQL Server and Oracle extensions can be generated for their respective databases they are not restricted in this way. This proves useful when working with a third database and wanting to generate a schema that later should be working with either SQL Server or Oracle.

Preview pane

The Preview pane (*screenshot below*) displays the structure of the table currently selected in the Selection tab. When a new table is selected in the Selection tab, click the **Reload** button in the Preview pane to refresh the preview.



A field can be specified to be imported as an element or attribute, or not to be imported, by clicking the symbol to the left of the column name. You can click through the element, attribute, and ignore options. In the screenshot above, the `city` field, for example, has been set to be imported as an element while the `street` field has been set to be imported as an attribute.

Datatype conversions

Information about the conversion of database datatypes to XML Schema datatypes is listed in the Appendices.

26.10.5 DB Import Based on XML Schema

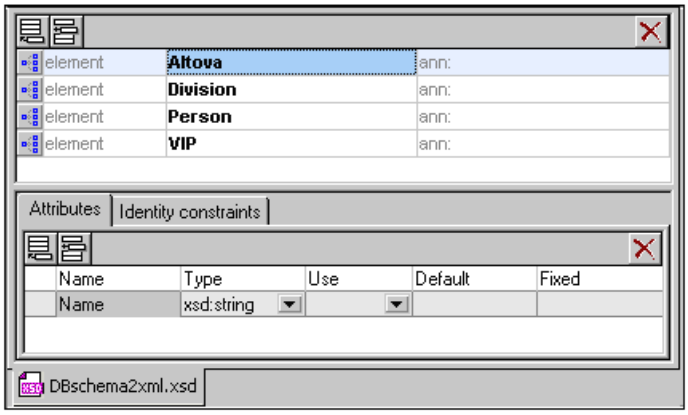
The **DB Import Based on XML Schema** command creates an XML document which is valid according to a given XML Schema and contains data imported from a database. For this feature, the following databases are supported:

- Microsoft Access 2000 and 2003
- Microsoft SQL Server
- Oracle
- MySQL
- Sybase
- IBM DB2

The data to be imported is determined by the table that is selected in the database. With the required XML Schema (that on which you wish to base the import) as the active document in Schema View, connect to the database. Then select the table/s you wish to import, and click Import. The data is imported into an XML document, and the document has the structure of the XML Schema that was active when the data was imported.

In the example below, data from an MS Access database is imported with an XML Schema active in Schema View. These would be the steps to carry out for the import:

1. Open the schema file in Schema View (*screenshot below*).



2. Select the menu command **DB Import based on XML Schema**. This opens the Connect to Data Source dialog.
3. Select the Microsoft Access (ADO) option and click **Next**.
4. Click **Browse** and select the database file. Then click **Next**.
5. In the DB Import Based on XML Schema dialog which pops up, go to the Tables tab, select one or more tables you wish to import (for example, `Altova`), then click **Import**. The table is imported into an XML document that is displayed in Grid View.

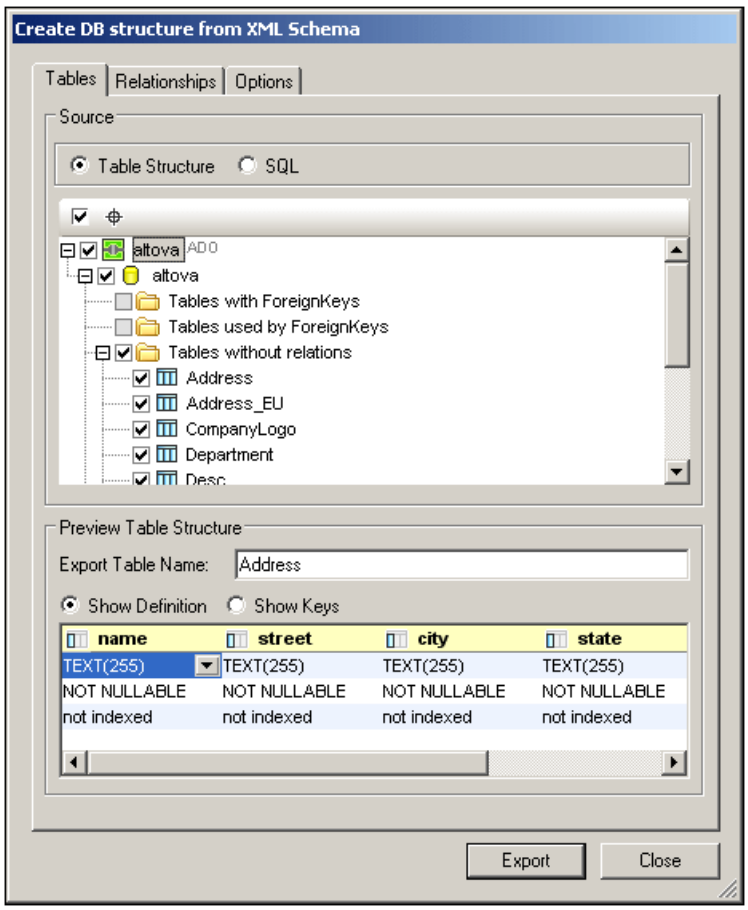
Datatype conversions

Information about the conversion of database datatypes to XML Schema datatypes is listed in the Appendices.

26.10.6 Create DB Structure from XML Schema

XMLSpy allows you to create an empty database (or skeleton database) based on an existing schema file. The method described below is generally the same for each type of database.

1. Open the schema file in Schema/WSDL View
2. Select the menu command **Convert | Create DB Structure from XML Schema**. This pops up the Connect to a Data Source dialog, which enables you to connect to a database (DB).
3. Use the steps described in the section Connecting to a Data Source to connect to the required database. For example, to connect to a Microsoft Access database, select the Microsoft Access radio button, and continue the process to select a database. You can use an existing database or create a new database in which the schema structure will be contained.
4. In the Create DB Structure from XML Schema dialog, tables are created from the schema and displayed in a tree format at the location where they will occur in the DB. For example, in the screenshot below, the Address table is created and selected for export. Tables that should not be exported should be deselected (by unchecking the check box or selecting the appropriate item from the context menu for that table).

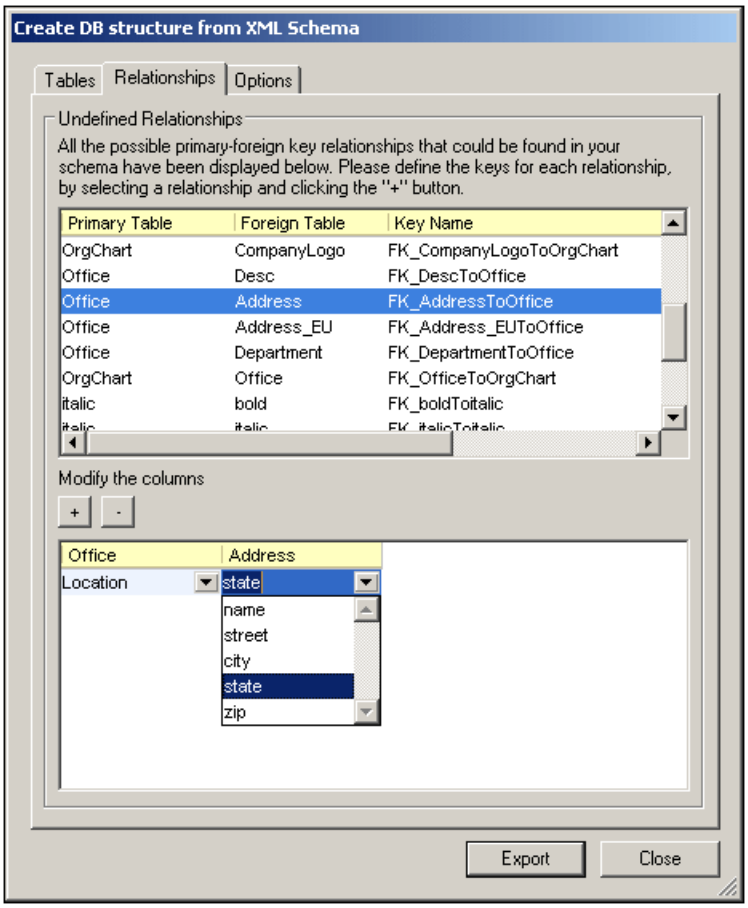


Creating DB tables with relationships

If the XML Schema from which the DB structure is generated has relationships defined in the form of identity constraints, then these relationships are automatically created in the generated

DB structure and displayed in the Table Structure. Tables with relationships are listed under the sections: Tables with ForeignKeys and Tables used by ForeignKeys. Tables without relationships are listed in the Independent Tables section.

In the Relationships tab, you can create and modify table relationships. The tab lists all possible primary-key/foreign-key relationships (*screenshot below*).



To create a relationship, do the following:

1. Select one of the possible primary-key/foreign-key relationships.
2. In the lower pane of the dialog, click the Plus button to create a relationship.
3. Select the required columns in each of the two tables from the respective dropdown lists.

You can also remove a relationship by selecting it and then clicking the Minus button.

Notes on database structure and connecting

The schema structure, defined by the identity constraints, is mirrored in the resulting database. The table below shows the type of database created, the restrictions, and the connecting methods, when using the **Create DB Structure from XML Schema** menu command.

	Directly	using ODBC	using ADO
MS Access (2000 and	OK	OK	OK

2003)	*		
MS SQL Server	OK *	OK	OK
Oracle	OK *	OK	OK
MySQL	-	OK *	OK ⌖
Sybase	-	OK *	OK
IBM DB2	-	OK *	OK

* *Recommended connection method for each database.*
⌖ *MySQL: When creating the ADO connection based on ODBC, it is recommended to use either the User or System DSN.*
\- *Not supported*

XMLSpy will map both hierarchical and flat formatted schemas. XMLSpy recognizes both formats automatically.
The flat format is mapped to SQL in two different ways.

- SQL Server DB, Oracle DB, or Sybase DB:
  A schema that was generated in flat format, for one of the above databases, will have the schema catalog name extracted and used in the generated SQL script as the DB name. This means that the resulting SQL script will be executed on a target DB whose name must be identical to the schema catalog name.
- Access (2000 or 2003), MySQL, or DB2 DB:
  A schema that was generated in flat format, for one of the above databases, will **ignore** the schema catalog name when the SQL script is generated. This means that the resulting SQL script will be executed on a target database that was logged into.

Datatype conversions

Information about the conversion of XML Schema datatypes to database datatypes is listed in the Appendices.

26.10.7 Export to Text Files

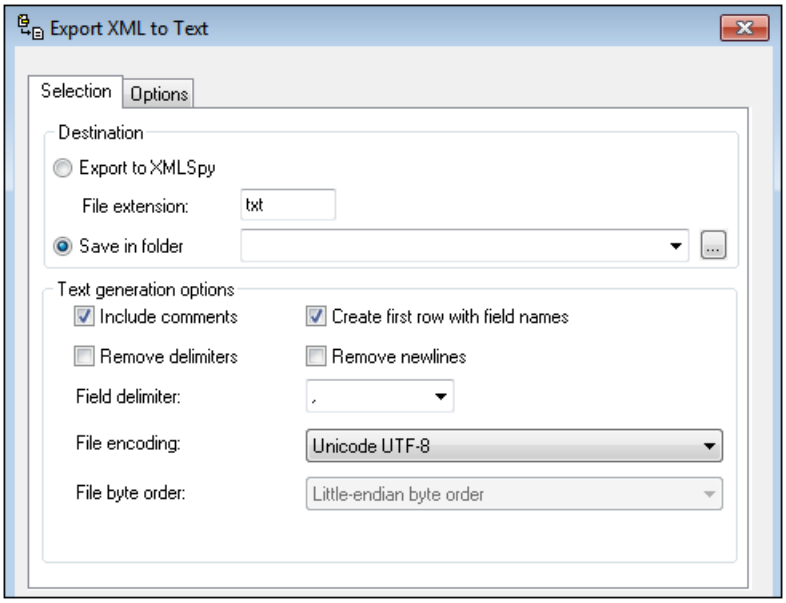
The command **Convert | Export to Text Files** exports XML data into text formats for exchange with databases or legacy systems. On clicking this command, the Export XML to Text dialog pops. It consists of two parts (*shown in separate screenshots below*):

- an upper part with two tabs: (i) Selection, and (ii) Export Options.
- a lower part, which is a Preview window.

After you have selected the desired options in this dialog (*described below*), click the **Export** button to export to text file/s.

Selection

In the Selection tab (*screenshot below*), you can select the destination of the file to be exported and text generation options.



Destination: The exported file can be saved directly to a folder. The file extension can be specified. The filenames will be those of the elements (in the XML file) that will be exported. Alternatively, untitled files can be exported to XMLSpy. These files will be displayed in the GUI, and can be saved later.

Include comments: Activate this option to include an XMLSpy-generated comment in the exported XML file. The comment will contain the SQL query used to select the data as well as a list in which there is one listitem for each column header in the database table.

Create first row with field names: When activated, the exported tables include the names of columns from the database. Otherwise column names will not be included in the exported text file.

Remove delimiters: Removes delimiters that are contained in text values in the exported data. Set the delimiter you want to remove by using the Delimiter combo box in this tab. For example, if this option is activated and the selected delimiter is the apostrophe, when you export the XML value `Ba'ker`, the string will be `Baker` in the exported text.

Remove newlines: Removes newlines from exported data.

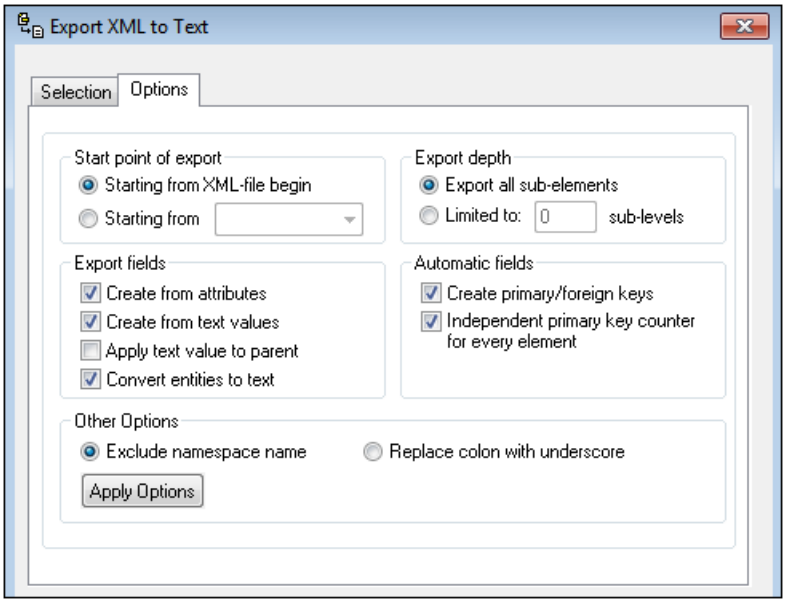
Delimiter: Select from the drop-down list the character that you wish to have removed during export. Alternatively, enter the desired character string.

Encoding: Select from the drop-down list, the desired encoding for files that are generated during export.

Byte order: If you are exporting 16-bit or 32-bit Unicode (UCS-2, UTF-16, or UCS-4) files, you can also switch between little-endian and big-endian byte order.

Export Options

Additional export options, which are described below, can be specified in the Options tab (*screenshot below*):



Start point of export: You can choose to export the entire XML document or restrict your export to the data hierarchy starting from the currently selected element. The number of sub-levels below the start point that will be exported is specified in the *Export Depth* option.

Export depth: Specifies the number of sub-levels below the start point that will be exported.

Export fields: Depending on your XML data, you may want to export only elements, attributes, or the textual content of your elements. Note that you can deselect the export of individual elements in the Preview window.

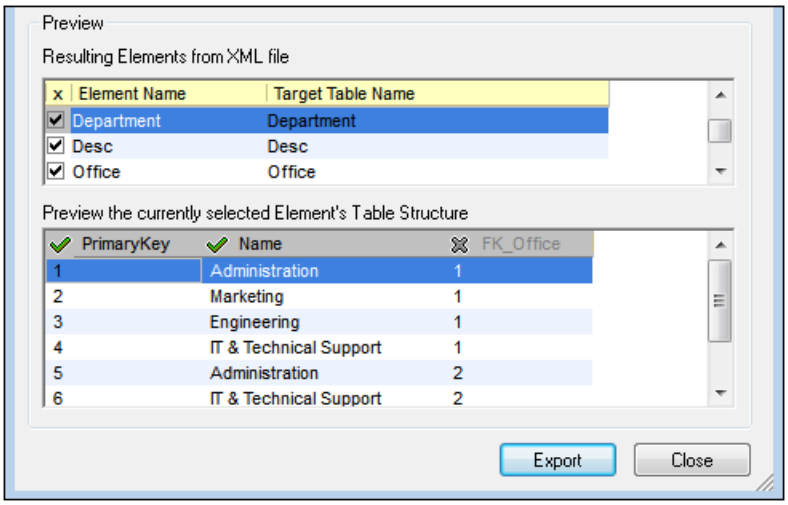
Automatic fields: XMLSpy will produce one output file or table for each element type selected. You can choose to automatically create primary/foreign key pairs to link your data in the relational model, or define a primary key for each element.

Exclude namespace name: Together with the *Replace Colon With Underscore* radio button this is an either/or choice. Specifies whether namespace prefixes of elements and attributes should be excluded or whether the colon in the namespace prefix should be replaced with an underscore.

Apply Options: After you have set options, click this button to apply the options. The preview in the preview pane will be updated with the new options.

Preview window

The Preview window (*screenshot below*) is displayed below the Selection and Options tabs.



The *Resulting Elements from XML File* pane shows the node names that will be exported and the name in the generated file. You can select/deselect nodes that will be exported. When an element is selected, a preview of its structure is shown in a second pane below. In this pane, clicking to the left of a column header name toggles the export of that column on and off. In the screenshot above, the last column (`FK_Office`) has been toggled off.

26.10.8 Export to a Database

The command **Convert | Export to a Database** exports XML data to a database. On clicking this command, the Connection Wizard starts up and enables you to set up a connection to the database you wish to update. After a connection has been established, the Export Data to Database dialog pops. It consists of two parts (*shown separately in the screenshots below*):

- an upper part with two tabs: (i) Selection, and (ii) Export Options.
- a lower part, which is a Preview window.

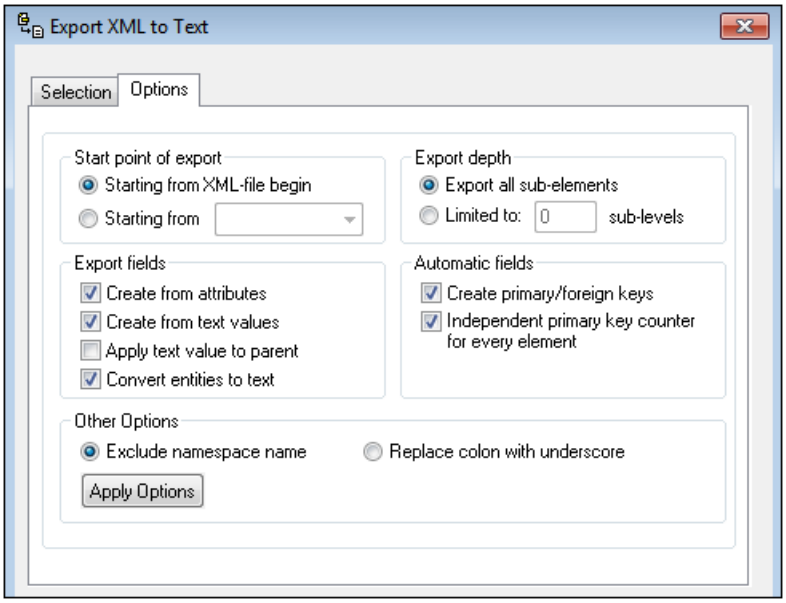
After you have selected the desired options in this dialog (*described below*), click the **Export** button to export to the database.

Selection

In the Selection tab, you can select the destination database and table generation options. The destination field selects the connection to the database. You must select whether the data is created as new tables, updates existing tables, or first tries to update an existing table and then creates a new table if an an update is not possible. You can also set a stop action based on the number of errors, and, optionally, SQL script logging.

Export Options

Export options, which are described below, can be specified in the Options tab (*screenshot below*):



Start point of export: You can choose to export the entire XML document or restrict your export to the data hierarchy starting from the currently selected element. The number of sub-levels below the start point that will be exported is specified in the *Export Depth* option.

Export depth: Specifies the number of sub-levels below the start point that will be exported.

Export fields: Depending on your XML data, you may want to export only elements, attributes, or the textual content of your elements. Note that you can deselect the export of individual elements in the Preview window.

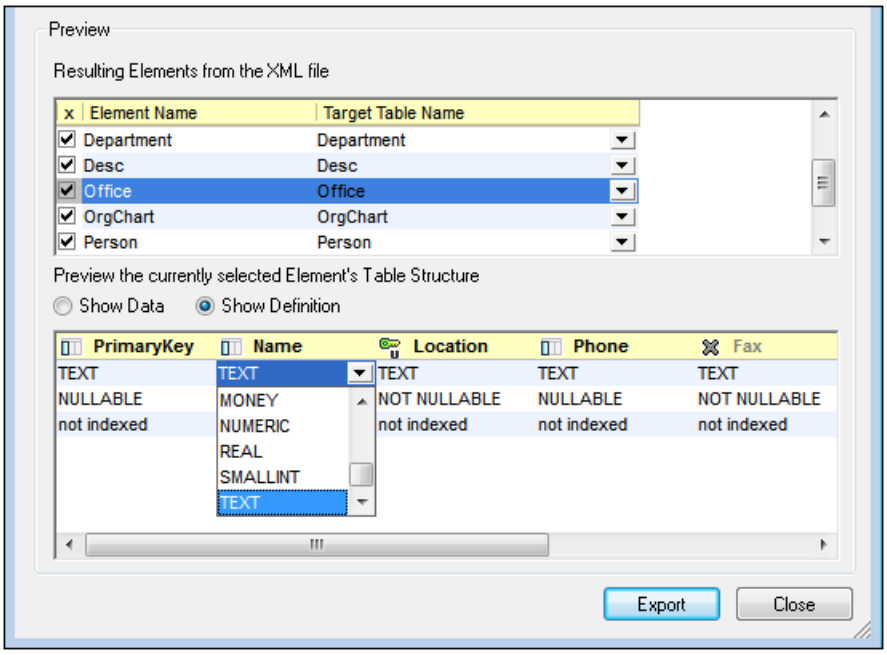
Automatic fields: XMLSpy will produce one output file or table for each element type selected. You can choose to automatically create primary/foreign key pairs to link your data in the relational model, or define a primary key for each element.

Exclude namespace name: Together with the *Replace Colon With Underscore* radio button this is an either/or choice. Specifies whether namespace prefixes of elements and attributes should be excluded or whether the colon in the namespace prefix should be replaced with an underscore.

Apply Options: After you have set options, click this button to apply the options. The preview in the preview pane will be updated with the new options.

Preview window

The Preview window (*screenshot below*) is displayed below the Selection and Options tabs.



The *Resulting Elements from XML File* pane shows the name of the nodes in the XML document that will be exported and its corresponding name in the generated file. You can select/deselect nodes that will be exported. When an element is selected, a preview of its structure in the generated file is shown in a second pane below. This preview can be switched between a preview of: (i) data in the generated structure (*Show Data*); or (ii) definitions of each column in the generated structure (*Show Definition*). The screenshot above shows the column definitions.

In this second pane, clicking to the left of a column name cycles the column through four settings: (i) Include in table structure; (ii) Unique constraint; (iii) Primary Key constraint; (iv) Exclude from table structure. In the screenshot above, the `Location` column has a Unique constraint, while the `Fax` column has been excluded from the table structure. All the other columns are included in the table structure.

When the element's table structure shows field definitions (*Show Definition*), the definitions can be edited by selecting the definition and selecting an option from the definition's combo box (*see screenshot above*).

26.10.9 Convert XML Instance to/ from JSON

If the active document is an XML document, this command generates a JSON document from it. If the active document is a JSON document, the command generates an XML document from it. The generated document is opened in a new window, and can then be saved to any location. Conversion options are described below. For more information about JSON and JSON editing support in XMLSpy, see the section JSON and JSON Schema.

Sample conversions

Given below is an example of a source XML document, and, below it, the JSON document generated by the **Convert XML Instance to/from JSON** command.

XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<Person first="Jim" last="James">
 <Address>
  <street>4 New Street</street>
  <city>New York</city>
  <state>NY</state>
  <code>10123</code>
 </Address>
 <Tel type="home">
   123 123-1234
 </Tel>
 <Tel type="office">
   123 987-9876
 </Tel>
</Person>
```

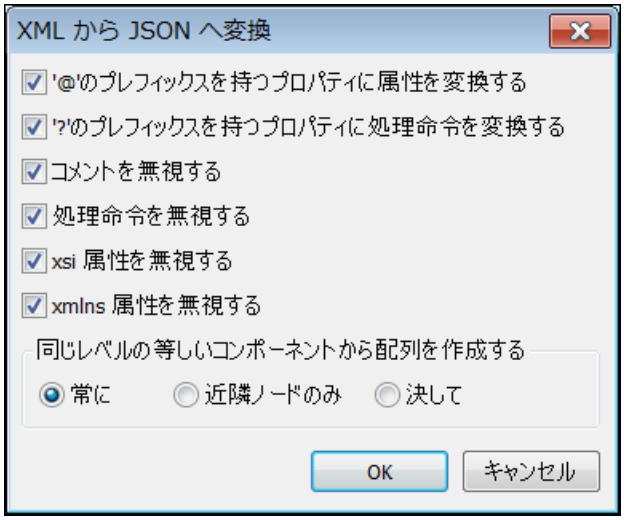
JSON document

```
{
        "XML":{
                "version":1.0,
                "encoding":"UTF-8"
        },
        "Person":{
                "first":"Jim",
                "last":"James",
                "Address":{
                        "street":"4 New Street",
                        "city":"New York",
                        "state":"NY",
                        "code":10123
                },
                "Tel":[{"type":"home",
                        "Text":"\r    123 123-1234\r "},{"type":"office",
                        "Text":"\r    123 987-9876\r "}]
        }
}
```

To convert a JSON document to XML, make the JSON document active and click the **Convert XML Instance to/from JSON** command.

XML to JSON conversion options

When you click the **Convert XML Instance to/from JSON** command to convert an XML instance document to a JSON instance document, the Convert XML to JSON dialog (*screenshot below*) appears. You can select whether you wish to convert to JSON or JSON5. Then set the conversion options you want, and click **OK**. A JSON instance document will be generated from the XML instance, and the generated JSON document will be opened in a new window.



The first two options define whether prefixes should be added to JSON property names so that conflicts with elements at the same level are avoided. The two listings below explain this. The XML *attribute* `somenode` has been converted to the JSON property `@somenode`. In this way, a conflict with the JSON property `somenode` (created from the XML *element* `somenode`) is avoided.

XML instance

```
<root somenode="value">
   <somenode>content</somenode>
</root>
```

JSON instance

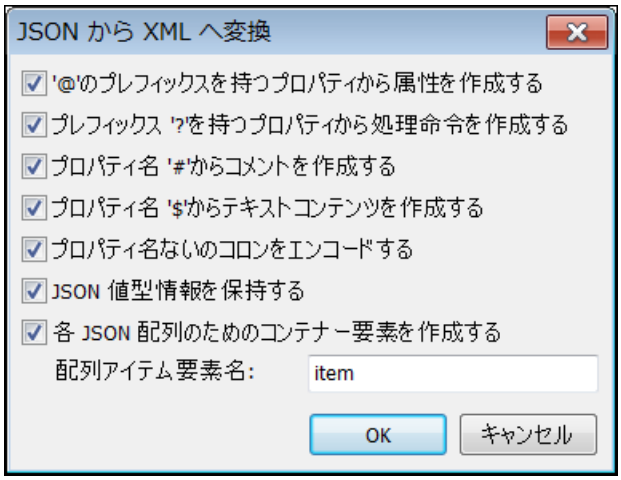
```
{
   "root":{
      "@somenode":"value",
      "somenode":"content"
   }
}
```

The next options enable you to specify whether certain types of XML nodes are to be converted or not. If XML comments are included they are given the name `"#"`. Text nodes (that typically occur in elements with mixed content) are given the name `"$"`. If an XML nodes has a namespace prefix, then the corresponding JSON name will be created with this namespace prefix. If elements with the same name exist at the same level, they are considered to be equal components. Similarly, nodes such as comments, processing instructions, and text() at the same level are also equal components. If equal components are present at the same level, you are able to choose whether to create an array or not. The options are whether to create the array out of all such equal components, only neighboring equal components, or not to create an array at all.

JSON to XML conversion options

When you click the **Convert XML Instance to/from JSON** command to convert a JSON instance document to an XML instance document, the Convert JSON to XML dialog (*screenshot*

below) appears. Set the conversion options you want, and click **OK**. An XML instance document will be generated from the JSON instance, and the generated XML document will be opened in a new window.



Note the following points:

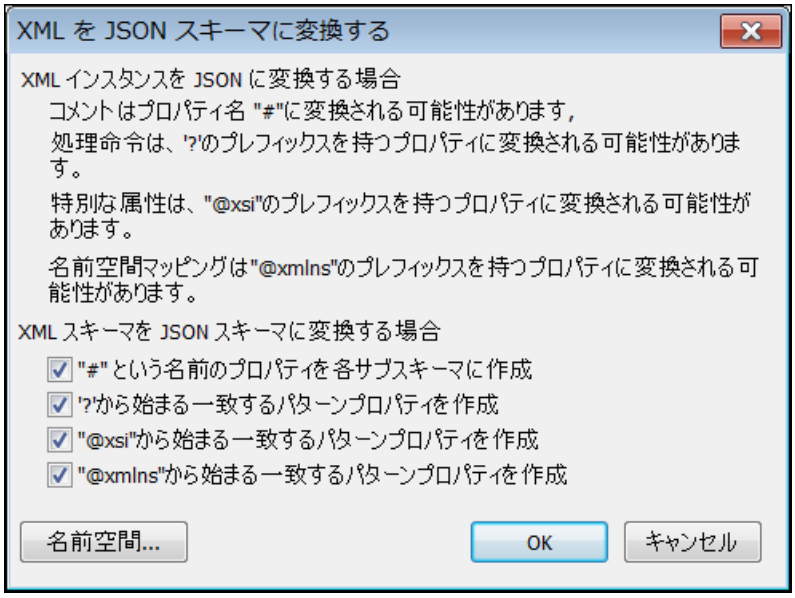
- JSON object properties are converted to XML elements. The first options in the dialog enable you to chose whether some types of properties are created or not.
- *Encode colons in property names:* If selected, colons in JSON names are encoded and not created as colons. If not selected, colons are left as is.
- *Keep JSON value type information:* If selected, a property's JSON type information is created as an attribute-value pair of the corresponding element.
- *Create container element for every JSON array:* The container element in the XML document will be given the name of the JSON array object. The items of the JSON array are created as XML elements within this container. Each is given the name you specify in the *Array item element name* text box.

26.10.10 Convert XML Schema to/ from JSON Schema

If the active document is an XML Schema, this command generates a JSON schema document from the XML schema. If the active document is a JSON schema, the command generates an XML Schema from the JSON schema. The generated document is opened in a new window, and can then be saved to any location. Conversion options are described below. For more information about JSON and JSON editing support in XMLSpy, see the section JSON and JSON Schema.

XML Schema to JSON Schema conversion options

When you click the **Convert XML Schema to/from JSON Schema** command to convert an XML Schema document to a JSON schema, the Convert XML Schema to JSON Schema dialog (*screenshot below*) appears. Set the conversion options you want, and click **OK**. A JSON schema will be generated from the XML Schema, and the generated document will be opened in a new window.

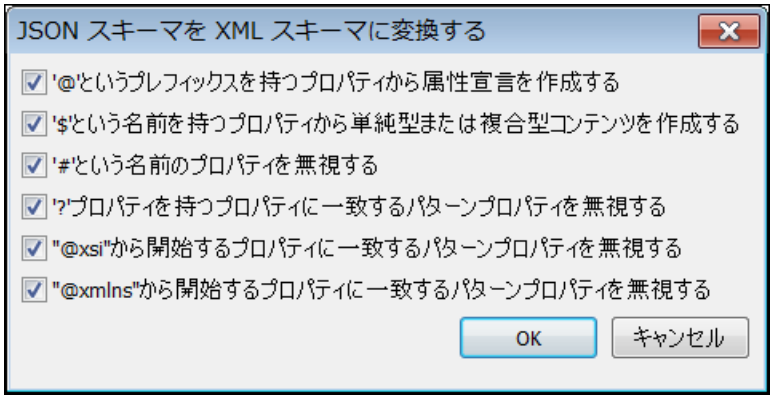


The top part of the dialog provides information about how certain XML Schema components are converted. The bottom part of the dialog provides the following options:

- *Create a property named "#" in each subschema:* If selected, a property with this name is created in each JSON schema definition.
- *Create pattern properties matching properties prefixed with '?', "@xsi", "@xmlns":* Specifies, for each of these prefixes, a pattern property to match properties with names that have these prefixes. For more information about pattern properties, see the section JSON Objects and Properties.
- *Always create arrays for particles with maxOccurs > 1:* In XML Schema, particles are the elements of complex content models. If the number of occurrences is more than one, then the particles are defined as an array in JSON Schema. Otherwise, they are defined as properties of a JSON object.

JSON Schema to XML Schema conversion options

When you click the **Convert XML Schema to/from JSON Schema** command to convert a JSON Schema document to an XML schema, the Convert JSON Schema to XML Schema dialog (*screenshot below*) appears. Set the conversion options you want, and click **OK**. An XML schema will be generated from the JSON Schema, and the generated document will be opened in a new window.

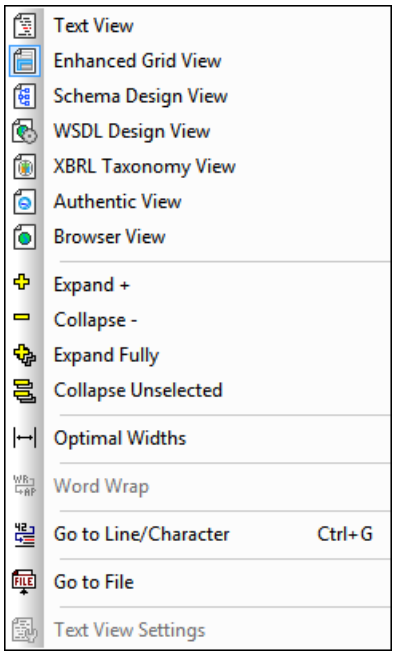


You can select the following options:

- Whether JSON property names that begin with `'@'` and `'$'` are created or not. They would be created, respectively, as attribute nodes and text nodes.
- Whether properties named `'#'` are created, as XML comment nodes, or not.
- Whether pattern properties that match properties prefixed with `'?'`, `"@xsi"`, and/or `"@xmlns"` are ignored or not. If not ignored, then the properties prefixed with `'?'`, `"@xsi"`, and `"@xmlns"` are converted, respectively, to processing instructions, `xsi:` prefixed attributes, and `xmlns:` prefixed attributes.

26.11 View Menu

The **View** menu (*screenshot below*) controls the display of the active Main window and allows you to change the way the document is displayed.



This section provides a description of commands in the **View** menu.

26.11.1 Text View

This command switches the current view of the document to Text View, which enables you to edit the document in its text form. It supports a number of advanced text editing features, described in detail in Text View section of this document.

Note: You can configure aspects of Text View in various tabs of the Options dialog (**Tools | Options**).

26.11.2 Enhanced Grid View

This command switches the current document into Grid View. If the previous view was Text View, the document is automatically checked for well-formedness.

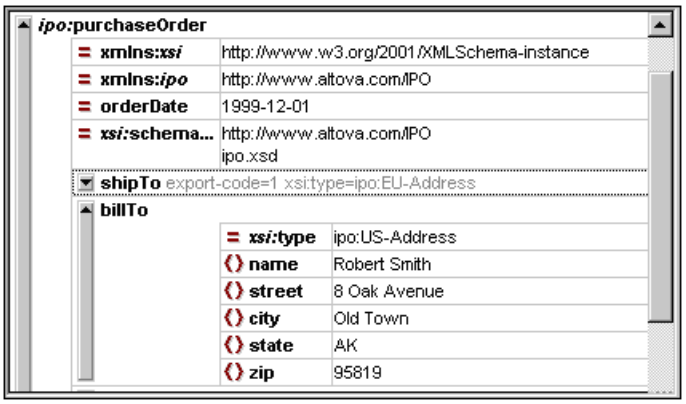


Table View
XMLSpy allows you to display recurring elements in a table structure in Grid View. This function is available wherever the Grid View can be activated, and can be used when editing any type of XML file: XML, XSD, XSLT, etc. For more information, see the Grid View section of this documentation.

26.11.3 Schema Design View

This command switches the current document, if it is an XML Schema document, to Schema Design View. For a detailed description of mechanisms available in this view, see the Schema View section of this documentation.

26.11.4 WSDL Design View

This command switches the current document, if it is a WSDL document (having a `.wsdl` file extension) to WSDL Design View. This view is described in detail in the WSDL View section of this documentation.

26.11.5 XBRL Taxonomy View

This command switches the current document to XBRL Taxonomy View if the document is an XBRL taxonomy document (having a `.xsd` file extension). Note that XBRL instance documents, which are XML files and have `.xml` suffixes, must be edited as normal XML files in other editing views and cannot be edited in XBRL Taxonomy View. For more information, see the XBRL View section of this documentation.

26.11.6 Authentic View

This command switches the current document to Authentic View.

Authentic View enables you to edit XML documents based on StyleVision Power Stylesheet templates created in Altova's StyleVision application. These templates (StyleVision stylesheets or SPS files) display XML documents in a graphical format that makes editing the XML document easier (than editing it in a text format with markup).

If an XML document is associated with an SPS file (**Authentic | Assign a StyleVision Stylesheet**), the XML document can be viewed in Authentic View. You can also open an SPS file as a new empty template in Authentic View, in one of two ways:

- Select the **File | New** command and then click the **Select a StyleVision stylesheet** button.
- Select the **Authentic | New Document** command and then browse for the SPS file.

See the Authentic View and StyleVision documentation for more information.

26.11.7 Browser View

This command switches the current document to Browser View. An XML-enabled browser renders the XML document using information from available CSS and/or XSL stylesheets.

When switching to Browser View, the document is first checked for validity if the *Validate upon saving* option in the File tab of the Options dialog (**Tools | Options**) is checked. For more information, see the Browser View section of this documentation.

26.11.8 Expand

This command (*shortcut:* numeric pad **'+'**) is enabled in Grid View and expands the selected element one level. The element remains selected after expansion, so you can expand the element additional levels by repeatedly clicking the shortcut **'+'** key.

26.11.9 Collapse

This command (*shortcut:* numeric pad **'-'**) is enabled in Grid View and collapses the selected element one level. You can expand or collapse any element by clicking the gray bar to the left of the element.

26.11.10 Expand Fully

This command (*shortcut:* * or x on the numeric keypad) is enabled in Grid View and in Text View if the Text View folding margin is active. It expands all descendant nodes of the selected element.

26.11.11 Collapse Unselected

This command (shortcut: **Ctrl** + numeric pad **'-'**) is enabled in Grid View and keeps the selected item uncollapsed while collapsing all others items. This helps maximize focus on one element and its children while reducing the focus on other nodes.

26.11.12 Optimal Widths

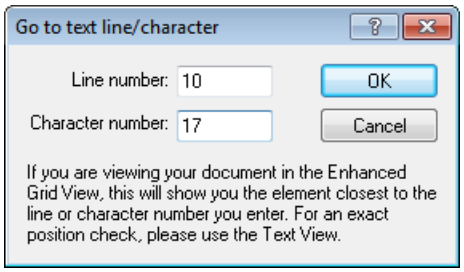
This command is enabled in Grid View and adjusts the widths of all columns in Grid View so that each column has a width that exactly accommodates in one line the longest text string in any of its cells. A maximum optimal width can be specified in the View tab of the Options dialog (**Tools | Options**). Note that optimal widths are calculated on the basis of the visible cells of columns. This enables the optimization of the view when individual elements are collapsed or expanded.

26.11.13 Word Wrap

This command enables or disables word wrapping in Text View. When word-wrapping is toggled on, text will wrap at the window's edge.

26.11.14 Go to Line/ Character

This command (*shortcut:* **Ctrl+g**) is enabled in Text View and Grid View. It pops up a dialog (*screenshot below*) in which you can enter the line number and character number to go to. In Text View, the cursor will jump to the position you entered. In Grid View, the node closest to the line and/or character number you entered will be highlighted.



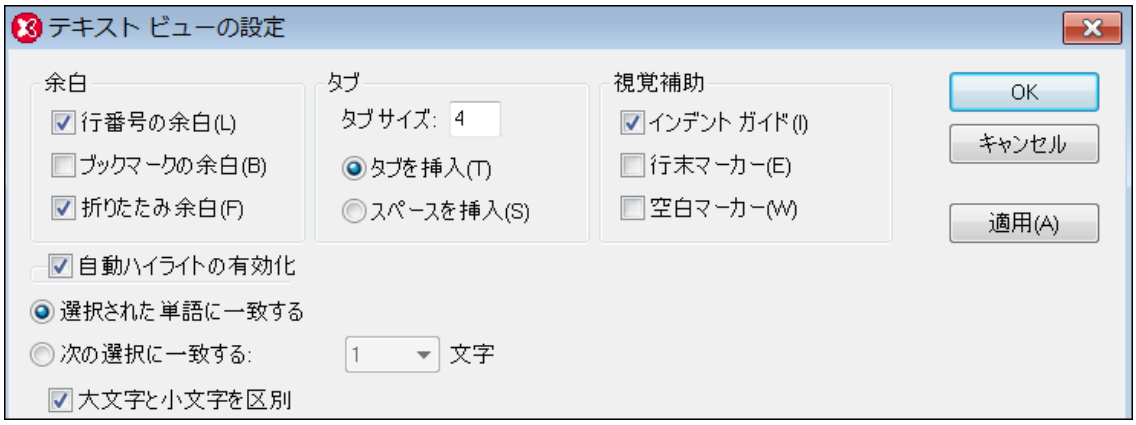
This feature is useful when you need to quickly navigate to a location, for example, when the location of an error is given in an error message.

26.11.15 Go to File

This command is enabled in Text View and Grid View. When the cursor is placed inside text that references a file (in Text View) or in a node (in Grid View) that contains text referencing a file, the referenced document is opened.It opens a document that is being referred to, from within the file you are currently editing.

26.11.16 Text View Settings

The **Text View Settings** command is enabled in Text View. It opens the Text View Settings dialog (*screenshot below*), in which you can configure Text View. A shortcut icon to open the dialog is available in the Text toolbar.



Margins

In the Margins pane, the Line Number, Bookmark, and Source Folding margins can be toggled on and off. Each of these is a separate margin in Text View and displays, respectively: (i) line numbers, (ii) bookmarks, and (ii) source folding icons to expand/collapse nodes. The settings of the Margins pane determine whether the margins are displayed in Text View or not. Bookmark commands are in the **Edit** menu. You can expand and collapse nodes in Text View only if the *Folding margin* setting is toggled on.

Tabs

The Tab pane enables you to set the tab size in terms of spaces. The radio buttons below the *Tab size* setting determine whether documents are displayed with tab or space indentation when pretty-printing-with-indentation is enabled in the View tab of the Options dialog (**Tools | Options**).

Visual Aid

The Visual Aid pane contains settings to toggle on indentation guides (dotted vertical lines that show the indentation of the text), end-of-line markers, and whitespace markers.

Enable auto-highlighting

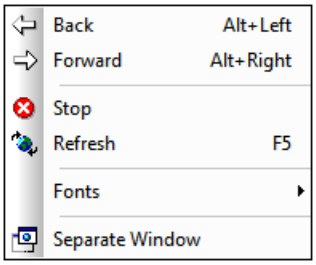
If highlighting is enabled, then all occurrences of a selection in Text View are highlighted. What constitutes a selection can be set via the options in this pane. A selection can be defined to be an entire word or a fixed number of characters, with the text-casing counting or not counting for a match. For a character selection, you can specify the minimum number of characters to match (for example, two or more characters). In Text View, all occurrences of character sequences that match your selection will be highlighted. For word searches, element names, attribute names, attribute values without quotes, and the angular brackets of element tags are considered to be separate words.

Key map

The key map is a list of XMLSpy shortcuts and their associated commands.

26.12 Browser Menu

The commands in the **Browser** menu are enabled in Browser View only. The **Back** and **Forward** commands, however, is enabled in Schema View also, where it takes you to the previously used command.



26.12.1 Back

The **Back** command (*shortcut:* **Alt + Left arrow**) is enabled in Browser View and Schema View.

In Browser View, the **Back** command displays the previously viewed page. The **Backspace** key achieves the same effect. The command is useful if you click a link in your XML document and then want to return to your XML document.

In Schema View, the **Back** command takes you to the previously viewed component or view. It can take you back to up to 500 previously viewed positions.

26.12.2 Forward

The **Forward** command (*shortcut:* **Alt + Right arrow**) is enabled in Browser View. In Schema View it is enabled only after you have used the **Back** command. The **Forward** command moves you forward through (i) previously viewed pages in Browser View, and (ii) previous views of schema components in Schema View.

26.12.3 Stop

The **Stop** command in enabled in Browser View and instructs the browser to stop loading your document. This is useful if large external files or graphics are being downloaded over a slow Internet connection, and you wish to stop the process.

26.12.4 Refresh

The **Refresh (F5)** command is enabled in Browser View and updates Browser View by reloading

the current document and documents related to the current document (such as CSS and XSL stylesheets, and DTDs).

26.12.5 Fonts

The **Fonts** command rolls out a sub-menu from which you can select the default font size for rendering the text of your XML document. The selection is available in Browser View only.

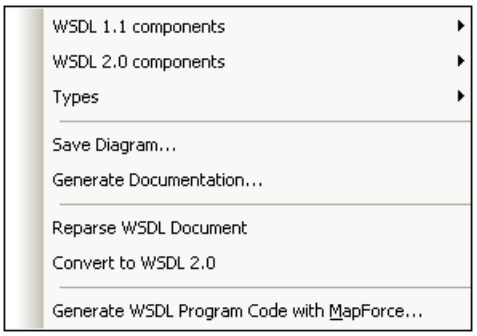
26.12.6 Separate Window

The **Separate Window** command is enabled in Browser View and undocks the Browser View of the document from the other views. As a separate window, Browser View can be displayed side-by-side with an editing view of the document.

To refresh the separated Browser View after making a change in an editing view, press **F5** in the editing view. To dock a separate Browser View window back into the window containing the other views, make the Browser View window active and click the **Separate Window** command.

26.13 WSDL Menu

The commands in the **WSDL** menu are available when viewing a WSDL document in WSDL View, which is graphical editor for creating and editing WSDL documents..



For a description of WSDL View, see the section WSDL View. To get started with WSDL, see the WSDL Tutorial.

See also: More information about working with WSDL documents is available in the sections, WSDL View and WSDL Tutorial.

26.13.1 WSDL 1.1 Components

Mousing over the **WSDL 1.1 Components** menu item scrolls out a submenu (*screenshot below*) from which various commands for editing WSDL 1.1 components can be selected.

Each item of the WSDL 1.1 menu (*screenshot above*) rolls out its own submenu, from which commands relating to that component can be selected. The commands in each of these submenus are described in the subsections of this section.

Messages

Insert message
Adds a new message to the WSDL document. The Messages item in the Overview entry helper opens, and the newly created message is highlighted there.

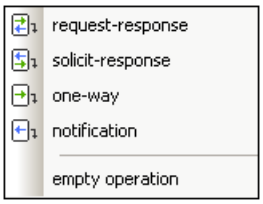
Delete message
Deletes the selected message from the input or output element.

Add message part (parameter)
Adds a message part (parameter) to the selected message.

Delete message part (parameter)
Deletes a message part (parameter) from the selected message.

Operations

Append Operation
Appends a new operation to the selected PortType. The type of operation to be appended can be selected from the submenu (*screenshot below*) of the **Append Operation** menu command.



Delete Operation
Deletes the selected PortType operation.

Add Input Element
Adds an input element to the selected PortType operation.

Add Output Element
Adds an output element to the selected PortType operation.

Add Fault Element
Adds a Fault element to the selected PortType operation.

Delete Input/Output/Fault Element
Deletes the selected PortType input, output or fault element.

Add New Message to Input/Output/Fault Element
Adds a new (default) message, to the currently selected PortType input, output, or fault element.

PortType

Insert PortType
Adds a new PortType to the PortTypes column of the Main Window.

Delete PortType
Deletes the selected PortType from the PortTypes column of the Main Window.

Binding

Insert Binding
Adds a new binding to the Bindings column of the Main Window.

Delete Binding
Deletes the selected binding from the Bindings column of the Main Window.

Append Child
Enables the addition of a new extensibility element to an input or output message. If the menu item is unavailable, it is not allowed in this position. See the W3C WSDL Specs for more information on Extensibilty items.

The following extensibility items are available:

- soap:body
- soap:header
- soap:headerfault
- soap:fault
- mime:content
- mime:multipartrelated
- mime:part
- mime:mimeXml
- http:urlencoded
- http:urlreplacement

Delete Extensibility
Deletes the selected extensibility item.

Service

Insert Service
Adds a new service in the Services column of the Main Window.

Delete Service
Deletes the selected service in the Services column of the Main Window.

Insert Port
Adds a new port to the selected service in the Services column of the Main Window.

Delete Port
Deletes the selected port from the currently selected service.

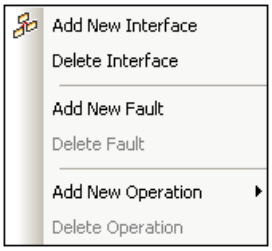
26.13.2 WSDL 2.0 Components

Mousing over the **WSDL 2.0 Components** menu item scrolls out a submenu (*screenshot below*) from which various commands for editing WSDL 2.0 components can be selected.

Each item of the WSDL 2.0 Components menu (*screenshot above*) rolls out its own submenu, from which commands relating to that component can be selected. The commands in each of these submenus are described in the subsections of this section.

Interface

The following commands are available in the **Interface** menu (*screenshot below*).



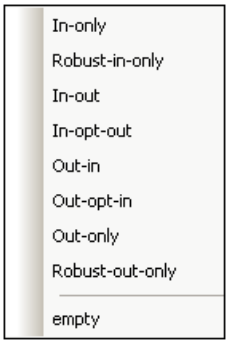
Add new interface
Adds a new interface box to the Interfaces column of the Main Window. The default name of the interface is highlighted in the interface box, enabling you to enter a new name directly.

Delete interface
Deletes the selected interface.

Add new fault
Adds a new `fault` element to the selected interface. The default name of the `fault` is highlighted in the interface box, enabling you to enter a new name directly.

Delete fault
Deletes the selected fault.

Add new operation
Adds a new `operation` element to the selected interface. The type of operation to be added is selected from the pop-out menu (*screenshot below*), and may be one of the operation types shown in the screenshot.

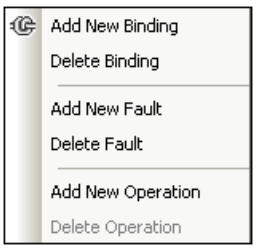


The default name of the `operation` is highlighted in the interface box, enabling you to enter a new name directly.

Delete operation
Deletes the selected operation.

Binding

The following commands are available in the **Binding** menu (*screenshot below*).



Add new binding

Adds a new binding box to the Bindings column of the Main Window. The default name of the binding is highlighted in the binding box, enabling you to enter a new name directly.

Delete binding

Deletes the selected binding.

Add new fault

Adds a new `fault` element to the selected binding. A `fault` element in a binding contains a `ref` attribute that references a fault declared in an interface. In the newly created fault in the binding, the interface fault that is to be referenced can be selected from the combo box of the newly created fault.

Delete fault

Deletes the selected fault.

Add new operation

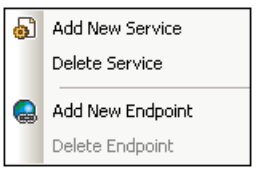
Adds a new `operation` element to the selected binding. An `operation` element in a binding contains a `ref` attribute that references an operation declared in an interface. In the newly created operation in the binding, the interface operation that is to be referenced can be selected from the combo box of the newly created binding operation.

Delete operation

Deletes the selected operation.

Service

The following commands are available in the **Service** menu (*screenshot below*).



Add new service
Adds a new service box to the Services column of the Main Window. The default name of the service is highlighted in the service box, enabling you to enter a new name directly. The interface reference can be selected from the combo box for the *Interface* property.

Delete service
Deletes the selected service.

Add new endpoint
Adds a new `endpoint` element to the selected service. The default name of the endpoint is highlighted in the service box, enabling you to enter a new name directly. The binding reference can be selected from the combo box for the *Binding* property. The address of the endpoint must be entered in the field for the *Address* property

Delete endpoint
Deletes the selected endpoint.

26.13.3 Types, Save Diagram
The **Types** menu item a has a sub-menu containing the following commands. These are described below.

- **New Schema**
- **Embed Schema**
- **Extract Schema(s)**
- **Edit Schema(s) in Schema View**

The **Save Diagram** command saves the design diagram as a PNG file.

Types | New Schema
This option only becomes active if the WSDL file does not contain a schema element. Please note that when using the **File | New** menu option, a schema element is included in the skeleton WSDL file. The menu item cannot be selected in this case (see below).

```
<types>
            <xs:schema/>
</types>
```

Types | Embed Schema
The command pops up an Open-File dialog, in which you can browse for the schema file you wish to embed. On clicking **OK** in the dialog, the schema is created as an inline schema within the `types` element. If the selected schema has already been imported, you will be prompted about whether you wish to embed the already imported schema. If you choose to embed the imported schema,it will be converted to an inline schema within the `types` element.

Types | Extract Schema(s)
On selecting this command, each of the embedded schemas (defined inline within the `types`

element) is opened as a temporary file in Schema View and a Save As dialog pops up for each file. If you choose to save a schema file, the schema will be extracted from the WSDL file, saved to the location you specify, and then imported into the WSDL file. It will no longer be an embedded schema, but an external, imported schema.

Types | Edit Schema(s) in Schema View
Opens a skeleton schema file if the WSDL file does not contain a reference to a specific schema. This is the case if you have used the **File | New** menu option. If a reference to a specific schema exists, then the schema opens in the embedded Schema View of the graphical WSDL editor.

26.13.4 Generate Documentation

The **WSDL | Generate Documentation** command generates detailed documentation of the current WSDL file. You can output the documentation as an HTML, MS Word, RTF, or PDF file. The documentation generated by this command can be freely altered and used; permission from Altova to do so is not required. Documentation is generated for components you select in the WSDL Documentation dialog (which appears when you select the Generate Documentation command). Related components are hyperlinked in the onscreen output, enabling you to navigate from component to component. Note that WSDL documentation can also be generated for **imported WSDL and XML Schema files**. The various documentation-generation options are described in the section, Documentation Options.

Note: In order to generate documentation in MS Word format, you must have MS Word (version 2000 or later) installed.

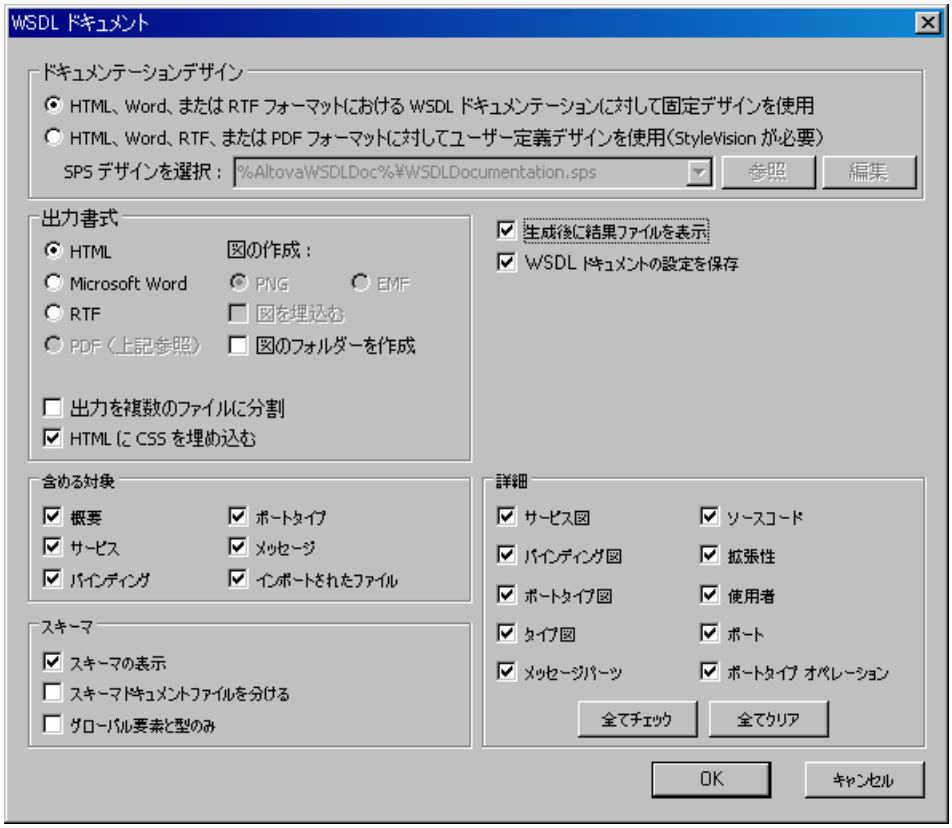
You can either use XMLSpy's fixed design for the generated document, or you can use a StyleVision SPS for the design. Using a StyleVision SPS enables you to customize the design of the generated documentation. How to do this is explained in the section, User-Defined Design.

Note: In order to use an SPS to generate WSDL documentation, you must have StyleVision installed on your machine.

Documentation Options

The **WSDL | Generate Documentation** command pops up the WSDL Documentation dialog (*screenshot below*), in which you can select options for the documentation.

In the Documentation Design pane of the dialog you can select whether to use the fixed XMLSpy design for the generated documentation or whether to use a customized design created in a StyleVision SPS. Select the option you want. Note that PDF output is available only for documentation generated with a StyleVision SPS, not for documentation generated using a fixed design. How to work with a user-defined design is described in the section, User-Defined Design.



The other options in the WSDL Documentation dialog are explained below. Depending on whether a WSDL 1.1 document or a WSDL 2.0 document is active, the items in the Include and Details pane of the dialog will be different. The screenshot above shows the WSDL Documentation dialog for a WSDL 1.1 document.

- The required format is specified in the Output Format pane: either HTML, Microsoft Word, RTF, or PDF. (The PDF output format is only available if you use a StyleVision SPS to generate the documentation.) On clicking **OK**, you will be prompted for the name of the output file and the location to which it should be saved.
- Microsoft Word documents are created with the `.doc` file extension when generated using a fixed design, and with a `.docx` file extension when generated using a StyleVision SPS.
- The documentation can be generated either as a single file or be split into multiple files. When multiple files are generated, each file corresponds to a component. What components are included in the output is specified using the check boxes in the Include pane. In fixed designs, links between multiple documents are created automatically.
- For HTML output, the CSS style definitions can be either saved in a separate CSS file or embedded in the HTML file (in the `<head>` element). If a separate CSS file is created, it will be given the same name as the HTML file, but will have a `.css` extension. Check or uncheck the *Embed CSS in HTML* check box to set the required option.
- The *Embed Diagrams* option is enabled for the MS Word, RTF, and PDF output options. When this option is checked, diagrams are embedded in the result file, either in PNG or EMF format. Otherwise diagrams are created as PNG or EMF files, which are displayed in the result file via object links.
- When the output is HTML, all diagrams are created as document-external PNG files. If the *Create folder for diagrams* check box is checked, then a folder will be created in the same folder as the HTML file, and the PNG files will be saved inside it. This folder will have a name of the format *`HTMLFilename`*`_diagrams`. If the *Create folder for diagrams* check box

is unchecked, the PNG files will be saved in the same folder as the HTML file.

- In the Include pane, you select which items you want to include in the documentation. The *Overview* option lists all components, organized by component type, at the top of the file. If the *Imported Files* (WSDL 1.1) or *Imported/Included Files* (WSDL 2.0) option is checked, then components in imported files (as well as included files in the case of WSDL 2.0) are included in the schema documentation.
- In the Schema pane, you can select whether schemas in the file are reported or not. If you choose to have schemas reported, you can further choose: (i) whether the schema documentation should be reported in a separate file or in the main documentation file, and (ii) whether the full schema should be reported or only global elements, simple types, and complex types.
- The Details pane lists the details that may be included for each component. Select the details you wish to include in the documentation. The **Check All** and **Uncheck All** buttons enable you to quickly select or deselect all the options in the pane.
- The *Show Result File* option is enabled for all output options. When this option is checked, the result files are displayed in Browser View (HTML output), MS Word (MS Word output), and the default applications for `.rtf` files (RTF output) and `.pdf` files (PDF output).

Parameter values

If the StyleVision SPS contains one or more parameter definitions, then on clicking **OK**, a dialog pops up listing all the parameters defined in the SPS. You can enter parameter values in this dialog to override the default parameter values that were assigned in the SPS.

User-Defined Design

Instead of the fixed standard XMLSpy design, you can create a customized design for the WSDL documentation. The customized design is created in a StyleVision SPS, which is a design template for the output document.

Creating the SPS

A StyleVision Power Stylesheet (or SPS) is created using Altova's StyleVision product. An SPS for generating WSDL documentation must be based on an XML Schema that specifies the structure of the WSDL documentation. Two schemas, one for WSDL 1.1 and the second for WSDL 2.0, are delivered with your XMLSpy package. They are, respectively, `WSDLDocumentation.xsd` and `WSDL20Documentation.xsd`, located respectively in the folders of the (My Documents) folder:

- `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Documentation\WSDL`.
- `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Documentation\WSDL20`.

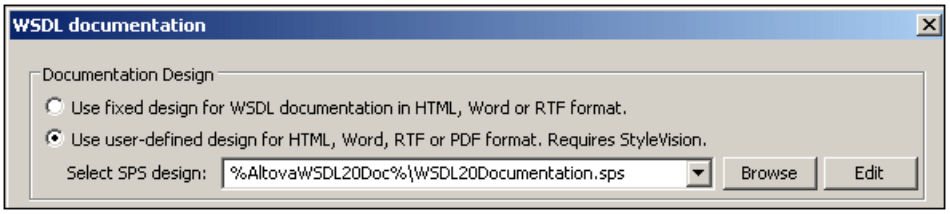
When creating the SPS design in StyleVision, nodes from the schema are placed in the design template and assigned styles and properties. Additional components, like links, tables and images, can also be added to the SPS design. In this way, the entire output document can be designed in the SPS. How to create an SPS design in StyleVision is described in detail in the StyleVision user manual.

The advantage of using an SPS for generating WSDL documentation is that you have complete control over the SPS design. Note also that PDF output of the WSDL documentation is available only if a user-defined SPS is used; PDF output is not available if the fixed XMLSpy design is used.

Specifying the SPS to use for WSDL documentation

After an SPS has been created, it can be used to generate WSDL documentation. The SPS you wish to use for generating the WSDL documentation is selected in the WSDL Documentation dialog (accessed via the **WSDL | Generate Documentation** command). In the Documentation Design pane of this dialog (*screenshot below*), select the *Use User-Defined Design* radio button. You can then click the **Browse** button and browse for the SPS you want. Click the dialog's **OK** button, and, in the Save dialog that pops up, select the folder for, and enter the name of, the output file.

Note: The SPS file must correctly locate the schema on which it is based: `WSDLDocumentation.xsd` or `WSDL20Documentation.xsd` (*see above*).



Two editable SPS designs, one each for WSDL 1.1 and WSDL 2.0, are delivered with XMLSpy. They are, respectively, in the `WSDL` and `WSDL20` sub-folders of the (My Documents) folder: `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2011\Documentation\`. They are named:

- `WSDL\WSDLDocumentation.sps`
- `WSDL20\WSDL20Documentation.sps`

These files, together with other SPS files you have recently browsed for, will be available in the combo box of the *Use User-Defined* option (*see screenshot above*).

Clicking the **Edit** button in the Documentation Design pane launches StyleVision and opens the selected SPS in a StyleVision window. In order to preview the result document in StyleVision, you will need a Working XML file. Sample XML files for this purpose, called `TimeService.xml` and `TimeService20.xml`, are supplied with your application and are located in the (My Documents) folder:

`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Documentation\WSDL(20)\SampleData`

Note: In order to use an SPS to generate WSDL documentation, you must have StyleVision installed on your machine.

26.13.5 Reparse WSDL Document

Reparses the document. This is required in some situations, for example, when a schema associated with the WSDL document has been modified. Reparsing the WSDL document will update it with information from the modified schema document.

26.13.6 Convert to WSDL 2.0

The **Convert to WSDL 2.0** command is enabled only when a WSDL 1.1 is active in WSDL View. It generates a WSDL 2.0 document from the active WSDL 1.1 document. Clicking this command pops up a File Save dialog, in which you can specify the location and name of the WSDL 2.0 file that will be generated by XMLSpy.

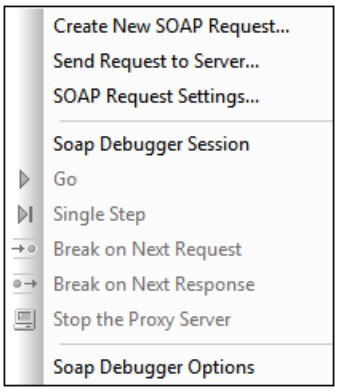
On clicking **OK** in the File Save dialog, a WSDL 2.0 document is generated and saved to the specified location, and opened in WSDL View in a new tab. The file can then be edited as required, just like any other WSDL 2.0 document.

26.13.7 Generate WSDL Program Code with MapForce

The **Generate WSDL Program Code with MapForce** command launches Altova's MapForce if the application is installed. MapForce enables you to map a schema to another DTD, XML Schema, or database, to generate XML, and to generate program code from the WSDL file.

26.14 SOAP Menu

XMLSpy supports SOAP versions 1.1 and 1.2, and WSDL versions 1.1 and 2.0.



The SOAP: How To section that follows the menu descriptions, shows you how to use the SOAP debugger using the **nanonull.com timeservice** server supplied by Altova. Please use this service to test the SOAP debugger. The AirportWeather web service, described on the following pages, might not always be available to you.

Use the SOAP functionality:

- To test your web services without having to implement client applications
- For quick testing of third party web services

For more info on these specifications please see:
SOAP http:/ / www.w3.org/ TR/ SOAP/
WSDL http:/ / www.w3.org/ TR/ wsdl

26.14.1 Create New SOAP Request

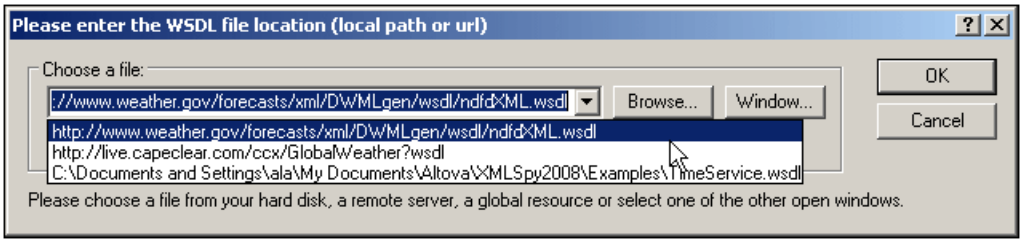
This command creates a new SOAP request document. It involves the following steps:

1. Enter the WSDL file location and connect to the SOAP server.
2. The server responds with a list of operations. Select the SOAP operation you want.
3. The server responds with a SOAP Request form in XML format. Define the SOAP Request form.

We demonstrate the process below by creating a SOAP request for the US National Digital Forecast Database (NDFD) SOAP Service (http://www.nws.noaa.gov/xml/).

Connecting to the SOAP server

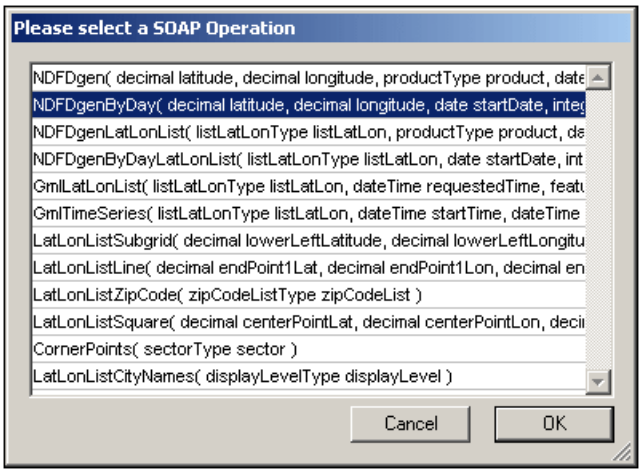
The connection is made via a WSDL file. In our example, the URI of the WSDL file is: `http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl`. To make the connection, click the **Create New SOAP Request** command, and enter the file URI in the dialog that appears (*screenshot below*).



Click **OK** to confirm the selection.

Select the required SOAP operation

The server responds with a list of operation which are displayed in a dialog (*screenshot below*).



Select an operation and click **OK**. We selected the `NDFDgenByDay` operation.

Define the SOAP Request

The server responds by sending an XML file, which is displayed in the Text View of XMLSpy. For the operation we selected, we received the following XML file.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC="http://
schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns:m0="http://www.weather.gov/forecasts/xml/DWMLgen/schema/DWML.xsd">
        <SOAP-ENV:Body>
                <m:NDFDgenByDay xmlns:m="http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl"
                SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
                        <latitude xsi:type="xsd:decimal">0.0</latitude>
                        <longitude xsi:type="xsd:decimal">0.0</longitude>
                        <startDate xsi:type="xsd:date">1967-08-13</startDate>
                        <numDays xsi:type="xsd:integer">0</numDays>
                        <format xsi:type="m0:formatType">String</format>
                </m:NDFDgenByDay>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

We filled in the parameters as required by the XML (*in bold below*):

```
<SOAP-ENV:Body>
        <m:NDFDgenByDay xmlns:m="http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
            <latitude xsi:type="xsd:decimal">45</latitude>
            <longitude xsi:type="xsd:decimal">-90</longitude>
            <startDate xsi:type="xsd:date">2009-10-13</startDate>
            <numDays xsi:type="xsd:integer">1</numDays>
            <format xsi:type="m0:formatType">24 hourly</format>
        </mNDFDgenByDay>
</SOAP-ENV:Body>
```

This completes the **definition** of this SOAP request. In the next step, we shall send the request.

26.14.2 Send Request to Server

This command sends a SOAP request to the SOAP server. The SOAP request is an XML document. It must be the active document in XMLSpy when the command is selected.How to define a SOAP request is described in the description of the command Create New SOAP Request.

After the SOAP request is sent, a response is received from the SOAP server. This response is an XML document, which is displayed in the Text View of XMLSpy. For example, show below is a screenshot of the XML document that was returned in response to the SOAP request we defined in the section Create New SOAP Request.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SO
   http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:
   http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/enc
3      <SOAP-ENV:Body>
4        <ns1:NDFDgenByDayResponse xmlns:ns1="http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/n
5          <dwmlByDayOut xsi:type="xsd:string"><?xml version="1.0"?>
6  <dwml version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/
   xsi:noNamespaceSchemaLocation="http://www.nws.noaa.gov/forecasts/xml/DWMLgen/schema/DWML.xsd">
7    <head>
23   <data>
24     <location>
25       <location-key>point1</location-key>
26       <point latitude="45.00" longitude="-90.00"/>
27     </location>
28     <moreWeatherInformation applicable-location="point1">
   http://forecast.weather.gov/MapClick.php?textField1=45.00&amp;amp;textField2=-90.00
29     </moreWeatherInformation>
30     <time-layout time-coordinate="local" summarization="24hourly">
35     <time-layout time-coordinate="local" summarization="12hourly">
42     <parameters applicable-location="point1">
43       <temperature type="maximum" units="Fahrenheit" time-layout="k-p24h-n1-1">
44         <name>Daily Maximum Temperature</name>
45         <value>45</value>
46       </temperature>
47       <probability-of-precipitation type="12 hour" units="percent" time-layout="k-p12h-n2-2">
48         <name>12 Hourly Probability of Precipitation</name>
49         <value>14</value>
50         <value xsi:nil="true"/>
51       </probability-of-precipitation>
52     </parameters>
53   </data>
54 </dwml>
55 </dwmlByDayOut>
56       </ns1:NDFDgenByDayResponse>
57     </SOAP-ENV:Body>
58 </SOAP-ENV:Envelope>
```

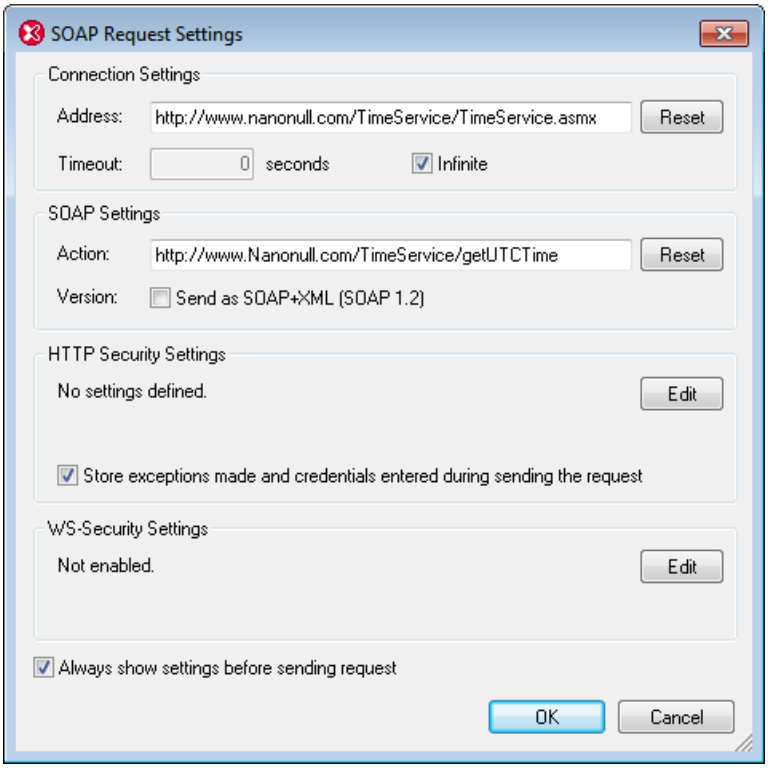
Saving and reusing a SOAP request
XMLSpy allows you to save a SOAP request and resend it at a later time. Do this as follows:

1. Save the SOAP request XML document (**File | Save as**).
2. Close the SOAP request file.
3. Reopen the SOAP request XML document, and select the menu option **SOAP | Send Request to Server**. (Any XML file can be used as a SOAP request document.)

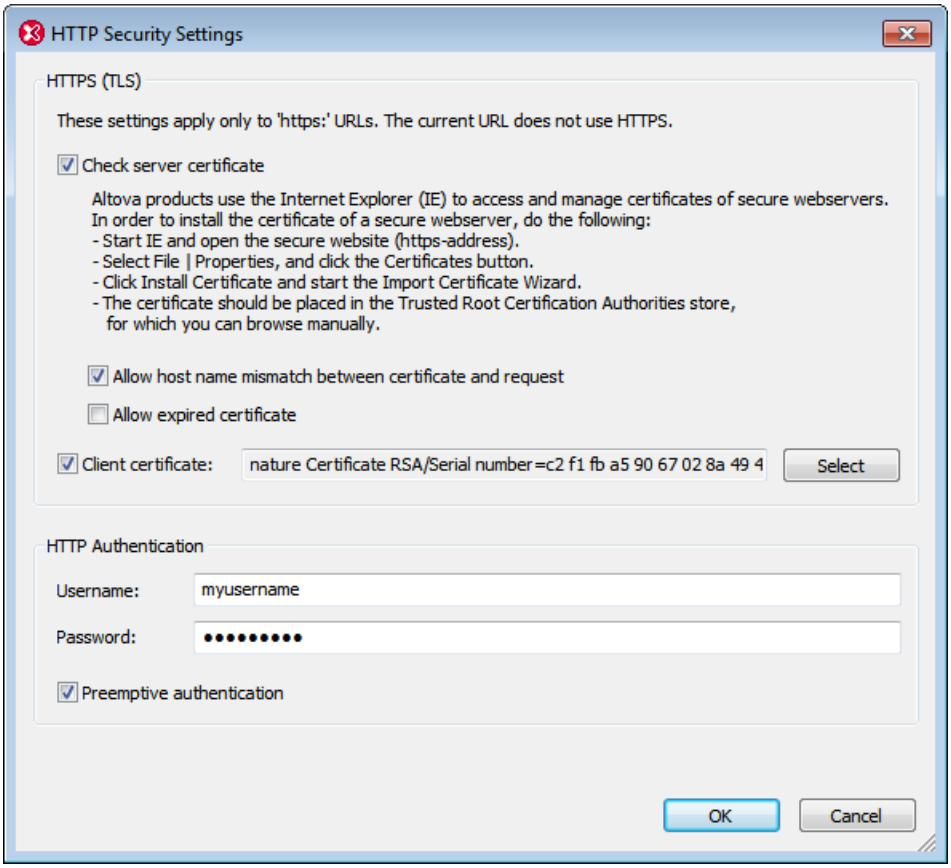
26.14.3 SOAP Request Settings

This command displays the SOAP Request Settings dialog (*screenshot below*), in which you can specify various settings of the SOAP request. These settings are described below.

1. Make the SOAP request document active.
2. Select the menu option **Soap | SOAP Request Settings**. This opens the SOAP Request Settings dialog box (*screenshot below*).

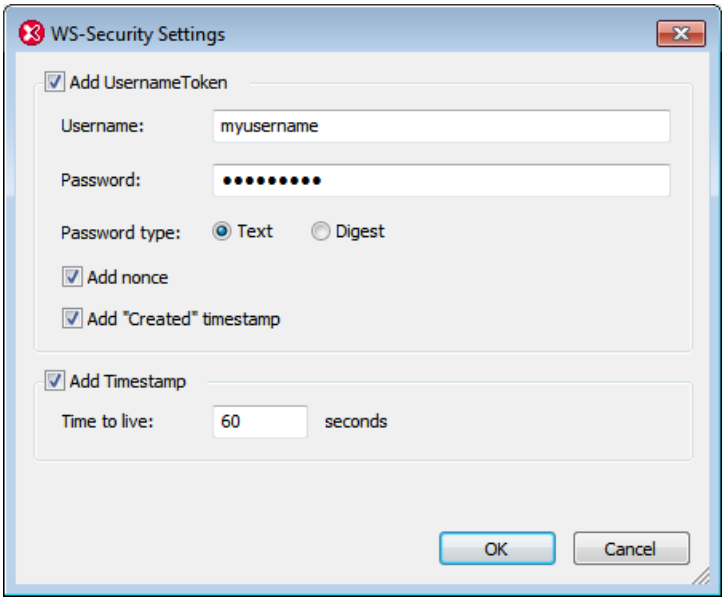


3. In the *Address* field, enter the desired connection endpoint. If the SOAP request was created from a WSDL file in XMLSpy, then the value of the *Address* field will be the location of the endpoint selected in the WSDL. Click **Reset** to obtain this endpoint. A connection timeout value can be specified in seconds. To set no timeout value, check the *Infinite* check box.
4. In the *Action* field, enter the SOAP action to perform. To send the request as SOAP 1.2, check the Send as *SOAP+XML (SOAP 1.2)* check box. If the SOAP request was created from a WSDL file in XMLSpy, then the SOAP action is received from the extensibility element under the corresponding SOAP binding operation in the WSDL file. In this case, the SOAP version is also pre-selected from the WSDL. (The SOAP version affects the value of the HTTP header Content-Type: `text/xml` or `application/soap+xml`.) Click **Reset** to obtain the SOAP action from the WSDL file.
5. The HTTP Security Settings pane provides a summary of the security settings lists. If the *Store exceptions while sending request* option is checked, then all the settings are saved when the request is sent, and can be re-used for the next request. Click the **Edit** button to display the HTTP Security Settings dialog (*screenshot below*). How to install server certificates is described below.



If you wish to allow a ***host name mismatch*** (between the host name in the server certificate and the actual address you use) or an ***expired server certificate***, then check these options in the dialog. If the server requires a ***client certificate***, you can specify the location of the client certificate. If authentication is required by the server, specify the ***user name*** and ***password*** for standard authentication. When the initial client request to the server contains the required authentication information, this process is referred to as ***preemptive authentication***. If required by the server, select the *Preemptive authentication* option. Otherwise, leave the *Preemptive authentication* option unselected.

6. In addition to security on the transport layer (HTTP security settings), you can also specify web service security settings if these are required by the web service. Click the **Edit** button of the WS Security Settings pane to display the WS Security Settings dialog (*screenshot below*). The security information includes the username, password, the type of the password, an automatically generated nonce code string, and a timestamp. You can also specify the validity period of the security information (*Add Timestamp*). The dialog creates an XML fragment that contains the security information and embeds this fragment in the SOAP request. *See listing below*.



7. When you are done, click **OK**.

About trusted certificates

Altova products use Internet Explorer (IE) to access and manage trusted certificates of secure webservers. Installing the certificate of a webserver in IE allows IE to access the webserver without issuing a warning or aborting the process. In order to install the certificate of a secure webserver, do the following:

- In Internet Explorer 8, open the secure website.
- Select **File | Properties**, and click the Certificates button.
- Click **Install Certificate** and start the Import Certificate Wizard. (This Wizard can also be accessed via **Tools | Internet Options| Content | Certificates | Import**.)
- The certificate should be placed in the Trusted Root Certification Authorities store, for which you can browse manually.
- Finish the Wizard steps, close the Certificates and Properties dialogs respectively by clicking **OK**. You might need to restart Internet Explorer.

Note: Only change the SOAP action settings if you can access all the SOAP methods and their corresponding SOAP actions.

Web service security information

Some web services require user authentication. (The web service security layer would be in addition to the HTTP security layer implemented by the server.) The web service authentication information is stored in the SOAP request as an XML fragment having a structure as listed below. This XML fragment is generated automatically in the SOAP request from the web service authentication information you enter in the WS Security Settings dialog.

```
<wsse:Security xmlns:wsse="..." xmlns:wsu="..." SOAP-ENV:mustUnderstand="true">
  <wsse:UsernameToken>
    <wsse:Username>usr</wsse:Username>
```

```
    <wsse:Password Type="...#PasswordText">pwd</wsse:Password>
    <wsse:Nonce EncodingType="... #Base64Binary">UqrtD963797WBRgWiJPu2w==</wsse:Nonce>
    <wsu:Created>2014-11-17T16:08:07.016Z</wsu:Created>
  </wsse:UsernameToken>
  <wsu:Timestamp>
    <wsu:Created>2014-11-17T16:08:07.016Z</wsu:Created>
    <wsu:Expires>2014-11-17T16:09:07.016Z</wsu:Expires>
  </wsu:Timestamp>
</wsse:Security>
```

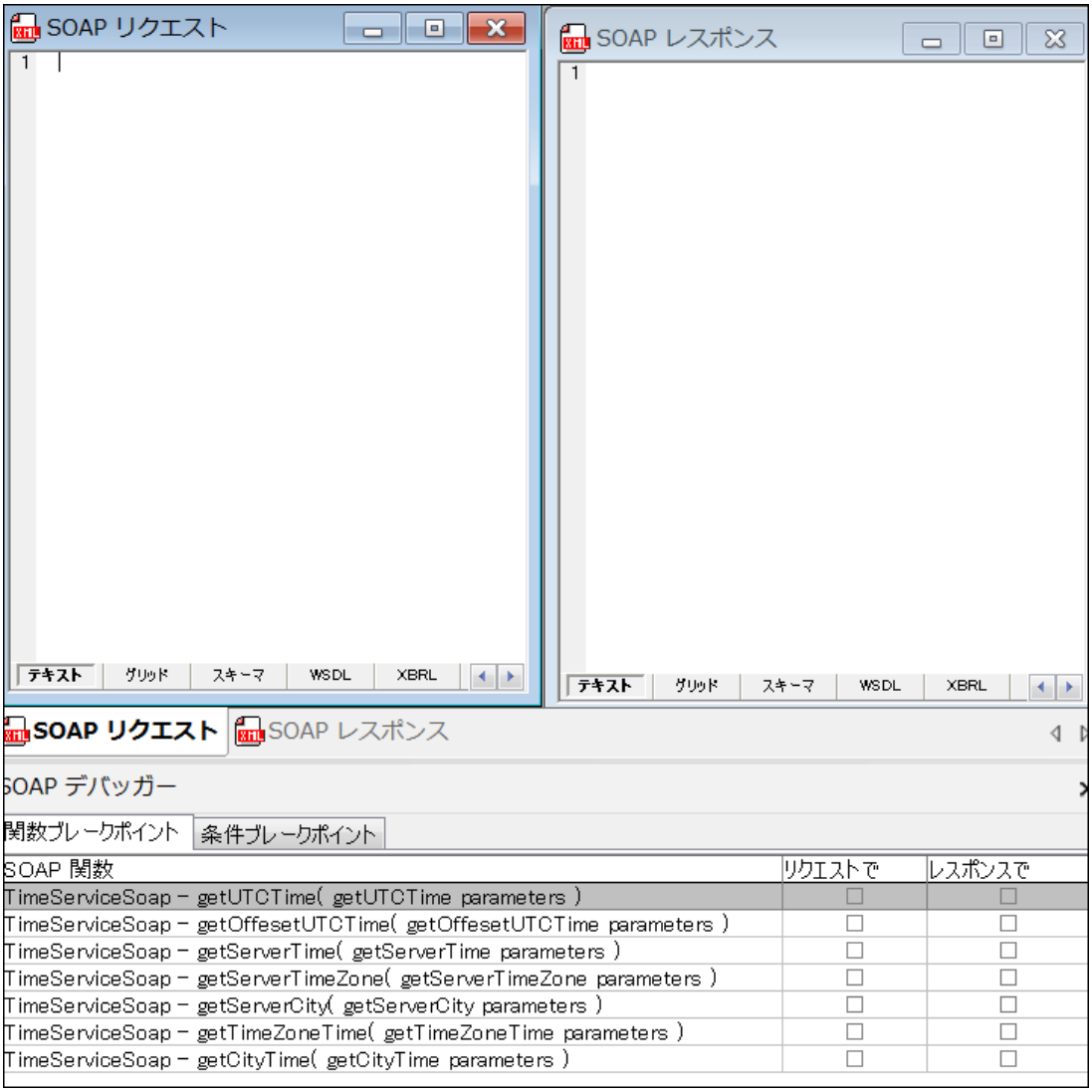
26.14.4 Soap Debugger Session

This command starts a SOAP debugger session.

- A dialog box is immediately opened after you select this command. You then have to select a WSDL file location, generally a URL. You can also select a file via a global resource (click the icon and select a global resource in the dialog that pops up) or a file in one of the open windows in XMLSpy.
- Select the source and target ports needed for the debugger proxy server and the web service, in the following dialog box.

  This opens the SOAP debuggers proxy server in its **inactive** state. Clicking one of the SOAP toolbar icons, starts the SOAP debugger and waits for the client requests.

  Please see the Soap - How to section for a more detailed description.



26.14.5 Go

This command activates the SOAP proxy server and processes the WSDL file until a breakpoint is encountered. The respective SOAP document then appears in one of the SOAP document windows.

26.14.6 Single Step

This command allows you to single-step through the incoming and outgoing SOAP requests and responses. The SOAP debugger stops for each request and response. The proxy server is also started if it was inactive.

26.14.7 Break on Next Request

This command causes the debugger to stop on the next SOAP request, and display the data in the SOAP Request document window. You can directly edit the data in this window before sending it on to the web service.

26.14.8 Break on Next Response

This command causes the debugger to stop on the next SOAP Response, and display the data in the SOAP Response document window. You can directly edit the data in this window before sending it on to the client.

26.14.9 Stop the Proxy Server

This command stops the debugger proxy server.

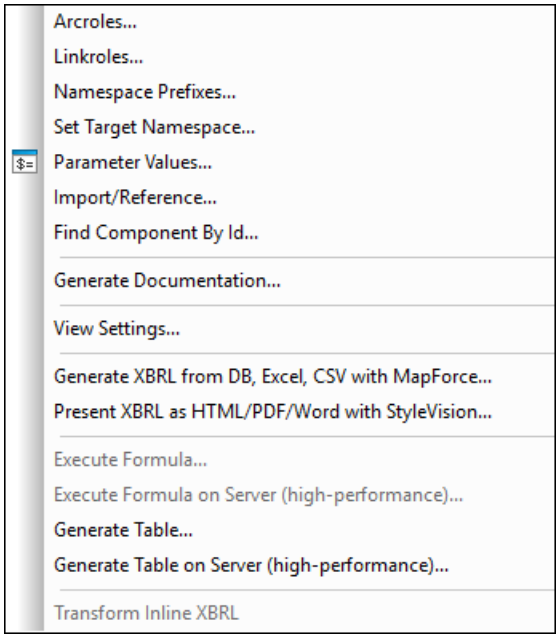
26.14.10 SOAP Debugger Options

The SOAP Debugger Options dialog (*screenshot below*) enables you to specify the computer's IP address, and other debugger options, which are listed below. Access the dialog with the **SOAP | SOAP Debugger Options** menu command.

- *Computer Address:* The address of the proxy server from which the debugger runs. The debugger on the proxy server takes requests from machines on the network and sends them to the web service. Since the debugger runs inside XMLSpy, the machine on which XMLSpy is installed also serves as the proxy server. The IP address of the machine is automatically detected and entered in this field. Only if the IP address cannot be detected automatically, do you need to enter the IP address (as an `http` address) in this field. To find out your computer's IP address, open a command prompt window, enter the command `ipconfig /all`, and press **Enter**.
- *Timeout:* This value is the amount of time the SOAP Debugger stays in a breakpoint. The default is 5 seconds.
- *Hide entry helpers; Hide project/info windows:* These options are useful for providing more screen space for the SOAP Debugger window.

26.15 XBRL Menu

The **XBRL** menu (*screenshot below*) contains commands that are enabled when a taxonomy is being edited in XBRL View. These commands are listed below and described in the sub-sections of this section.



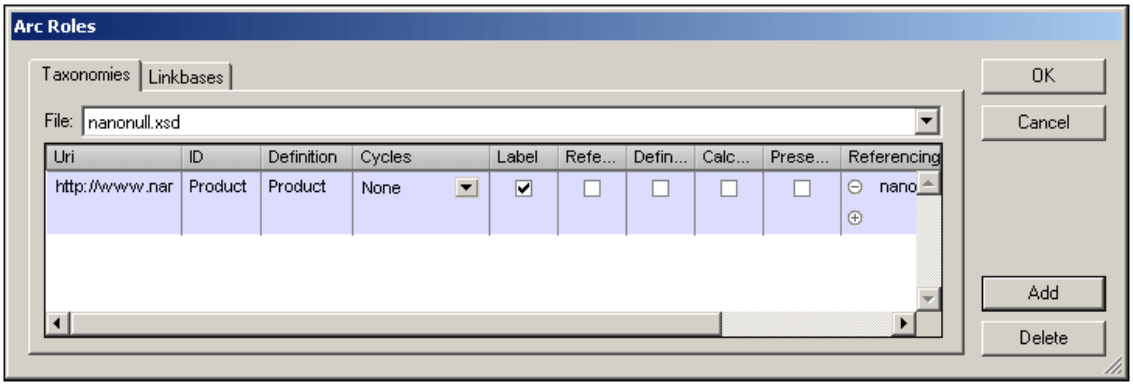
- Arcroles: defines arcroles
- Linkroles: defines linkroles
- Namespace Prefixes: manages taxonomy namespaces
- Set Target Namespace: defines and declares the target namespace of the taxonomy
- Import/Reference: imports an XBRL taxonomy or references a linkbase
- Find Formula Component by ID: finds formula components on the basis of the user-supplied ID
- Generate Documentation: generates documentation of the current XBRL taxonomy
- View Settings: sets default for XBRL View
- Generate XBRL from DB, Excel, CSV with MapForce: launches Altova MapForce for generating an XBRL instance file
- Present XBRL as HTML/PDF/Word with StyleVision: launches Altova StyleVision for designing an XBRL report
- Execute Formula: Executes formulas and/or assertions from the DTS associated with the active XBRL instance document
- Generate Table: Generates XBRL Tables from an XBRL instance
- Transform Inline XBRL: Generates the Inline XBRL part of an XHTML document as XBRL

For more information about XBRL, see the sections XBRL and Editing Views | XBRL View.

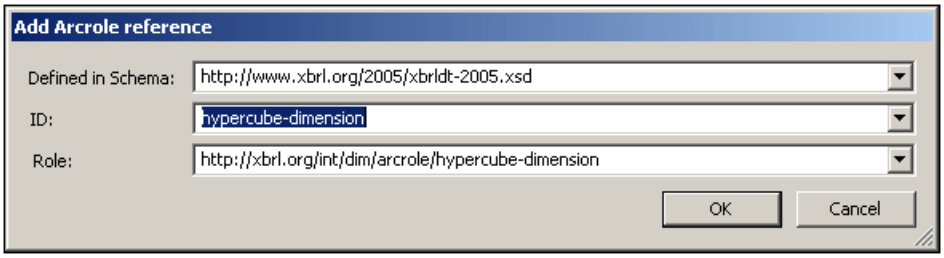
26.15.1 Arcroles

The **Arcroles** command pops up the Arc Roles dialog (*screenshot below*) in which arcroles can be created for a taxonomy. Arcroles are stored in the concept definitions file, within the `appinfo` element. They specify the role of an arc.

In the **Taxonomies tab** of the Arc Roles dialog (*screenshot below*), only taxonomies that are editable or that contain an arcrole or linkrole are listed in the combo box. You can add an arcrole to a taxonomy by clicking the **Add** button. Then define the arcrole's URI, ID, definition, and cycles. To specify in which kinds of relationships the arcrole should be available, check the boxes of the required relationship kinds. Linkbases that reference an arcrole can be added to the Referencing Linkbase Files column.



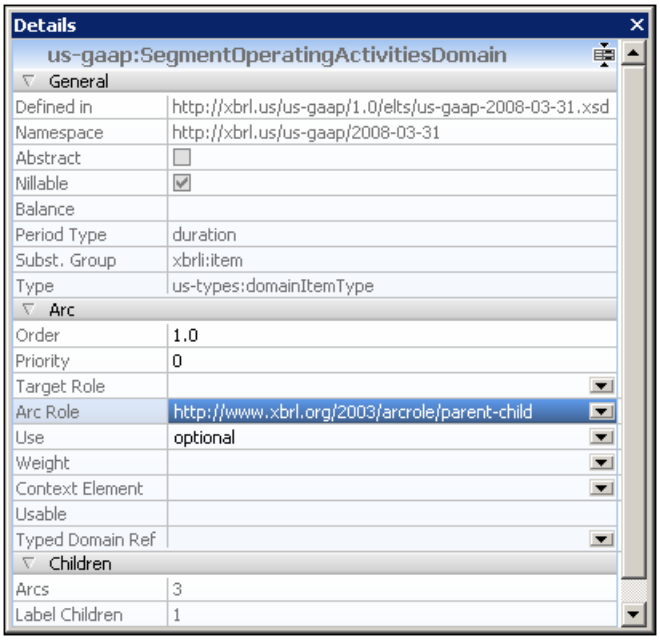
The **Linkbases tab** provides another view of the taxonomy's arcroles. In this view, you add and view arcroles according to individual linkbases (for example calculation or presentation linkbases). Select a linkbase in the combo box and then add or delete an arcrole as required. When you click the **Add** button, the Add Arcrole Reference dialog (*screenshot below*) pops up.



Each of the entries in this dialog is a combo box that enables you to select from available options. The *Defined in Schema* field enables you to select the taxonomy in which the arcrole is defined. The *ID* and *Role* combo boxes provide the available arcroles. After you have selected an arcrole and clicked **OK**, the reference is added to the linkbase. In the Taxonomies tab, the arcrole you referenced will show the referencing linkbase in the Referencing Linkbase Files column. If you wish to make this arcrole available for a particular kind of relationship, you will still, however, have to check the appropriate relationship kind check box.

After an arcrole has been created in the taxonomy it can be used to specify the role of an arc in a relationship kind for which the arc is available according to its definition. In the screenshot above, for example, the arcrole has been made available for arcs in label relationships.

The arcrole of an arc is selected in the Details entry helper (*screenshot below; arcrole highlighted*).



With the element at the `to` end of an arc selected, in the Details Entry Helper, select the required item from the dropdown list of the arcrole entry.

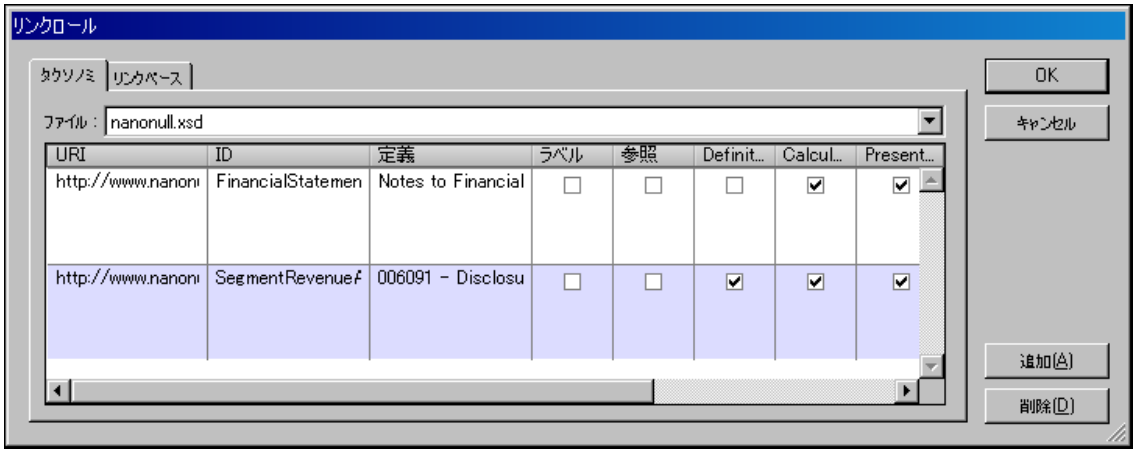
26.15.2 Linkroles

The **Linkroles** command pops up the Link Roles dialog (*screenshot below*) in which linkroles can be created for a taxonomy. Linkroles are stored in the concept definitions file, within the `appinfo` element (*see listing below*). Linkroles are used not only in `definitionLink` elements but also in the containing elements of other relationship kinds (for example, in `calculationLink` and `presentationLink` elements).

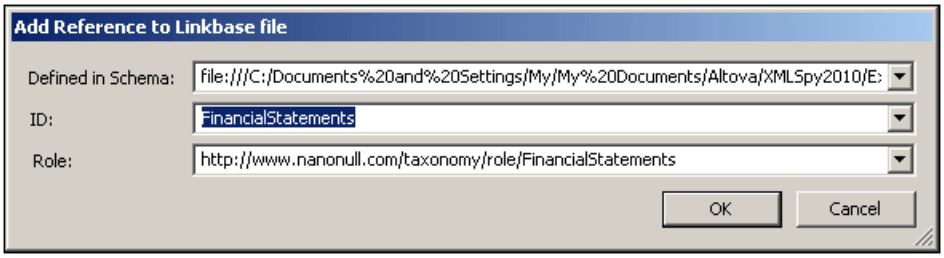
```
<xs:appinfo>
  <link:roleType id="SegmentRevenueAndOperatingIncome"
  roleURI="http://www.nanonull.com/taxonomy/role/SegmentRevenueAndOperatingIncome">
    <link:definition>006091 - Disclosure - Segment Revenue and Operating Income</link:definition>
    <link:usedOn>link:calculationLink</link:usedOn>
    <link:usedOn>link:definitionLink</link:usedOn>
    <link:usedOn>link:presentationLink</link:usedOn>
  </link:roleType>
</xs:appinfo>
```

In the listing above, notice that there are `usedOn` elements that specify in which kind of relationships this linkrole may be used.

In the **Taxonomies tab** of the Link Roles dialog (*screenshot below*), you can add a linkrole to a taxonomy file by clicking the **Add** button. Then define the linkrole's URI and ID (*refer to listing above*). To specify for which kinds of relationships a linkrole should be available, check the boxes of the required relationship kinds. In the Referencing Linkbase Files column, for each linkrole, you can add or delete the linkbase files that reference the linkrole.



The **Linkbases tab** provides another view of the taxonomy's linkroles. In this view, you add and view linkroles according to individual linkbases (for example calculation or presentation linkbases). Select a linkbase in the combo box and then add or delete a linkrole as required. For example, you could add a linkrole to the calculation linkbase. When you click the **Add** button, the Add Reference to Linkbase File dialog (*screenshot below*) pops up.

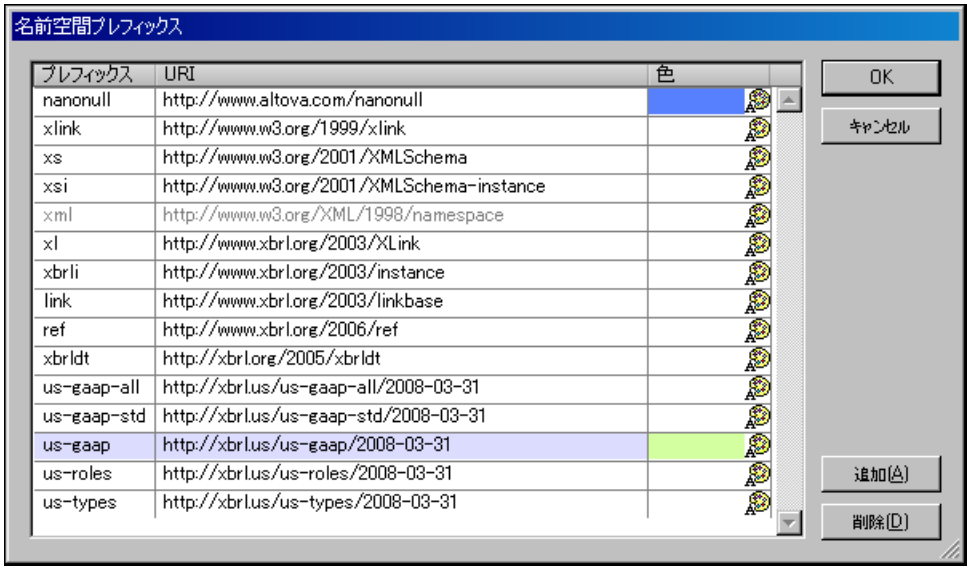


Each of the entries in this dialog is a combo box that enables you to select from available options. The *Defined in Schema* field enables you to select the taxonomy in which the linkrole is defined. The *ID* and *Role* combo boxes provide the available linkroles. After you have selected a linkrole and clicked **OK**, the reference is added to the linkbase. In the Taxonomies tab, the linkrole you referenced will show the referencing linkbase in the Referencing Linkbase Files column. If you wish to make this linkrole available for a particular kind of relationship, you will still, however, have to check the appropriate relationship kind check box.

After a linkrole has been created in the taxonomy it is used when creating relationships.

26.15.3 Namespace Prefixes

The **Namespace Prefixes** command pops up the Namespace Prefixes dialog (*screenshot below)*, which displays all the namespaces in the taxonomy, including those of imported taxonomies. In the Namespace Prefixes dialog you can edit namespaces and prefixes, and set background colors for individual namespaces. When a background color is set for a namespace, elements in that namespaces appear in the Main Window and entry helpers with that background color. Note that a color setting for a given namespace applies for that namespace across all taxonomy documents opened in XBRL View.

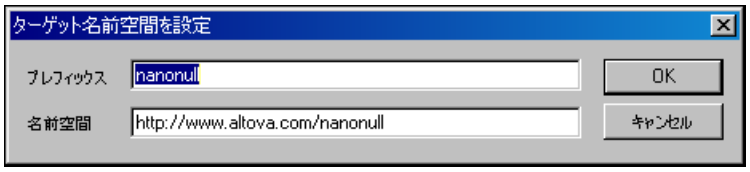


To add or delete a namespace, use the **Add** or **Delete** buttons, respectively. A color is assigned to a namespace via the color palette for that namespace. When you are done with editing in the Namespaces dialog, click **OK** to finish.

The **target namespace** of the taxonomy is also listed in the Namespaces dialog. The target namespace, however, should not be modified in this dialog, but via the **Set Target Namespace** command. For more information on target namespaces, see the XBRL section of the documentation.

26.15.4 Set Target Namespace

The **Set Target Namespace** command enables a target namespace to be set for a taxonomy document. Clicking the command pops up the Set Target Namespace dialog (*screenshot below*). In it you can enter the desired target namespace and a prefix for it. Click **OK** to finish.



The target namespace will be defined and it will also be declared:

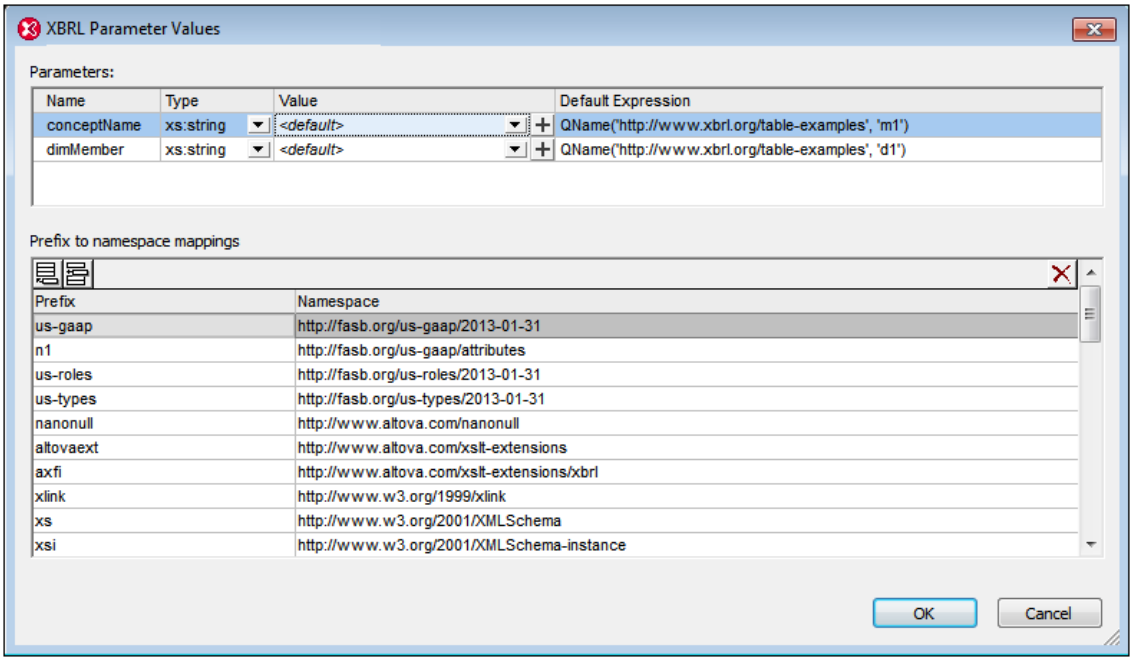
```
<xs:schema targetNamespace="http://www.altova.com/XBRL/Taxonomies"
           xmlns:ns1="http://www.altova.com/XBRL/Taxonomies" >
      ...
</xs:schema>
```

In the listing above, the target namespace is defined with the `targetNamespace` attribute and it is then declared with a prefix of `ns1`.

26.15.5 Parameter Values

The **Parameter Values** command displays the XBRL Parameter Values dialog (*screenshot below*). It displays parameters defined in the Formula tab (formula parameters) and Table tab (table parameters). In the XBRL Parameter Values dialog, you can edit the parameter's datatype

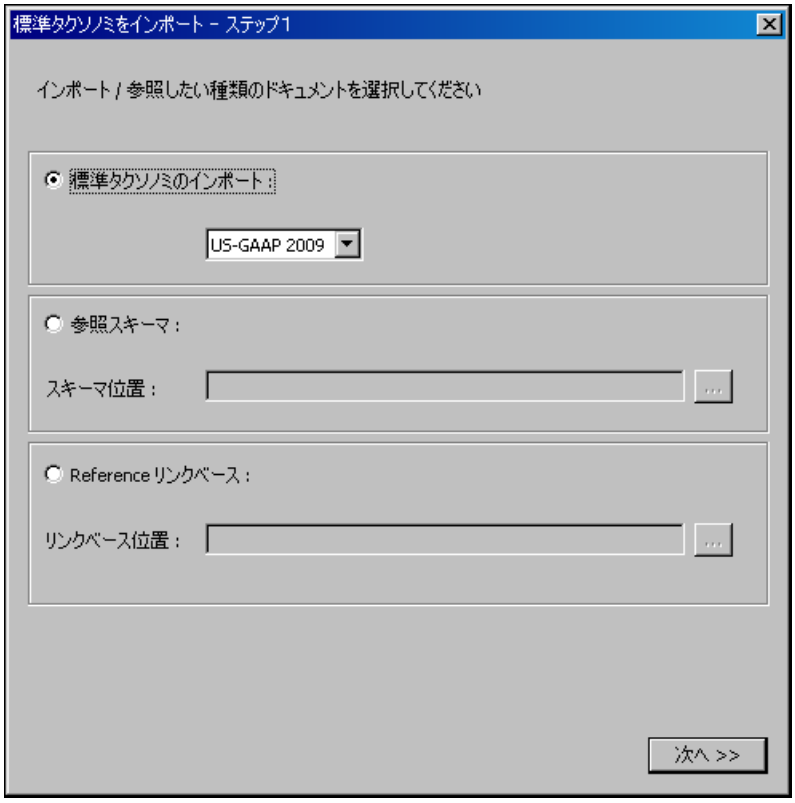
and provide a parameter value that overrides the default value. The parameter value you enter will override the default value that you entered via the diagram. Since table parameters can take multiple values, you can add additional parameter values for a parameter by clicking the **+** icon in the *Value* column.



The values of global parameters as assigned in this dialog are evaluated for table parameters only. Formula parameters, although displayed, are not editable in this dialog.

26.15.6 Import/ Reference

The **Import/Reference** command pops up a dialog (*screenshot below*) in which you can specify the schema to import or the linkbase to reference.



The dialog provides the following radio button options:

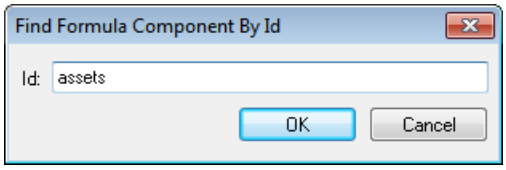
- *Importing a standard taxonomy:* This option enables you to quickly and correctly import a US-GAAP taxonomy or an IFRS taxonomy. Select the required standard taxonomy from the dropdown menu of the combo box and click Next. This takes you to the respective next dialog, for completing your specification of the required taxonomy. The process is described in the section, Creating a New Taxonomy.
- *Importing any taxonomy (Reference Schema):* This option enables you to import any taxonomy by specifying the location of the taxonomy file (`.xsd` file).

```
<xs:import namespace="http://www.altova.com/nanonuTl"
schemaLocation="XBRL%20Examples/NanonuTl/nanonuTl.xsd"/>
```

- *Referencing a linkbase:* A linkbase can be specified for inclusion in the taxonomy. Do this by specifying the location of the linkbase file and clicking **Finish**. A reference to the linkbase file is created in the taxonomy. The relationship type of the newly referenced linkbase can then be specified right-clicking the filename and selecting the **Set Linkbase Kind** command.

26.15.7 Find Component by ID

In taxonomies with large formula or table linkbases containing several components of the same kind, it might be helpful to search for a component by its ID. The menu command **XBRL | Find Component By Id** enables a search by ID. On clicking the command a dialog pops asking for the ID to find (*screenshot below*).



Click **OK** to start he search.

26.15.8 Generate Documentation

The **XBRL | Generate Documentation** command generates detailed documentation of the current XBRL taxonomy. You can output the documentation as an HTML, MS Word, RTF, or PDF file. The documentation generated by this command can be freely altered and used; permission from Altova to do so is not required. Documentation is generated for components you select in the XBRL Taxonomy Documentation dialog (which appears when you select the Generate Documentation command). Related components are hyperlinked in the onscreen output, enabling you to navigate from component to component. The various documentation-generation options are described in the section, Documentation Options.

Note: In order to generate documentation in MS Word format, you must have MS Word (version 2000 or later) installed.

You can either use XMLSpy's fixed standard design for the generated document, or you can use a StyleVision SPS for the design. Using a StyleVision SPS enables you to customize the design of the generated documentation as well as to generate PDF as an additional output format. How to work with an SPS this is explained in the section, User-Defined Design.

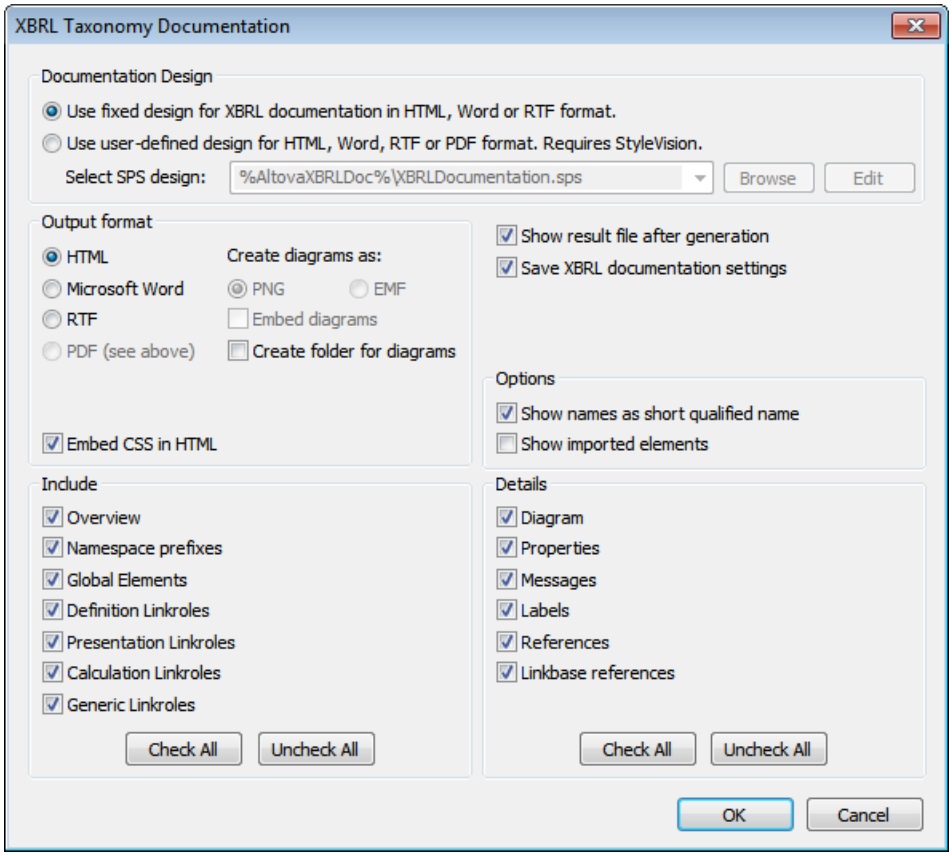
Note: In order to use an SPS to generate schema documentation, you must have StyleVision installed on your machine.

Documentation Options

The **XBRL | Generate Documentation** command pops up the XBRL Taxonomy Documentation dialog (*screenshot below*), in which you can select options for the documentation.

In the Documentation Design pane of the dialog you can select whether to use the fixed standard XMLSpy design for the generated documentation or whether to use a customized design created in a StyleVision SPS. Select the option you want. Note that PDF output is available only for documentation generated with a StyleVision SPS, not for documentation generated using a fixed design. How to work with a user-defined design is described in the section, User-Defined Design.

Note: In order to use an SPS to generate schema documentation, you must have StyleVision installed on your machine.



Clicking the **Generate Documentation** command opens the XBRL Taxonomy Documentation dialog:

- The required format is specified in the Output Format pane: either HTML, Microsoft Word, RTF, or PDF. (The PDF output format is only available if you use a StyleVision SPS to generate the documentation.) On clicking **OK**, you will be prompted for the name of the output file and the location to which it should be saved.
- Microsoft Word documents are created with the `.doc` file extension when generated using a fixed design, and with a `.docx` file extension when generated using a StyleVision SPS. In order to generate documentation in MS Word format, you must have MS Word (version 2000 or later) installed.
- For HTML output, the CSS style definitions can be either saved in a separate CSS file or embedded in the HTML file (in the `<head>` element). If a separate CSS file is created, it will be given the same name as the HTML file, but will have a `.css` extension. Check or uncheck the *Embed CSS in HTML* check box to set the required option.
- The *Embed Diagrams* option is enabled for the MS Word and RTF output options. When this option is checked, diagrams are embedded in the result file, either in PNG or EMF format. Otherwise diagrams are created as PNG or EMF files, which are displayed in the result file via object links.
- When the output is HTML, all diagrams are created as document-external PNG files. If the *Create folder for diagrams* check box is checked, then a folder will be created in the same folder as the HTML file, and the PNG files will be saved inside it. This folder will have a name of the format *`HTMLFilename`*`_diagrams`. If the *Create folder for diagrams* check box is unchecked, the PNG files will be saved in the same folder as the HTML file.
- In the Include pane, you select which items you want to include in the documentation. The *Overview* option lists all components, organized by component type, at the top of the file.

The **Check All** and **Uncheck All** buttons enable you to quickly select or deselect all the options in the pane.

- The Details pane lists the details that may be included for each component. Select the details you wish to include in the documentation. The **Check All** and **Uncheck All** buttons enable you to quickly select or deselect all the options in the pane. The *Messages* check box is only enabled, if *Generic Linkroles* is checked in the Include pane. All other checkboxes are enabled if *Global Elements* or *Generic Linkroles* is checked in the Include pane.
- The *Show Result File* option is enabled for all output options. When this option is checked, the result files are displayed in Browser View (HTML output), MS Word (MS Word output), and the default applications for `.rtf` files (RTF output) and `.pdf` files (PDF output).
- In the Options pane, you can select (i) whether element name should be shown with just a prefix (short qualified name) or in its expanded for (with the full namespace); and (ii) whether imported elements should also be displayed.

Parameter values

If the StyleVision SPS contains one or more parameter definitions, then on clicking **OK**, a dialog pops up listing all the parameters defined in the SPS. You can enter parameter values in this dialog to override the default parameter values that were assigned in the SPS.

User- Defined Design

Instead of the fixed standard XMLSpy design, you can create a customized design for XBRL taxonomy documentation. The customized design is created in a StyleVision SPS, which is a design template for the output document.

Creating the SPS

A StyleVision Power Stylesheet (or SPS) is created using Altova's StyleVision product. An SPS for generating XBRL taxonomy documentation must be based on an XML Schema that specifies the structure of the XBRL taxonomy documentation. This schema is called `XBRLDocumentation.xsd`, and it is delivered with your XMLSpy package. It is stored in the folder: `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Documentation\XBRL`.

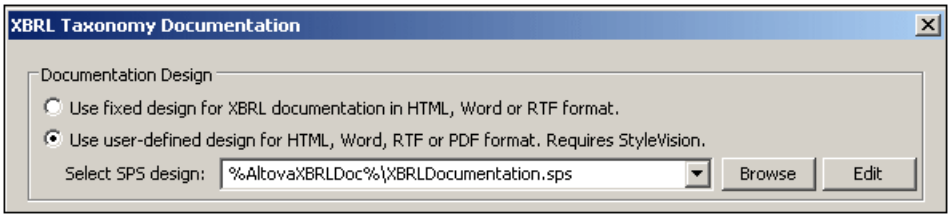
When creating the SPS design in StyleVision, nodes from the `XBRLDocumentation.xsd` schema are placed in the design template and assigned styles and properties. Additional components, like links, tables and images, can also be added to the SPS design. In this way, the entire output document can be designed in the SPS. How to create an SPS design in StyleVision is described in detail in the StyleVision user manual.

The advantage of using an SPS for generating XBRL taxonomy documentation is that you have complete control over the SPS design. Note also that PDF output of the XBRL taxonomy documentation is available only if a user-defined SPS is used; PDF output is not available if the fixed XMLSpy design is used.

Specifying the SPS to use for XBRL taxonomy documentation

After an SPS has been created, it can be used to generate XBRL taxonomy documentation. The SPS you wish to use for generating the XBRL taxonomy documentation is selected in the XBRL Taxonomy Documentation dialog (accessed via the **XBRL | Generate Documentation** command). In the Documentation Design pane of this dialog (*see screenshot below*), select the *Use User-Defined Design* radio button. You can then click the **Browse** button and browse for the SPS you want. Click the dialog's **OK** button, and, in the Save dialog that pops up, select the folder for, and enter the name of, the output file.

Note: The SPS file must correctly locate the schema on which it is based: `XBRLDocumentation.xsd` (*see above*).



One editable SPS design for XBRL taxonomy documentation generation is delivered with XMLSpy. It is named `XBRLDocumentation.sps` and is in the folder: `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2011\Documentation\XBRL\`. This SPS file, together with other SPS files you have recently browsed for, will be available in the combo box of the *Use User-Defined* option (*see screenshot above*).

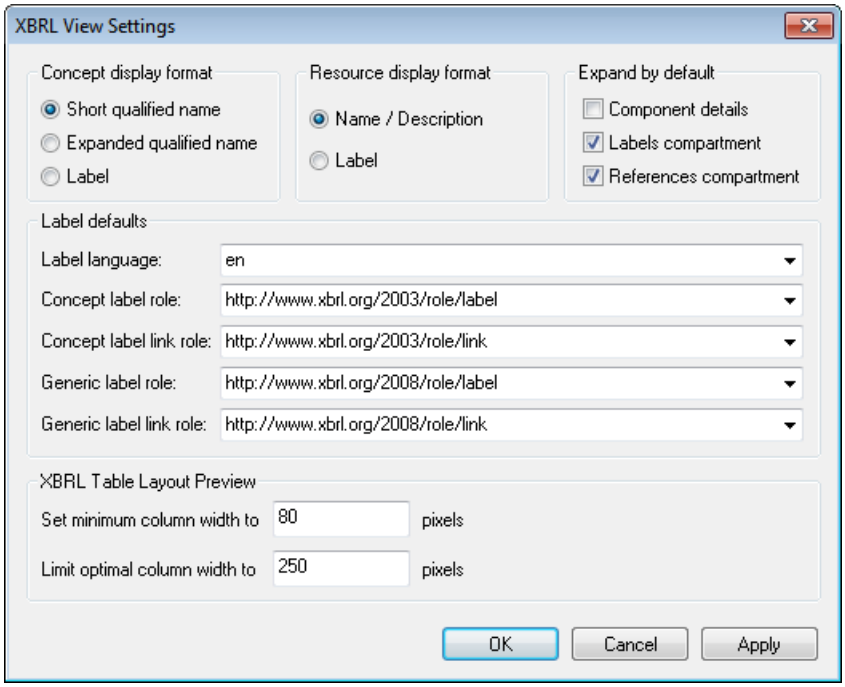
Clicking the **Edit** button in the Documentation Design pane launches StyleVision and opens the selected SPS in a StyleVision window. In order to preview the result document in StyleVision, you will need a Working XML file. A sample XML file for this purpose, called `nanonull.xml`, is supplied with your application and is located in the folder:

`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017 \Documentation\XBRL\SampleData`

Note: In order to use an SPS to generate XBRL taxonomy documentation, you must have StyleVision installed on your machine.

26.15.9 View Settings

The **View Settings** command pops up the XBRL View Settings dialog (*screenshot below*), in which you can specify default settings for XBRL View.



The following settings can be made:

- ***Display of concept names*** can be set to the short or expanded qualified name or to labels. These defaults apply to the Main Window and the Details entry helper. The element-name display setting for the Global Elements entry helper is independent of the settings made in the XBRL View Settings dialog; the settings are in the entry helper's menu bar.
- ***Resource display format:*** In the Formula and Table tabs, resources can be displayed either by their names (if no name has been assigned, the description of the resource is used), or with a label.
- ***Expand by default:*** In the Main Window, element details, the labels box, and the references box can be set by default to the expanded state. Note that, if the labels or references boxes are set to be shown expanded by default, then the expanded boxes will be visible only when the Element details are expanded (either by default or manually). Each time the view is refreshed (for example, when the view is switched from Text View to XBRL View), XBRL View reverts to the default settings.
- ***Label defaults*** specifies the default language and the default label roles to use if labels are not defined. The combo box for each property displays a list of available values.
- ***XBRL Table Layout Preview:*** The minimum and maximum column widths can be set in pixels.

26.15.10 Generate XBRL from DB, Excel, CSV with MapForce

This command starts the generation of an XBRL instance file based on the currently active taxonomy. The data for the instance file is obtained from an MS Excel datasheet, a database, or a CSV file. The XBRL instance file is generated by Altova's MapForce program which you must have installed on your machine. The command starts Altova MapForce and loads the taxonomy into it. You can then specify the source data file and graphically design the required instance file output.

26.15.11 Present XBRL as HTML/ PDF/ Word with StyleVision

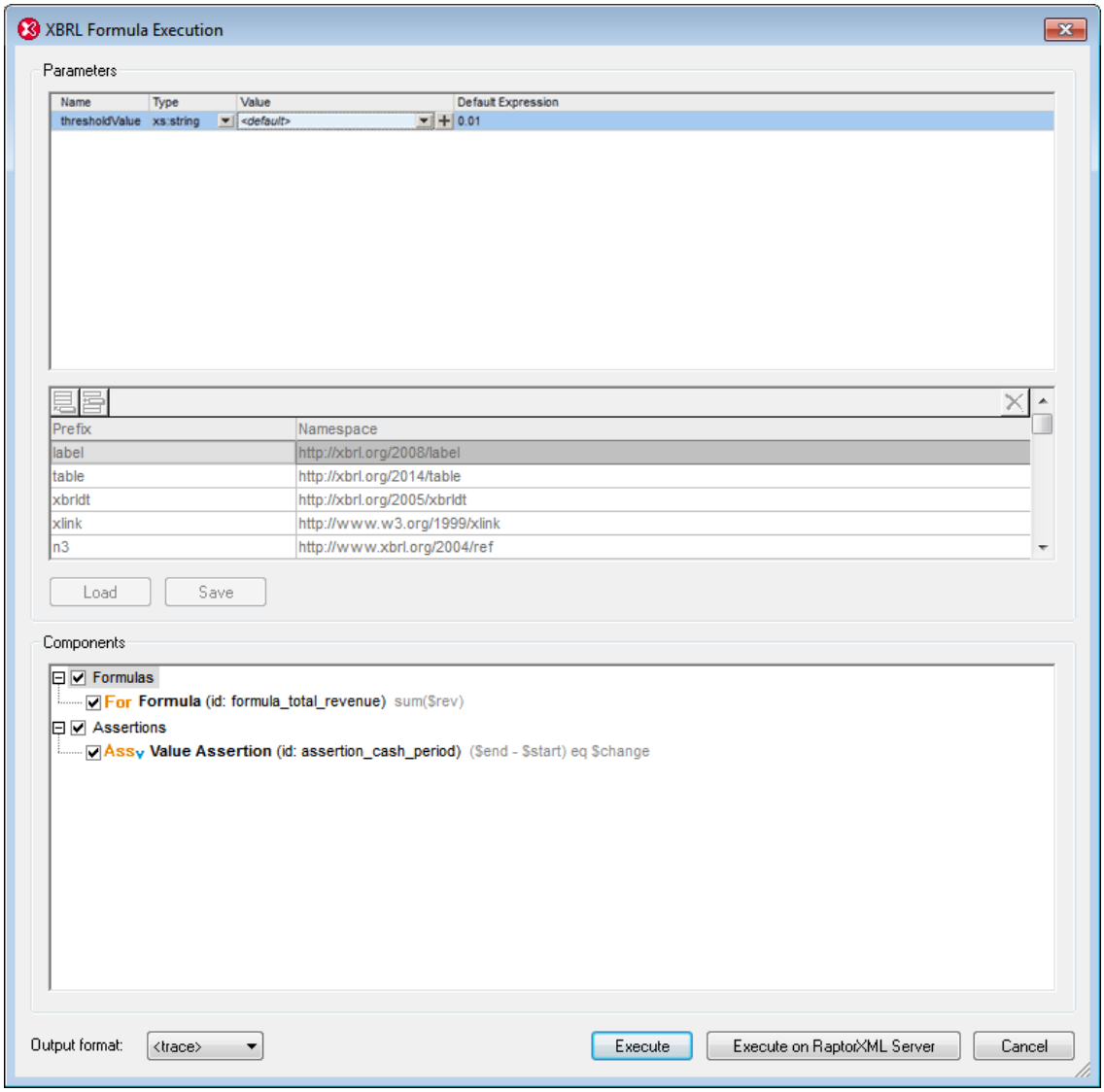
This command loads the currently active taxonomy into Altova StyleVision, which enables you to generate a design for reports based on the taxonomy. In order to use this command, you must have Altova's StyleVision program installed on your machine.

26.15.12 Execute Formula (on Server)

The **Execute Formula** and **Execute Formulas on Server (high-performance)** commands are enabled when an XBRL instance document is the active document in Text View or Grid View. These commands execute formulas and/or assertions defined in the DTS associated with the XBRL instance file. (A DTS, short for Discoverable Taxonomy Set, is a collection of taxonomies.) Formulas are evaluated with data in the XBRL instance file, and the results are output in an XBRL instance file. Assertions are evaluated separately, and the results are output in a JSON or XML file.

The **Execute Formula on Server (high-performance)** command uses an associated RaptorXML+XBRL Server to execute formulas. Use the command **Tools | Manage Raptor Servers** to set up a RaptorXML+XBRL Server.

If there are no formulas or assertions defined in the DTS, a message to this effect is displayed. If there is a valid formula or assertion in the DTS, the XBRL Formula Execution dialog (*screenshot below*) pops up.



Parameters
If parameters are defined in the DTS, each parameter will be displayed in the Parameters pane and a value can be entered for it. Parameter names are read-only. Mandatory parameters are displayed with a red exclamation mark, and the **OK** button is disabled till the parameter is assigned a value. Optional parameters have a default value. If a required type is specified, the type is displayed. Parameters that require multiple values are indicated with a **+** icon, which can be clicked to add a new value. Note that optional parameters without a value will not be passed to the engine for execution. Default values are read-only and will be executed if the user does not enter a value.

Namespace mappings
This table defines prefixes that are used in the QNames of parameters and types. Additional namespaces for use in parameter evaluation may be defined here.

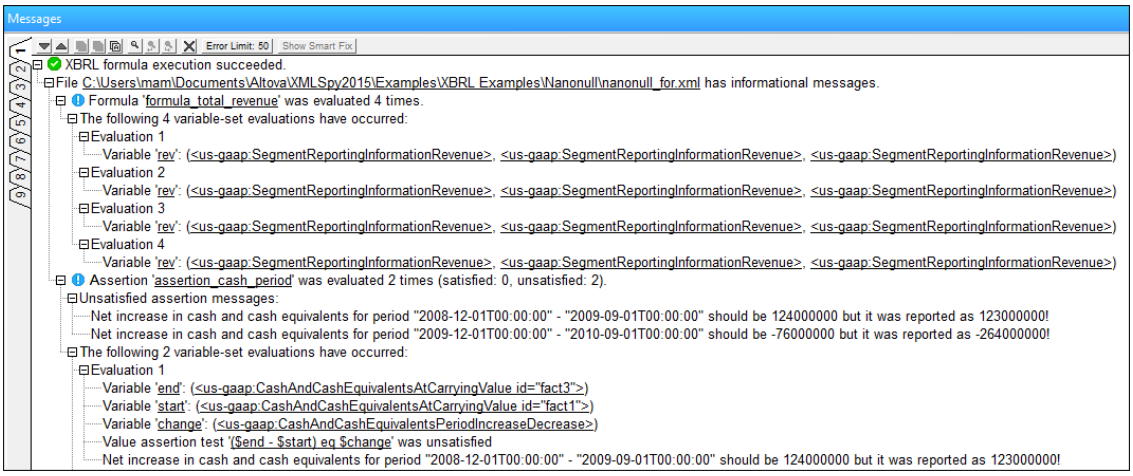
Saving and loading parameters
Parameter settings, including namespace mappings, can be saved in JSON or XML format by clicking the **Save** button. The file format is determined by the file extension given to the file. Note

that optional parameters without a value will not be saved. Once saved, a parameters file can be loaded into the dialog via the **Load** button.

Components
This pane contains a tree view that allows the selection of formula and/or assertion components to be executed. Each item shows an icon and its description, as well as the ID and expression if these are available. To select a component for execution, check its check box. Outputs of assertion executions can be either JSON or XML; select the output format from the *Output Format* combo box. The format of an XBRL formula execution is always XML.

Execution
You can select whether the execution should be done with XMLSpy's internal engine or with Altova's RaptorXML Server. In case of an execution error, an error message is shown in the output window. Otherwise, a success message is displayed. The output files, `assertions-ouput-file.xml/json` and/or `formula-output-file.xml`, are opened in new document windows, not saved to disk. You will need to explicitly save the file to the desired location on disk.

Trace
If you select **<trace>** in the *Output Format* combo box (at the bottom left of the dialog), extra debug information for all "variable set evaluations" will be collected during the formula execution and then be displayed in the Messages window (*see screenshot below*). The trace lists the individual variable set evaluations for each formula/assertion at the points where the actual assignment of the variables in that evaluation are displayed. If the variables reference instance facts, clicking on the values takes you to the corresponding fact element in the instance. Clicking on the formula/assertion or variable name will take you to the corresponding definition in the formula linkbase files. In the case of validation assertions, the assertion messages that have been generated in that evaluation step are also displayed.



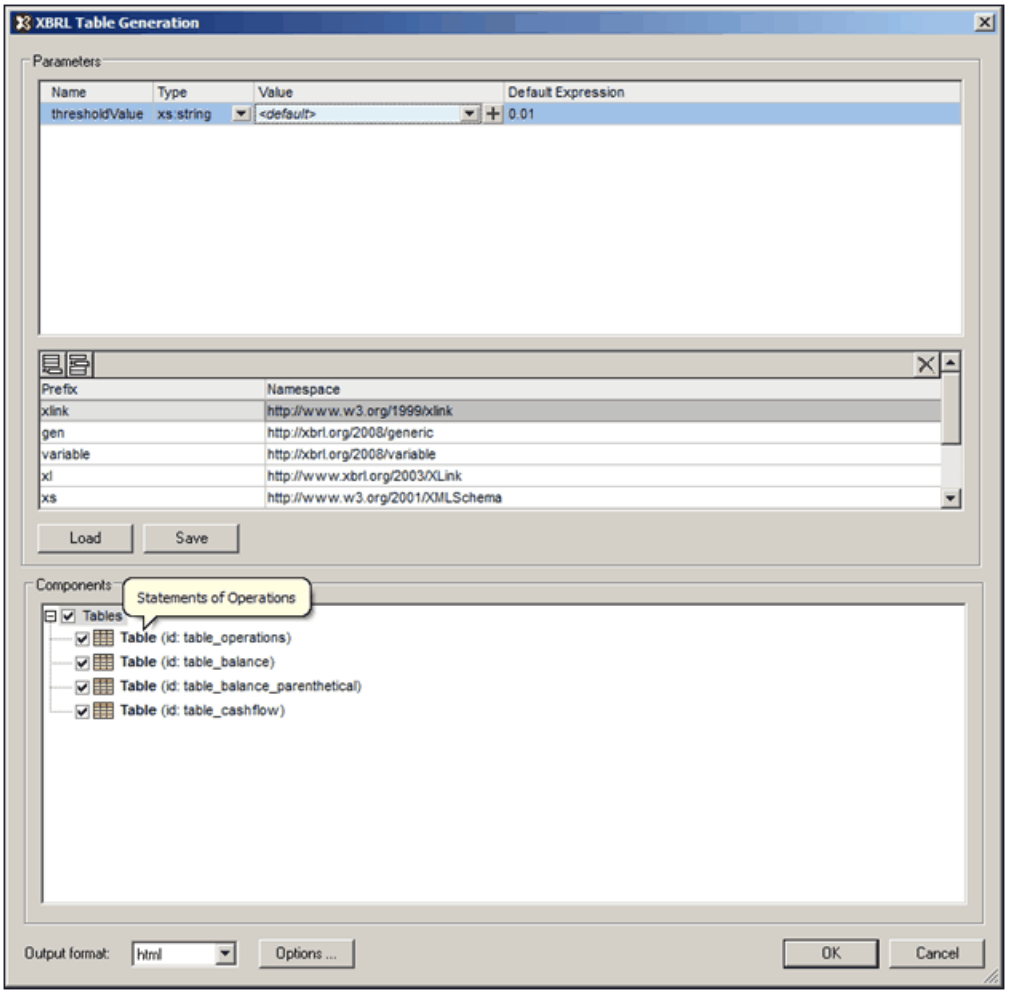
Note: Running a trace can require significant overheads in terms of memory as well as computation speed. When using large XBRL instances, assertions can be evaluated millions of times, and each evaluation might need to store the values of up to 40 variables. So, this feature should only be used for debugging with small/reduced samples, otherwise execution will be slow and XMLSpy might even run out of memory. For this reason, each trace is has a hard-coded limit of 1000 evaluations.

26.15.13 Generate Table (on Server)

The **Generate Table** and **Generate Table on Server (high-performance)** commands are enabled when (i) an XBRL instance document is the active document in Text View or Grid View, or (ii) an XBRL taxonomy is the active document in XBRL View, Text View, Grid View, or Schema View. These commands generate an XML or HTML document containing XBRL tables defined in the DTS associated with the active document. (A DTS, short for Discoverable Taxonomy Set, is a collection of taxonomies.) In the case of XBRL instance files, tables are generated with data in the XBRL instance file.

The **Generate Table on Server (high-performance)** command uses an associated RaptorXML+XBRL Server to generate tables. Use the command **Tools | Manage Raptor Servers** to set up a RaptorXML+XBRL Server.

If there are no tables defined in the DTS, a message to this effect is displayed. If there is a valid table definition in the DTS, the XBRL Table Generation dialog (*screenshot below*) pops up.



Parameters

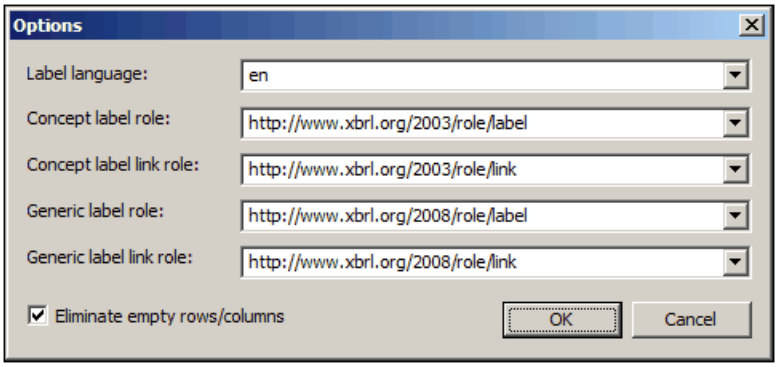
If parameters are defined in the DTS, each parameter will be displayed in the Parameters pane and a value can be entered for it. Parameter names are read-only. Mandatory parameters are displayed with a red exclamation mark, and the **OK** button is disabled till the parameter is

assigned a value. Optional parameters have a default value. If a required type is specified, the type is displayed. Parameters that require multiple values are indicated with a **+** icon, which can be clicked to add a new value. Note that optional parameters without a value will not be passed to the engine for execution. Default values are read-only and will be executed if the user does not enter a value.

Namespace mappings
This table defines prefixes that are used in the QNames of parameters and types. Additional namespaces for use in parameter evaluation may be defined here.

Saving and loading parameters
Parameter settings, including namespace mappings, can be saved in JSON or XML format by clicking the **Save** button. The file format is determined by the file extension given to the file. Note that optional parameters without a value will not be saved. Once saved, a parameters file can be loaded into the dialog via the **Load** button.

Components
This pane contains a tree view that allows the selection of table components to be executed. Each item shows an icon and its description, as well as the ID if this is available. To select a table component for execution, check its check box. Outputs can be either in XML or HTML format; select the output format from the *Output Format* combo box. If the HTML output format is selected, additional options for labels and the treatment of empty rows are available via the **Options** button (*screenshot below*). If preferred label options are available, these are used; otherwise, the defaults specified in the Options dialog are used.



Execution
In case of an execution error, an error message is shown in the output window. Otherwise, a success message is displayed. The output files, `table-ouput-file.xml/html` is opened in new document window, not saved to disk. You will need to explicitly save the file to the desired location on disk.

26.15.14 Transform Inline XBRL

The **Transform Inline XBRL** command is enabled when an XHTML document containing inline XBRL is the active document in Text View, Grid View, or Browser View. The command extracts the Inline XBRL data from the active XHTML document and generates one or more XBRL documents containing the extracted data. The generated XBRL documents are opened in new windows, and can be saved to file. In order for the command to work correctly, all resources referenced by the Inline XBRL document must be available for processing.

26.16 Tools Menu

The Tools menu allows you to:

- Check the spelling of your XML documents
- Access the scripting environment of XMLSpy. You can create, manage and store your own forms, macros and event handlers
- View the currently assigned macros
- Compare any two files to check for differences
- Compare any two folders to check for differences
- Access customized commands that use external applications. These commands can be created in the Tools tab of the Customize dialog..
- Define global resources
- Change the active configuration for global resources in XMLSpy
- Add RaptorXML Servers for XML and XBRL validation, and to configure RaptorXML validation options
- Select a Raptor Server configuration as the active configuration
- Customize your version of XMLSpy: define your own toolbars, keyboard shortcuts, menus, and macros
- Define global XMLSpy settings

26.16.1 Spelling

XMLSpy's spellchecker with built-in language dictionaries (*see note below*) is enabled in Text View, Grid View, and Authentic View. If you wish to spellcheck a document that you have been editing in another view, you can switch to Text View or Grid View and run a spelling check. For example, if you have been editing an XML Schema document in Schema View, switch to Text View or Grid View for the spelling check.

Note: The selection of built-in dictionaries that ship with Altova software does not constitute any language preferences by Altova, but is largely based on the availability of dictionaries that permit redistribution with commercial software, such as the MPL, LGPL, or BSD licenses. Many other open-source dictionaries exist, but are distributed under more restrictive licenses, such as the GPL license. Many of these dictionaries are available as part of a separate installer located at http://www.altova.com/dictionaries. It is your choice as to whether you can agree to the terms of the license applicable to the dictionary and whether the dictionary is appropriate for your use with the software on your computer.

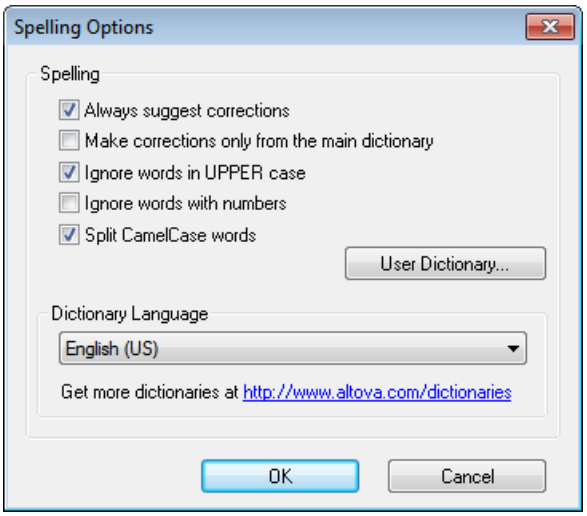
This section describes how to use the spellchecker. It is organized into the following sub-sections:

- Selecting the spellchecker language
- Defining the scope of the check
- Running the spelling check

Selecting the spellchecker language

The spellchecker language can be set as follows:

1. Click the **Tools | Spelling Options** menu command.
2. In the XML Spelling Options dialog that pops up, click the **MoreSpelling Options** button.
3. In the Spelling Options dialog that now pops up (*screenshot below*), select one of the installed dictionaries from the dropdown list of the Dictionary Language combo box.

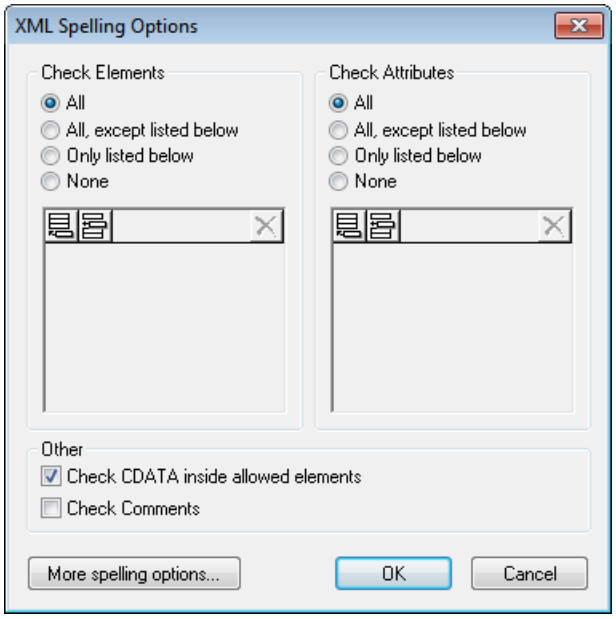


4. Click **OK** to finish.

The dictionary language you selected will be used by the spellchecker for spelling checks. If the language you want is not already installed, you can download additional language dictionaries. How to do this is described in the section, Adding dictionaries for the spellchecker.

Defining the scope of the check

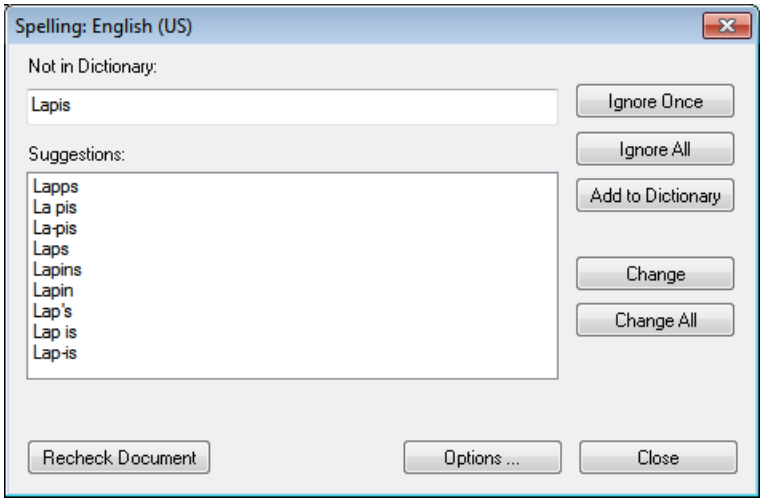
When the spellchecker is run in Text View or Grid View, the scope of the check can be defined immediately before starting the check. Do this by selecting the command **Tools | Spelling Options** and by defining the required scope in the XML Spelling Options dialog that pops up (*see screenshot below*).



You can select which elements and attributes are to be checked and whether CDATA sections and comments should be checked. Also see the description of the Spelling Options command for related information.

Running the spellchecker

The **Tools | Spelling (Shift+F7)** command automatically starts checking the currently active XML document according to the defined scope. If an unknown word is encountered, the *Spelling: Not in Dictionary* dialog pops up (*screenshot below*). Otherwise the spelling check runs through to completion.



The various parts of the *Spelling: Not in Dictionary* dialog and the available options are described below:

Not in Dictionary

This text box contains the word that cannot be found in either the selected language dictionary or user dictionary. The following options are available:

- You can edit the word in the text box manually or select a suggestion from the *Suggestions* pane. Then click **Change** to replace the word in the XML document with the edited word. (Double-clicking a suggestion inserts it directly in the XML document.) When a word is shown in the *Not in Dictionary* text box, it is also highlighted in the XML document, so you can edit the word directly in the document if you like. Clicking **Change All** will replace all occurrences of the word in the XML document with the edited word.
- You can choose to not make any change and to ignore the spellchecker warning—either just for the current occurrence of the word or for every occurrence of it.
- You can add the word to the user dictionary and so allow the word to be considered correct for all checks from the current check onwards.

Suggestions
This list box displays words resembling the unknown word (supplied from the language and user dictionaries). Double-clicking a word in this list automatically inserts it in the document and continues the spellchecking process.

Ignore once
This command allows you to continue checking the document while ignoring the first occurrence of the unknown word. The same word will be flagged again if it appears in the document.

Ignore all
This command ignores all instances of the unknown word in the whole document.

Add to dictionary
This command adds the unknown word to the **user dictionary**. You can access the user dictionary (in order to edit it) via the Spelling Options dialog.

Change
This command replaces the currently highlighted word in the XML document with the (edited) word in the *Not in Dictionary* text box.

Change all
This command replaces all occurrences of the currently highlighted word in the XML document with the (edited) word in the *Not in Dictionary* text box.

Recheck Document
The **Recheck Document** button restarts the check from the beginning of the document.

Options
Clicking the **Options** button opens a dialog box depending on the current view.

- If the current view is Authentic View, the Spelling Options dialog box is opened.
- If the current view is Text View or Grid View, then the XML Spelling Options dialog box is opened.

For more information about these dialog boxes, see the section Spelling Options.

Close
This command closes the Spelling dialog box.

26.16.2 Spelling Options

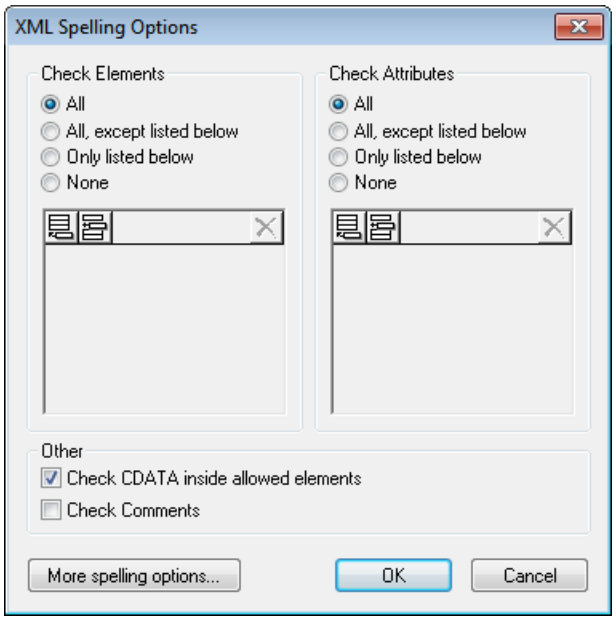
The **Tools | Spelling Options** command opens the **Spelling Options**.Depending on which view is active, the **Tools | Spelling Options** command opens either the **Spelling Options** dialog directly (Schema View, WSDL View, XBRL View, Authentic View, Browser View), or the **XML Spelling Options** dialog (Text View, Grid View,). The XML Spelling Options dialog has a **Spelling Options** button to access the Spelling Options dialog.

The various settings available in these two dialogs are described in the sub-sections of this section:

- Spelling check context
- Spelling options
- Adding dictionaries for the spellchecker
- Working with the user dictionary

XML Spelling Options dialog

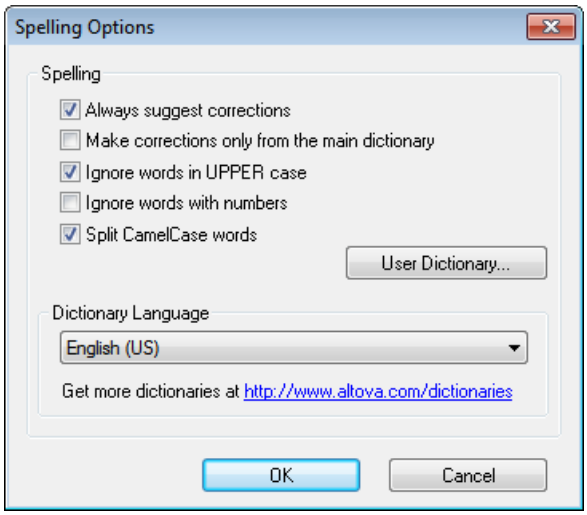
Clicking the **Tools | Spelling Options** command in Text View or Grid View opens the XML Spelling Options dialog (*screenshot below*), in which you can select the scope of the spelling check. You can select which elements and attributes are to be checked and whether CDATA sections and comments should be checked.



You can compile a list of elements and/or attributes that you wish to have spellchecked or have excluded from the spelling check. Alternatively, you can choose to run the spelling check on all elements and/or attributes or on no element or attribute.

Clicking the **More Spelling Options** button at the bottom of the dialog opens the Spelling Options dialog.

Spelling options
The Spelling Options dialog is used to define global spellchecker options.



Always suggest corrections:
Activating this option causes suggestions (from both the language dictionary and the user dictionary) to be displayed in the Suggestions list box. Disabling this option causes no suggestions to be shown.

Make corrections only from main dictionary:
Activating this option causes only the language dictionary (main dictionary) to be used. The user dictionary is not scanned for suggestions. It also disables the **User Dictionary** button, preventing any editing of the user dictionary.

Ignore words in UPPER case:
Activating this option causes all upper case words to be ignored.

Ignore words with numbers:
Activating this option causes all words containing numbers to be ignored.

Split CamelCase words
CamelCase words are words that have capitalization within the word. For example the word "CamelCase" has the "C" of "Case" capitalized, and is therefore said to be CamelCased. Since CamelCased words are rarely found in dictionaries, the spellchecker would flag them as errors. To avoid this, the *Split CamelCase words* option splits CamelCased words into their capitalized components and checks each component individually. This option is checked by default.

Dictionary Language
Use this combo box to select the dictionary language for the spellchecker. The default selection is US English. Other language dictionaries are available for download free of charge from the Altova website.

Adding dictionaries for the spellchecker

For each dictionary language there are two Hunspell dictionary files that work together: a `.aff` file and `.dic` file. All language dictionaries are installed in a `Lexicons` folder at the following location:

On Windows 7 and Windows 8: `C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons`
On Windows XP: `C:\Documents and Settings\All Users\Application Data\Altova\SharedBetweenVersions\SpellChecker\Lexicons`

Within the `Lexicons` folder, different language dictionaries are each stored in different folder: `<language name>\<dictionary files>`. For example, on a Windows 7 or Windows 8 machine, files for the two English-language dictionaries (`English (British)` and `English (US)`) will be stored as below:

```
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (British)\en_GB.aff
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (British)\en_GB.dic
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (US)\en_US.dic
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (US)\en_US.dic
```

In the Spelling Options dialog, the dropdown list of the *Dictionary Language* combo box displays the language dictionaries. These dictionaries are those available in the `Lexicons` folder and have the same names as the language subfolders in the `Lexicons` folder. For example, in the case of the English-language dictionaries shown above, the dictionaries would appear in the Dictionary Language combo box as: *English (British)* and *English (US)*.

All installed dictionaries are shared by the different users of the machine and the different major versions of Altova products (whether 32-bit or 64-bit).

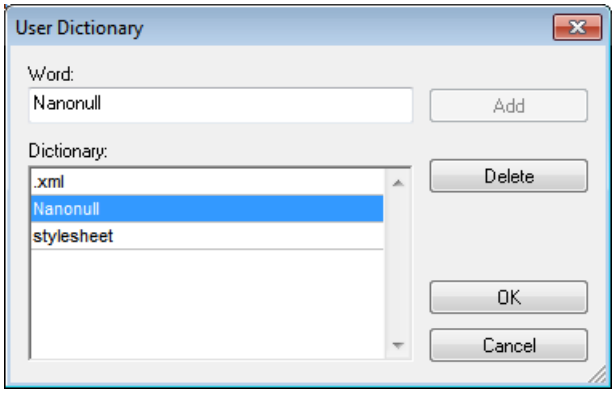
You can add dictionaries for the spellchecker in two ways, neither of which require that the files be registered with the system:

- By adding Hunspell dictionaries into a new subfolder of the `Lexicons` folder. Hunspell dictionaries can be downloaded, for example, from http://wiki.services.openoffice.org/wiki/Dictionaries or http://extensions.services.openoffice.org/en/dictionaries. (Note that OpenOffice uses the zipped `OXT` format. So change the extension to `.zip` and unzip the `.aff` and `.dic` file to the language folders in the `Lexicons` folder. Also note that Hunspell dictionaries are based on Myspell dictionaries. So Myspell dictionaries can also be used.)
- By using the Altova dictionary installer, which installs a package of multiple language dictionaries by default to the correct location on your machine. The installer can be downloaded via the link in the Dictionary language pane of the Spelling Options dialog (*see screenshot below*). Installation of the dictionaries must be done with administrator rights, otherwise installation will fail with an error.

Note: It is your choice as to whether you agree to the terms of the license applicable to the dictionary and whether the dictionary is appropriate for your use with the software on your

computer.

Working with the user dictionary
Each user has one user dictionary, in which user-allowed words can be stored. During a spellcheck, spellings are checked against a word list comprising the words in the language dictionary and the user dictionary. You can add words to and delete words from the user dictionary via the User Dictionary dialog (*screenshot below*). This dialog is accessed by clicking the User Dictionary button in the Spelling Options dialog (*see second screenshot in this section*).



To add a word to the user dictionary, enter the word in the Word text box and click **Add**. The word will be added to the alphabetical list in the Dictionary pane. To delete a word from the dictionary, select the word in the Dictionary pane and click **Delete**. The word will be deleted from the Dictionary pane. When you have finished editing the User Dictionary dialog, click **OK** for the changes to be saved to the user dictionary.

Words may also be added to the User Dictionary during a spelling check. If an unknown word is encountered during a spelling check, then the Spelling dialog pops up prompting you for the action you wish to take. If you click the **Add to Dictionary** button, then the unknown word is added to the user dictionary.

The user dictionary is located at:

On Windows 7 and Windows 8: `C:\Users\<user>\Documents\Altova\SpellChecker\Lexicons\user.dic`
On Windows XP: `C:\Documents and Settings\<user>\My Documents\Altova\SpellChecker\Lexicons\user.dic`

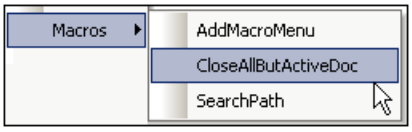
26.16.3 Scripting Editor

The **Scripting Editor** command opens the Scripting Editor window. How to work with the Scripting Editor is described in the Scripting section of this documentation.

Note: The .NET Framework version 2.0 or higher will have to be installed on your machine in order for the Scripting Editor to run.

26.16.4 Macros

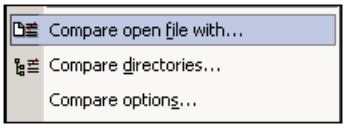
Mousing over the **Macros** command rolls out a submenu containing the macros defined in the Scripting Project that is currently active in XMLSpy (*screenshot below*).



Clicking a macro in the submenu (*see screenshot above*) runs the macro.

26.16.5 Comparisons

XMLSpy provides a comparison (or differencing) feature, with which you can compare XML and Text files, as well as folders, in order to check for differences.



There are the following menu items in the **Tools** menu that enable you to perform comparison tasks on files and folders:

- Compare open file with
- Compare directories
- Compare options

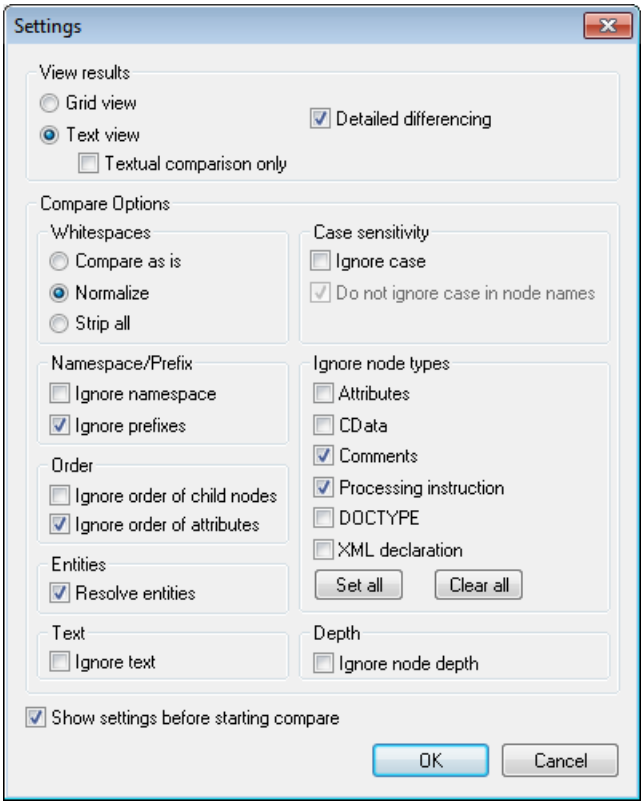
These commands are described in detail in the following sub-sections.

Compare Open File With

This command allows you to compare the open file with another file. The comparison shows both files tiled vertically in the main window with the differences between them highlighted in each file in green. You can compare files as XML documents (where the structure and semantics of tags is significant) or as text documents.

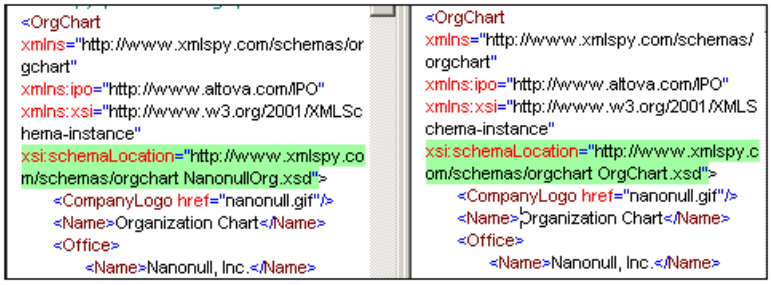
To compare the open file with another file:

1. With the file active in the Main Window (only one open file can be active in the Main Window at a given time), click **Tools | Compare open file with...**.
2. Browse for the file you wish to compare the open file with, and select it. You can also select a file via a global resource or a URL (click the **Browse** button) or a file in one of the open windows in XMLSpy (click the **Window** button).
3. Click **OK**. The second file is opened, and the Settings dialog appears:

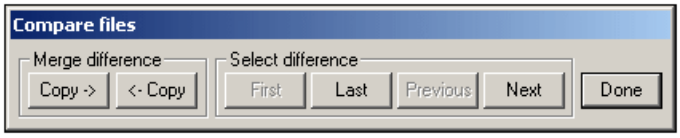


These settings are described under Compare Options. If you do not wish to have this dialog appear each time you start a compare session, uncheck **Show settings before starting compare**, which is at the bottom of the dialog.

4. Select the required settings, then click **OK**. The two files now appear in the Main Window. Lines that are different are highlighted in green. One difference is selected at a time and is indicated in both files in a darker green color. (*Also see the 'Highlight Colors' section below.*)



A Compare Files control window also appears:



To select the next difference, click the **Next** button. The **First**, **Last**, and **Previous** buttons help you to navigate among the differences and to select a difference.
The "Merge difference" pane enables you to copy the selected difference from one file to

the other. In order to enable merging, the following Compare options must be set: Detailed differencing must be checked and Ignore node depth must be unchecked. If you wish to undo a merge, stop the Compare session, select the file in which the change is to be undone, and select **Edit | Undo** or press **Ctrl + Z**.

5. When you are done with reviewing and/or merging differences, click **Done**.

Highlight colors

The differences and merged differences in the two files are indicated by the following highlight colors:

Difference

Current difference

Merged difference

Current merged difference

Note:

- While the Compare session is active, no editing or change of views is allowed. Any attempt to edit or change the view of either file will pop up a message warning that the Compare session will be ended.
- Compare settings can be changed during a Compare session (by selecting **Tools | Compare options...**), but will only take effect from the next Compare session onward; the new settings will not affect the current session.

Compare Directories

The **Compare Directories** command allows you to compare two directories, with or without their sub-directories. Directories are compared to indicate missing files, and whether files of the same name are different or not.

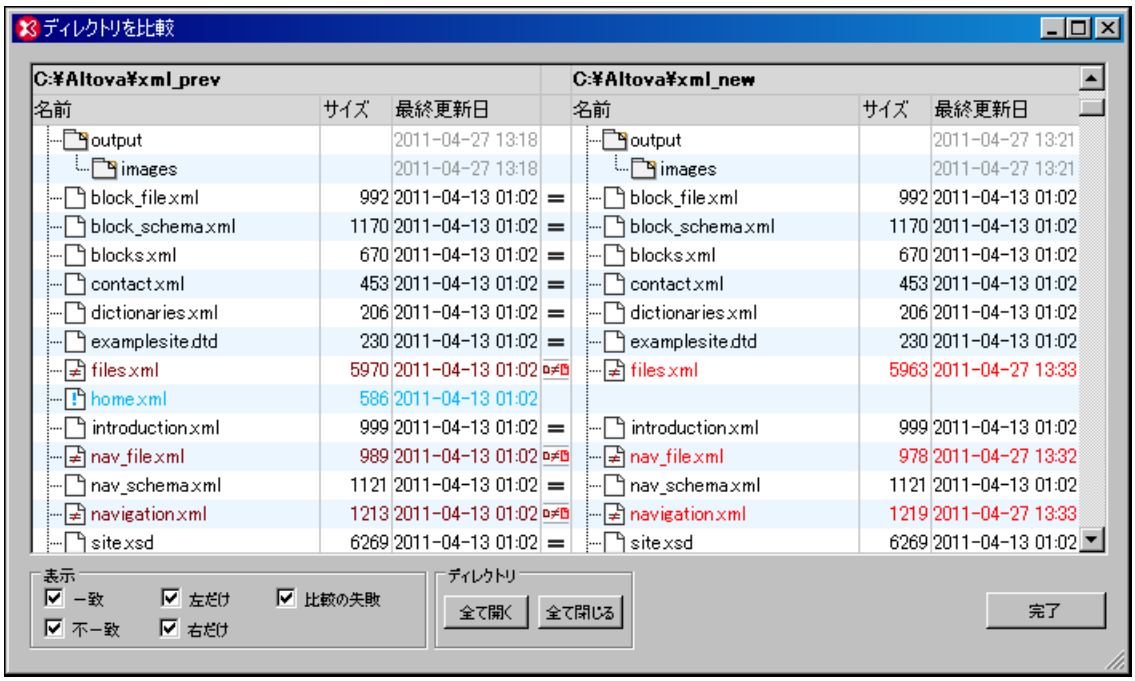
To compare two directories:

1. Click **Tools | Compare directories**. The following dialog appears.

2. Browse for the directories to compare, and check the **Include subdirectories** check box if you wish to include subdirectories in the Compare.
3. Select the file types you want to compare in the **Files of type** field. There are three options in the dropdown menu: (i) XML filetypes; (ii) Filetypes defined in XMLSpy; and (iii) all filetypes. Zip archives are treated as directories, and directories and files in the Zip archive are displayed in the results window. To ensure that Zip archives are read as

directories, make sure that the required extension is included in the Files of Type entry.
4. Click **OK**. The Settings dialog (described in Compare Options) appears.
5. Select the required settings for comparing files.
6. Click **OK**. A dialog will appear indicating the progress of the Compare.

The result will appear in a window, and will look like this:



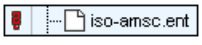
Directory symbols

All directory names are given in black.

- Directory is collapsed and its contents are not displayed.
- Directory is expanded, indicated by the turned down corner. The contents are displayed.
- Directory contains files, all of which either cannot be compared or are not different from the corresponding file in the compared directory.
- Directory contains one or more files that do not exist in the compared directory.
- Directory contains one or more files that are different from the corresponding file in the compared directory.
- Directory contains one or more files that do not exist in the compared directory and one or more files that are different from the corresponding file in the compared directory.

File symbols

The colors in which file names appear depend on their compare status. These colors are noted below.



This file cannot be compared (with the corresponding file in the compared directory). A question mark appears in the middle column. The file name appears in black.

This file is not different from the corresponding file in the compared directory. An equals-to sign appears in the middle column. The file name appears in black.

This file does not exist in the compared directory. The middle column is empty. The file name appears in blue.

This file is different from the corresponding file in the compared directory, and the last modification to this file is more recent than the last modification to the corresponding file. The newer file appears in a brighter red, and the icon shows the brighter red file symbol on the side having the newer file.

Viewing options

- Select what files to show by checking or unchecking the options in the Show pane at the bottom of the Result window.
- Open or close all subdirectories by clicking the appropriate button in the Directories pane.
- Expand or collapse subdirectories by double-clicking the folder icon.
- The Size and Last Modified can be toggled on and off by right-clicking on either file titlebar and clicking Size and Last Modified.

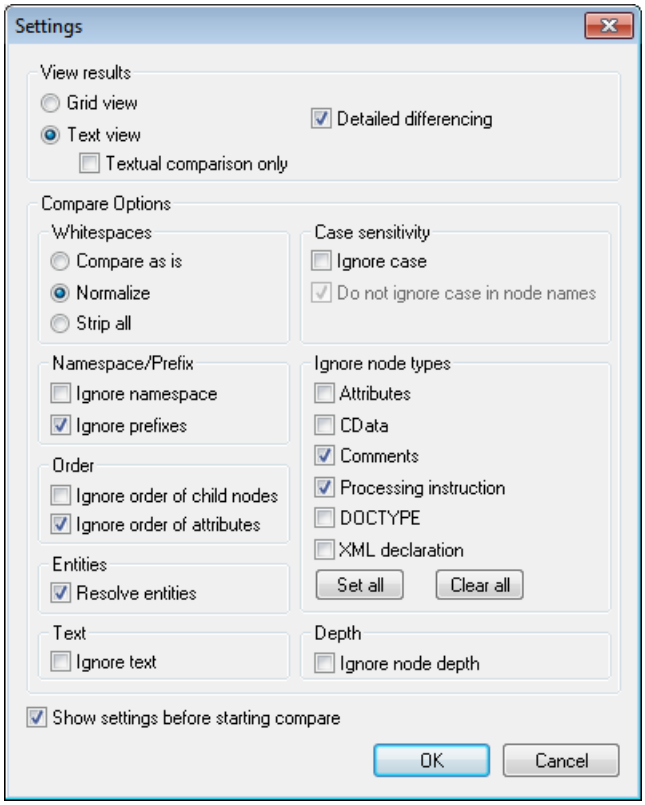
- Change column widths by dragging columns.
- The Result window can be maximized, minimized, and resized.

Comparing and merging files

Double-clicking on a line opens both files on that line in the Main Window, and directly starts a File Compare for the two files. You can then continue as in a regular Compare session (see Compare open file with...).

Compare Options

Click **Tools** | **Compare options** to open the Settings dialog (*see screenshot*). In it you make the settings for your Compare sessions. The settings that are current when a Compare session is started are the settings that are applied to that Compare session.



View results

Selects the view in which results are shown. You can select from the following options:

- Grid View (XML comparison)
- Text View with Textual Comparison Only unchecked (XML comparison)
- Text View with Textual Comparison Only checked (Text comparison)

If a view that provides XML comparison is selected, then the documents are treated as XML documents, and XML Compare Options are enabled. If Text comparison is selected, only Compare Options valid for Text comparison (Whitespaces and Case-sensitivity) are enabled; all other Compare Options are disabled.

Note: You can merge differences in both Grid View and Text View, and in both XML and Text comparison modes. If you wish to undo a merge, stop the Compare session, select the file in which the change is to be undone, and select **Edit | Undo** or press **Ctrl + Z**.

Detailed differencing

If unselected, differences in immediate sibling elements are represented as a single difference, and the merge option is disabled. If selected, differences in immediate siblings are represented as separate differences, and merging is enabled.

Note: The **Detailed differencing** check box must be checked to enable merging.

Whitespaces

Whitespace characters are space, tab, carriage return, and line feed. When whitespace is normalized, consecutive whitespace characters are reduced to one whitespace character; however, note that, according to the XML specification, leading and trailing whitespace in attribute values are completely removed when whitespace is normalized. The options here compare files with: (i) whitespace unchanged; (ii) whitespace normalized; and (iii) all whitespace stripped. The Whitespaces option is available for both XML and Text comparisons.

Case sensitivity

If the **Ignore case** check box is checked, then you have the option of ignoring or not ignoring case in node names (for XML comparisons only). The Case-sensitivity option is available for both XML and Text comparisons.

Namespace/Prefix

These are options for ignoring namespaces and prefixes when searching for differences.

Order

If **Ignore order of child nodes** is checked, then the position of child nodes relative to each other does not matter. The comparison is made for the entire set of child nodes, and if the only difference between a child node in one document and a child node in the compare document is the relative position in the nodeset, then this difference is ignored. Each child element node is identified by its name, its attributes, and its position. If the names of more than one node in the sibling set are the same, then the position of these nodes is used to identify the nodes even if the "Ignore order of child nodes" option is checked. This option applies to each level separately. If **Ignore order of child nodes** is unchecked, then differences in order are represented as differences.

The option of ignoring the **order of attributes** is also available, and applies to the order of attributes of a single element.

Entities

If "Resolve entities" is selected, then all entities in the document are resolved. Otherwise the files are compared with the entities as is.

Text

If "Ignore text" is selected, then differences in corresponding text nodes are not reported.

Ignore node types

Check the node types that will not be compared in the Compare session. Node types that may be ignored are Attributes, CDATA, Comments, Processing Instructions, DOCTYPE statements, and the XML declaration.

Depth

If **Ignore node depth** is checked, then the additional depth of any element (i.e. more levels of descendants) relative to the depth of the corresponding element in the compared file is ignored. This option must be unchecked to enable merging.

Show settings before starting compare

Checking this option causes the Settings dialog (this dialog) to appear before each file or directory comparison is carried out (via the **Compare open file with** and **Compare Directories** commands). Having the Settings dialog appear before each comparison allows you to check and modify the settings for each comparison.

If this command is unchecked, then the Compare session will start directly when a comparison is invoked.

26.16.6 User- defined Tools

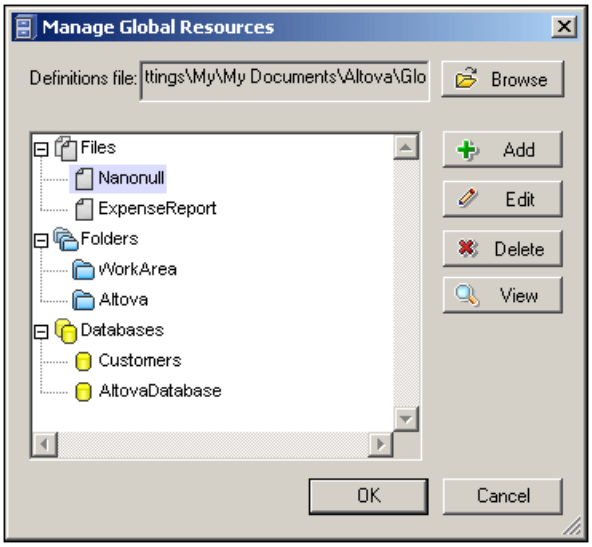
Placing the cursor over the **User-defined Tools** command rolls out a sub-menu containing custom-made commands that use external applications. You can create these commands in the Tools tab of the Customize dialog. Clicking one of these custom commands executes the action associated with this command.

The **User-Defined Tools | Customize** command opens the Tools tab of the Customize dialog (in which you can create the custom commands that appear in the menu of the **User-Defined Tools** command.)

26.16.7 Global Resources

The **Global Resources** command pops up the Global Resources dialog (*screenshot below*), in which you can:

- Specify the Global Resources XML File to use for global resources.
- Add file, folder, and database global resources (or aliases)
- Specify various configurations for each global resource (alias). Each configuration maps to a specific resource.

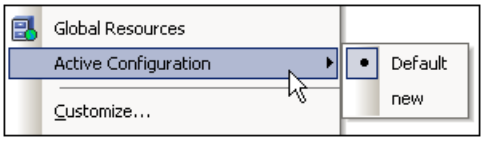


How to define global resources is described in detail in the section, Defining Global Resources.

Note: The Altova Global Resources dialog can also be accessed via the Global Resources toolbar (**Tools | Customize | Toolbars | Global Resources**).

26.16.8 Active Configuration

Mousing over the **Active Configuration** menu item rolls out a submenu containing all the configurations defined in the currently active Global Resources XML File (*screenshot below*).

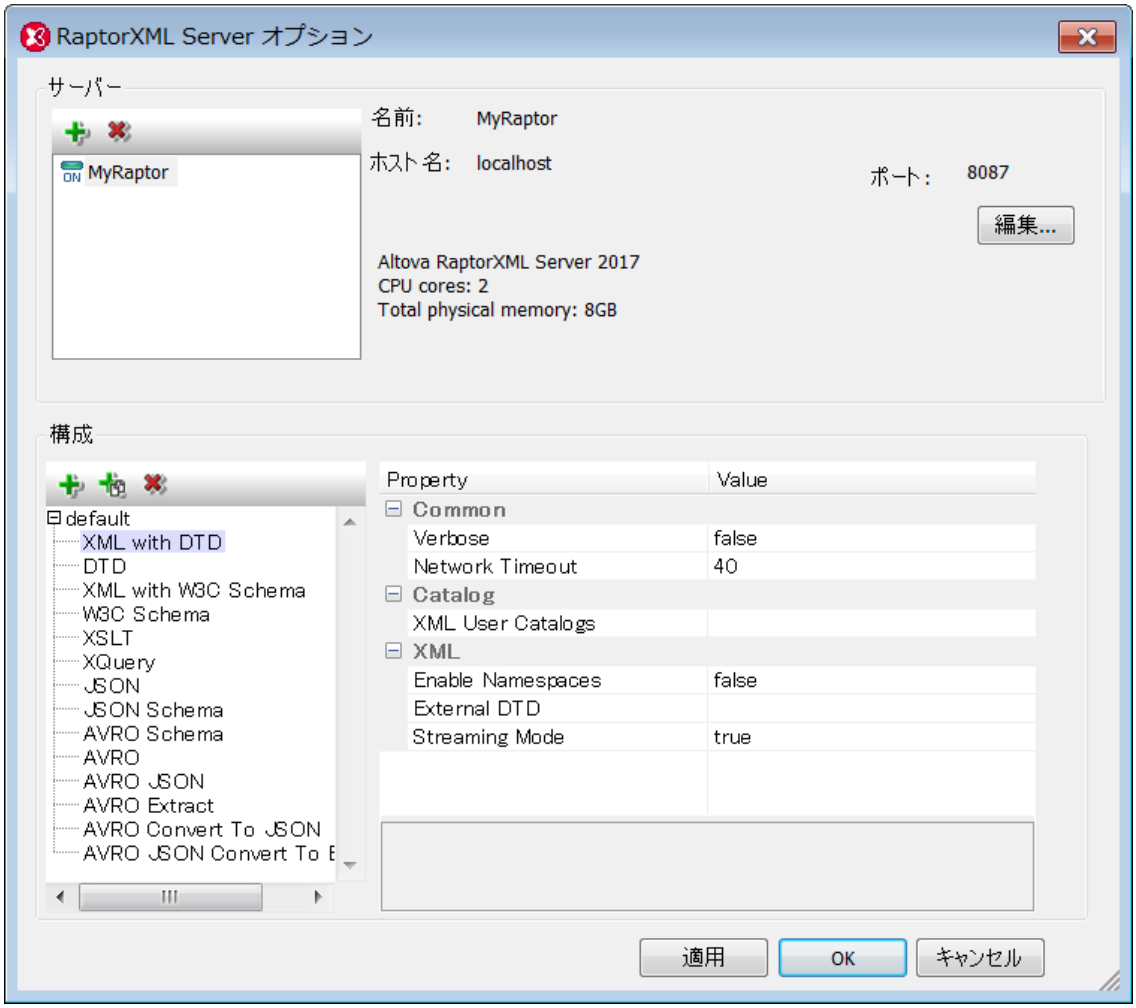


The currently active configuration is indicated with a bullet. In the screenshot above the currently active configuration is `Default`. To change the active configuration, select the configuration you wish to make active.

Note: The active configuration can also be selected via the Global Resources toolbar (**Tools | Customize | Toolbars | Global Resources**).

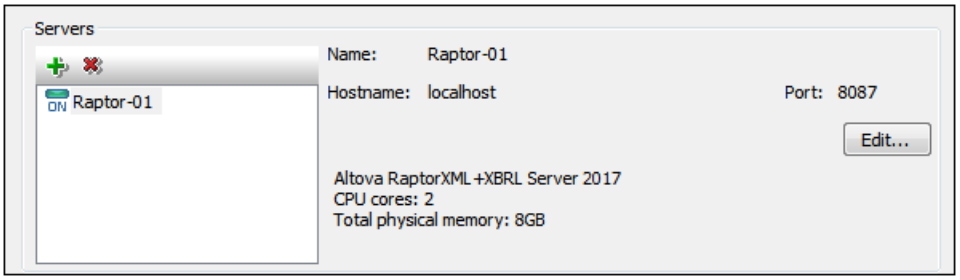
26.16.9 Manage Raptor Servers

The **Manage Raptor Servers** command enables you to add multiple Raptor Servers to the pool of available Raptor Servers and to then define multiple configurations for each server. For an overview of how to use RaptorXML Server for validating XML and XBRL documents, see the topic Validating with RaptorXML Server.



Adding a Raptor Server

In the dialog's *Servers* pane (*screenshot below*), click the **Add Server** icon, then enter the name by which you wish to identify the Raptor server, the network name of the machine on which Raptor is installed (host name), and the port of the Raptor Server. Click **OK** to save the settings.



- *Name:* Any string you choose. It is used in XMLSpy to identify a particular RaptorXML Server.
- *Host name:* The name or IP address of the network machine on which the Raptor server is installed. Processing will be faster if you use an IP address rather than a host name. The

IP address corresponding to `localhost` (the local machine) is `127.0.0.1`.

- *Port:* The port via which the Raptor server is accessed. This port is specified in Raptor's configuration file (called `server_config.xml`). The port must be fixed and known so that requests can be correctly addressed to the service. For more information about the Raptor configuration file, see the user manuals: *RaptorXML Server* and *RaptorXML+XBRL Server*.

After entering the server information, click **OK**. The server name you entered appears in the server list (in the left of the pane). A green icon appears next to the server's name, indicating that the Raptor server has been started and is running. The details of the server are displayed in the pane (*see screenshot above*). A red icon indicates that the server is offline. If the server cannot be found, an error message is displayed.

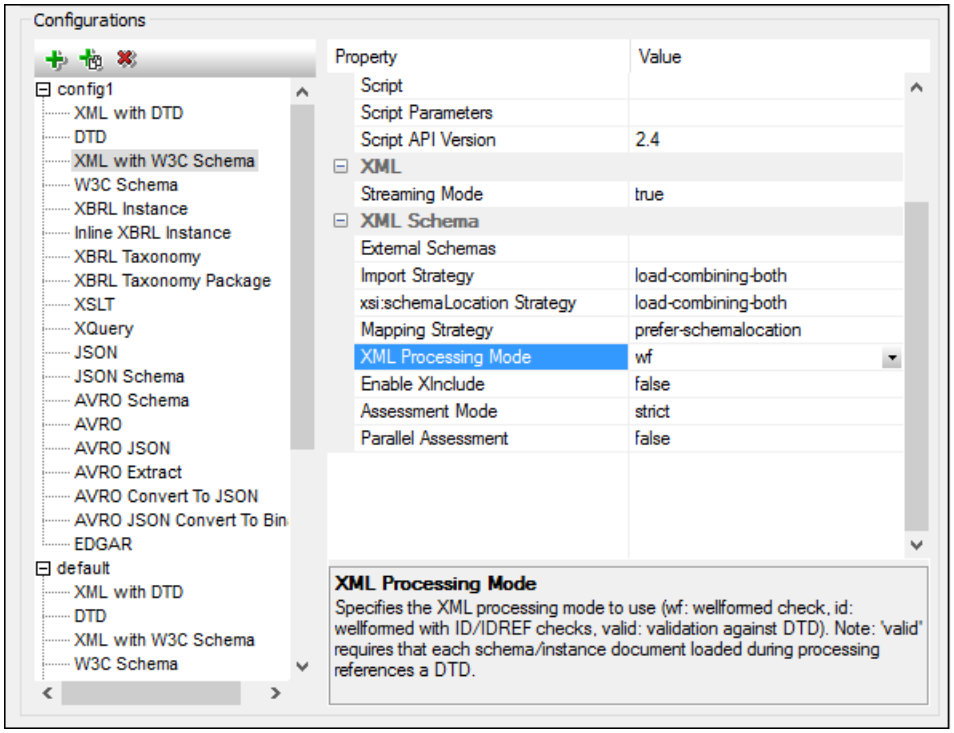
Note: The Raptor server must be running when the server is added. This is necessary so that XMLSpy can obtain information about the server and store it. If, after the server has been added, the server is offline or cannot be found, then these situations are indicated, respectively, by a red icon or an error message.

To edit a server's name, host name, or port, select the server in the left-hand pane, click the **Edit** button, and, in the dialog that appears, edit the information you want to change. To remove a server from the pool, select the server and click the **Remove Selected Server** icon.

Server Configurations

A configuration is a set of RaptorXML validation options. When a server is added, it will have a configuration named `default`. This is a set of RaptorXML options set to their default values. You can edit these values. You can also add new configurations that contain other option values. After you have defined multiple server configurations, you can select one configuration to be the active configuration. This is the configuration that will be used when the **Validate on Server** command is executed.

The *Configurations* pane has two parts: (i) a left-hand pane, which shows the configurations and the types of document that can be validated; (ii) a right-hand pane, which displays the options, organized in groups, of the validation type that is selected in the left-hand pane; at the bottom of the right-hand pane is a description of the selected option (*see screenshot above*).



Adding a configuration

In the *Configurations* pane of the RaptorXML Server Options dialog (*screenshot above*), click **Add a Configuration**. A new configuration is added with default option values. You can also create a new configuration by clicking **Copy Selected Configuration**. This creates a new configuration with option values that are the same as that of the copied configuration. New configurations are created with default names of the type `config<X>`; you can edit the name of a configuration by double-clicking it and entering the new name. You can then edit any of the configuration's option values.

Editing a configuration's option values

First select the validation document in the left-hand pane. This displays the options of that group in the right-hand pane. To edit the value of an option, do one of the following (depending on the type of option value):

- If the value can be one of a set of predefined values, select the value you want from the combo box of that option's value column.
- If the value is not constrained, click in the option's value filed and enter the value you want.
- If the value is a file path, in addition to being able to enter the value, you can also browse for the file you want by using the **Browse** button in the option's value column.

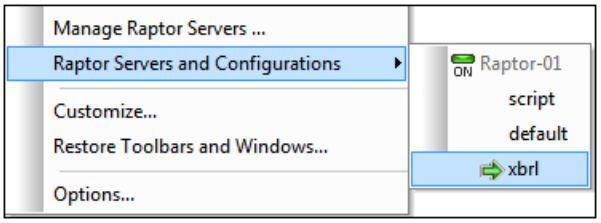
If you select an option, its description is displayed in a box at the bottom of the right-hand pane. For more detailed descriptions of each option, see the command line interface chapters of the RaptorXML Server and RaptorXML(+XBRL) Server user manuals.

Removing a configuration

In the left-hand pane, select the configuration to be removed and click **Remove Selected Configuration**.

26.16.10 Raptor Servers and Configurations

If you have defined multiple configurations on multiple servers, you can select a server and one of its configurations as the active configuration. The active configuration will be used for subsequent validations. On placing the cursor over the **Tools | Raptor Servers and Configurations** command (*see screenshot below*), a submenu appears that contains all the added servers, together with the configuration of each. Select the server configuration you want to make the active configuration. In the screenshot below, the `xbrl` configuration of the server named `Raptor-01` has been selected as the active configuration (indicated by the green arrow).



Note: You can also select the active configuration in the dropdown menu of the **Validate on Server** icon . This menu also has a command to validate EDGAR on the active server.

26.16.11 Customize

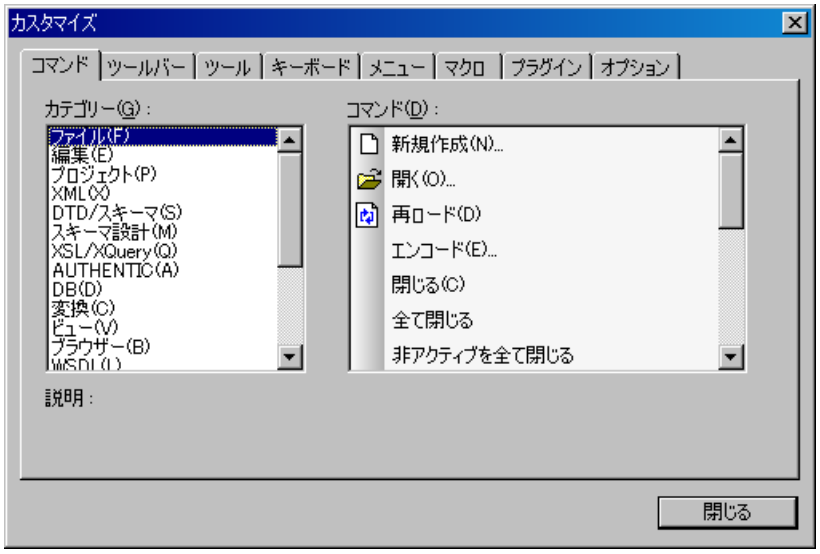
The **Customize** command lets you customize application menus and toolbars to suit your personal needs. Clicking the command pops up the Customize dialog, which has the following tabs:

- Commands: All application and macro commands can be dragged from this tab into menu bars, menus and toolbars.
- Toolbars: Toolbars can be activated, deactivated, and reset individually.
- Tools: Commands that open external programs from within the interface can be added to the interface.
- Keyboard: Keyboard shortcuts can be created for individual application and macro commands.
- Menu: Menu bars and context menus to be customized are selected and made active in this tab. Works together with the Commands tab.
- Macros: Macros can have new commands associated with them.
- Plug-ins: Plug-ins can be activated and integrated in the interface.
- Options: Display options for toolbars are set in this tab.

This section also describes the context menu that appears when the Customize dialog is open and menu bar, menu, or tool bar items are right-clicked.

Commands

The **Commands** tab allows you customize your menus and toolbars. You can add application commands to menus and toolbars according to your preference. Note, however, that you cannot create new application commands or menus yourself.



To add a command to a toolbar or menu:

1. Select the menu item **Tools | Customize**. The Customize dialog appears.
2. Select the **All Commands** category in the *Categories* list box. The available commands appear in the *Commands* list box.
3. Click on a command in the *Commands* list box and drag it to an to an existing menu or toolbar. An I-beam appears when you place the cursor over a valid position to drop the command.
4. Release the mouse button at the position you want to insert the command.

Note the following points.

- When you drag a command, a small button appears at the tip of mouse pointer: This indicates that the command is currently being dragged.
- An "**x**" below the pointer indicates that the command cannot be dropped at the current cursor position.
- If the cursor is moved to a position at which the command can be dropped (a toolbar or menu), the "**x**" disappears and an I-beam indicates the valid position.
- Commands can be placed in menus or toolbars. If you have created you own toolbar, you can use this customization mechanism to populate it.
- Moving the cursor over a closed menu, opens that menu, allowing you to insert the command anywhere in that menu.

Adding commands to context menus

You can also add commands to context menus by dragging commands from the *Commands* list box into the context menu. The procedure is as follows:

1. In the Customize dialog, click the **Menu** tab.
2. In the Context Menu pane, select a context menu from the combo box. The selected context menu pops up.
3. In the Customize dialog,, switch back to the Commands tab.
4. Drag the command you wish to create from the *Commands* list box and drop it into the desired location in the context menu.

Deleting a command or menu

To delete a command from a menu, context menu (see above for details of accessing context menus), or toolbar, or to delete an entire menu, do the following.

1. Open the Customize dialog (**Tools | Customize**). The Customize dialog appears.
2. With the Customize dialog open (and any tab selected), right-click a menu or a menu command, and then select **Delete** from the context menu that pops up. Alternatively, drag the menu or menu command till an "**x**" icon appears below the mouse pointer, and then drop the menu or menu command. The menu or menu command will be deleted.

To re-instate deleted menu commands, use the mechanisms described in this section. To re-instate a deleted menu, go to **Tools | Customize | Menu**, and click the **Reset** button in the *Application Frame Menus* pane. Alternatively, go to **Tools | Customize | Toolbars**, select Menu Bar, and click the **Reset** button.

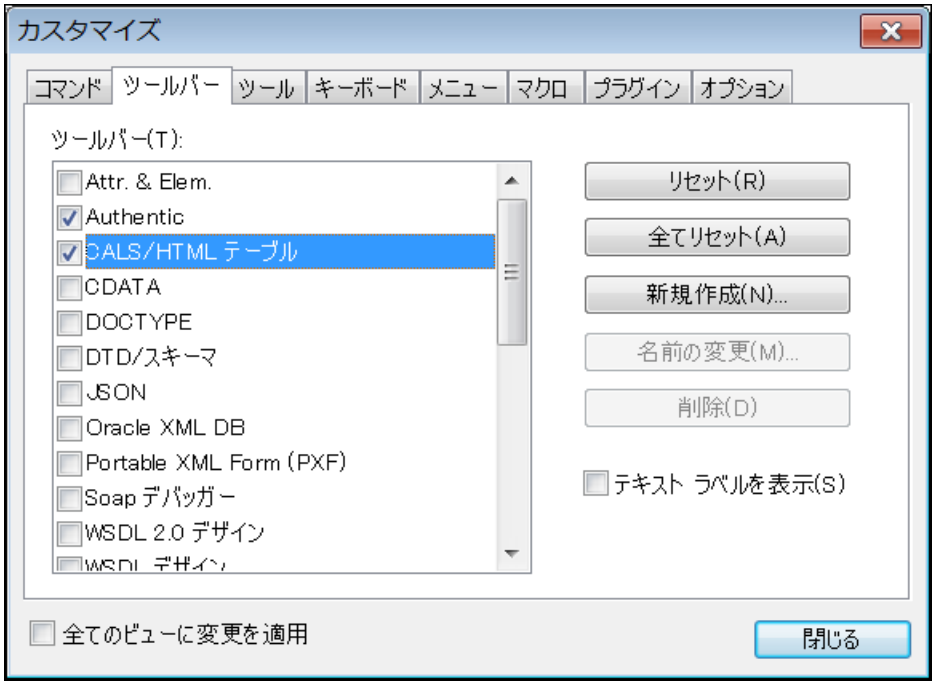
Toolbars

The **Toolbars** tab allows you: (i) to activate or deactivate specific toolbars (that is, to decide which ones to display in the interface); (ii) to set what icons are displayed in each toolbar; and (iii) to create your own specialized toolbars.

The toolbars contain icons for the most frequently used menu commands. Information about each icon is displayed in a tooltip and in the Status Bar when the cursor is placed over the icon. You can drag a toolbar to any location on the screen, where it will appear as a floating window.

Note: To add a command to a toolbar, drag the command you want from the *Commands* list box in the **Commands** tab to the toolbar. To delete a command from a toolbar, open the Customize dialog, and with any tab selected, drag the command out of the toolbar (see **Commands** for more details).

Note: Toolbar settings defined in a particular view are, by default, valid for that view only. To make the settings apply to all views, click the check box at the bottom of the dialog.

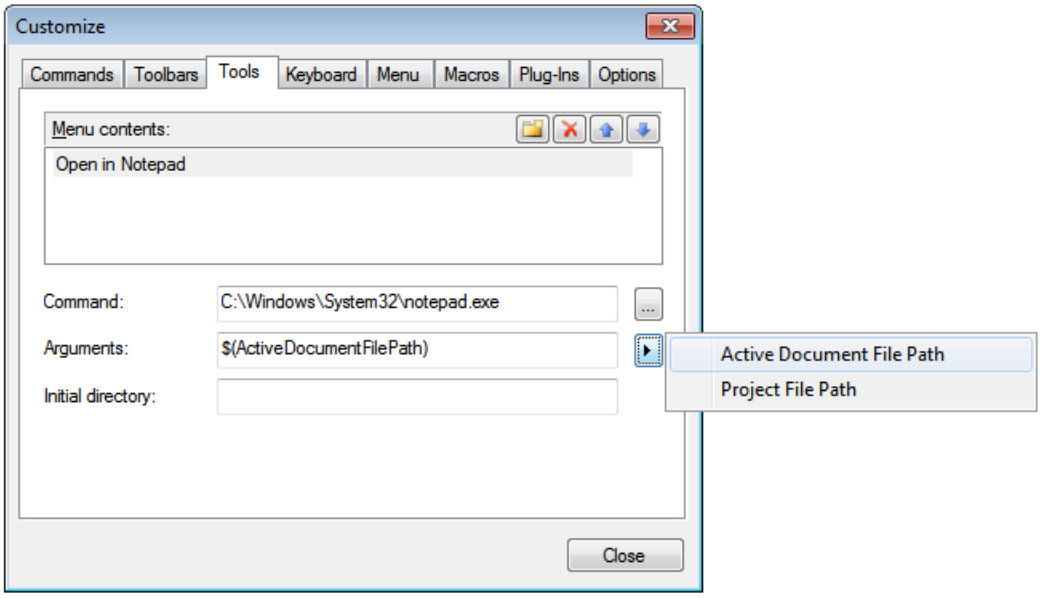


The following functionality is available:

- *To activate or deactivate a toolbar:* Click its check box in the *Toolbars* list box.
- *To apply changes to all views:* Click the check box at the bottom of the dialog. Otherwise, changes are applied only to the active view. Note that only changes made **after** clicking the *All Views* check box will apply to all views.
- *To add a new toolbar*: Click the **New** button and give the toolbar a name in the Toolbar Name dialog that pops up. From the **Commands** tab drag commands into the new toolbar.
- *To change the name of an added toolbar:* Select the added toolbar in the Toolbars pane, click the **Rename** button, and edit the name in the Toolbar Name dialog that pops up.
- *To reset the Menu bar*: Select the *Menu Bar* item in the Toolbars pane, and then click **Reset**. This resets the Menu bar to the state it was in when the application was installed.
- *To reset all toolbar and menu commands:* Click the **Reset All** button. This resets all toolbars and menus to the states they were in when the application was installed.
- *To delete a toolbar:* Select the toolbar you wish to delete in the Toolbars pane and click **Delete**.
- *To show text labels of commands in a particular toolbar*: Select that toolbar and click the *Show Text Labels* check box. Note that text labels have to be activated for each toolbar separately.

Tools

The **Tools** tab allows you to set up commands to use external applications from within XMLSpy. These commands will be added to the **Tools | User-defined Tools** menu. For example, the active file in the main window of XMLSpy can be opened in an external application, such as Notepad, by clicking a command in the **Tools | User-defined Tools** menu that you created.



To set up a command to use an external application, do the following:

1. In the *Menu Contents* pane (*see screenshot above*), click the **New** icon in the title bar of the pane and, in the item line that is created, enter the name of the menu command you want. In the screenshot above, we have entered a single menu command, **Open in Notepad**. We plan to use this command to open the active document in the external Notepad application. More commands can be added to the command list by clicking the **New** icon. A command can be moved up or down the list relative to other commands by using the **Move Item Up** and **Move Item Down** icons. To delete a command, select it and click the **Delete** icon.
2. To associate an external application with a command, select the command in the *Menu Contents* pane. Then, in the *Command* field, enter the path to, or browse for, the executable file of the external application. In the screenshot above, the path to the Notepad application has been entered in the *Command* field.
3. The actions available to be performed with the external application are displayed when you click the flyout button of the *Arguments* field (*see screenshot above*). These actions are described in the list below. When you select an action, a code string for the action is entered in the *Arguments* field.
4. If you wish to specify a current working directory, enter it in the *Initial Directory* field.
5. Click **Close** to finish.

The command/s you created will appear in the **Tools | User-defined Tools** menu, and in the context menu of Project window files and folders—in the **User-defined Tools** submenu.

When you click the command (in the **Tools | User-defined Tools** menu) that you created, the action you associated with the command will be executed. The command example shown in the screenshot above does the following: It opens, in Notepad, the document that is active in the Main Window of XMLSpy. The external application command is also available in the context menu of files in the Project window (right-click a file in the Project window to display that file's context menu). Via the Project Window you can also open multiple files (for applications that allow this) by making a multi-selection and then selecting the command from the context menu.

Arguments

The *Arguments* field specifies the action to be executed by the external application command. The following arguments are available.

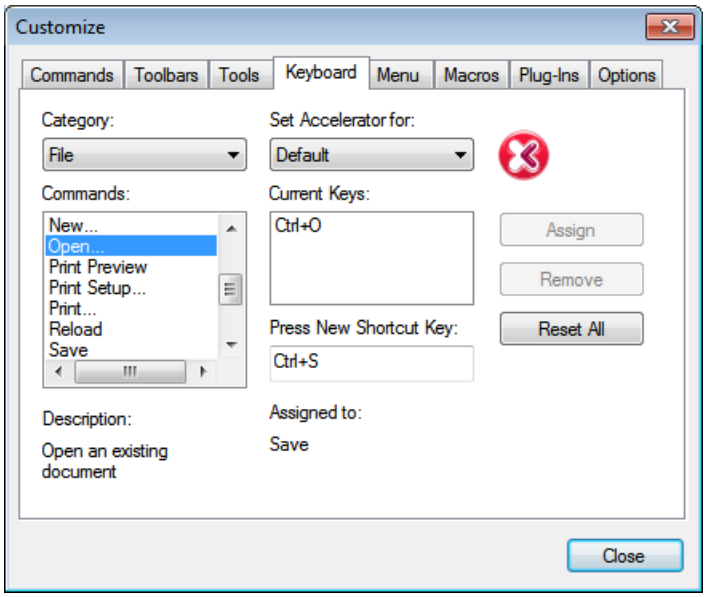
- *Active Document File Path:* The command in the **Tools | User-defined Tools** menu opens the document that is active in XMLSpy in the external application. The command in the context menu of a file in the Project window opens the selected file in the external application.
- *Project File Path:* Opens the XMLSpy project file (the `.spp` file) in the external application.

Initial directory

The *Initial Directory* entry is optional and is a path that will be used as the current directory.

Keyboard

The **Keyboard** tab allows you to create new keyboard shortcuts, or change existing shortcuts, for any application command.



To assign a new shortcut to a command, or to change an existing shortcut, do the following.

1. Select the *All Commands* category in the *Category* combo box. Note that if a macro has been selected as an Associated Command, then macros are also available for selection in the *Category* combo box and a shortcut for the macro can be set.
2. In the *Commands* list box, select the command to which you wish to assign a new shortcut or select the command the shortcut of which you wish to change.
3. Click in the *Press New Shortcut Key* text box, and press the shortcut you wish to assign to that command. The shortcut appears in the *Press New Shortcut Key* text box. If the shortcut has not yet been assigned to any command, the **Assign** button is enabled. If the shortcut has already been assigned to a command, then that command is displayed below the text box and the **Assign** button is disabled. (To clear the *Press New Shortcut Key* text box, press any of the control keys, **Ctrl**, **Alt** or **Shift**).
4. Click the **Assign** button to assign the shortcut. The shortcut now appears in the *Current Keys* list box. You can assign multiple shortcuts to a single command.
5. Click the **Close** button to confirm.

Deleting a shortcut

A shortcut cannot be assigned to multiple commands. If you wish to delete a shortcut, click it in the Current Keys list box and then click the **Remove** button.

Set accelerator for

Currently, accelerators can be set only as default. No other mode is available.

Default keyboard shortcuts

The default shortcuts of commonly used commands are listed below. An overview of all the application's menu commands is available in the Keyboard Map (**Help | Keyboard Map**).

Function-key shortcuts (incl. for validation and transformation)

F1	Help Menu
F1 + Alt	Open Last File
F3	Find Next
F4 + CTRL	Close Active Window
F4 + Alt	Close XMLSpy
F5	Refresh
F6 + CTRL	Cycle through Open Windows
F7	Check Well-formedness
F8	Validate
F10	XSL Transformation
F10 + CTRL	XSL:FO Transformation

File and Application commands

Alt + F1	Open Last File
CTRL + O	File Open
CTRL + N	File New
CTRL + P	File Print
CTRL + S	File Save
CTRL + F4	Close Active Window
CTRL + F6	Cycle through Open Windows
CTRL + TAB	Switch between Open Documents
Alt + F4	Close XMLSpy

Numeric keypad shortcuts (to expand/collapse nodes)

Num +	Expand
Num *	Expand Fully
Num –	Collapse

CTRL + Num –	Collapse Unselected

Miscellaneous keys

Up/Down Arrow Keys	Move Cursor or Selection Bar
Esc	Abandon Edits or Close Dialog Box
Return	Confirm Selection
Del	Delete Character or Selected
Shift + Del	Cut

Editing commands

CTRL + A	Select All
CTRL + F	Find
CTRL + G	Go to Line/Char
CTRL + H	Replace
CTRL + V	Paste
CTRL + X	Cut
CTRL + Y	Redo
CTRL + Z	Undo

Text View commands

CTRL + E	Jump between Start/End Tags
CTRL + Shift + E	Select Element that Contains Cursor
CTRL + Alt + E	Go to Parent Element
CTRL + "+"	Zoom In
CTRL + "–"	Zoom Out
CTRL + 0	Reset Zoom
CTRL + mousewheel forwd	Zoom In
CTRL + mousewheel back	Zoom Out

Grid View commands

CTRL + D	Append CDATA
CTRL + E	Append Element
CTRL + I	Append Attribute
CTRL + M	Append Comment
CTRL + T	Append Text
CTRL + Shift + D	Insert CDATA
CTRL + Shift + E	Insert Element
CTRL + Shift + I	Insert Attribute
CTRL + Shift + M	Insert Comment
CTRL + Shift + T	Insert Text

CTRL + Alt + D	Add Child CDATA
CTRL + Alt + E	Add Child Element
CTRL + Alt + I	Add Child Attribute
CTRL + Alt + M	Add Child Comment
CTRL + Alt + T	Add Child Text

- *Schema View commands*

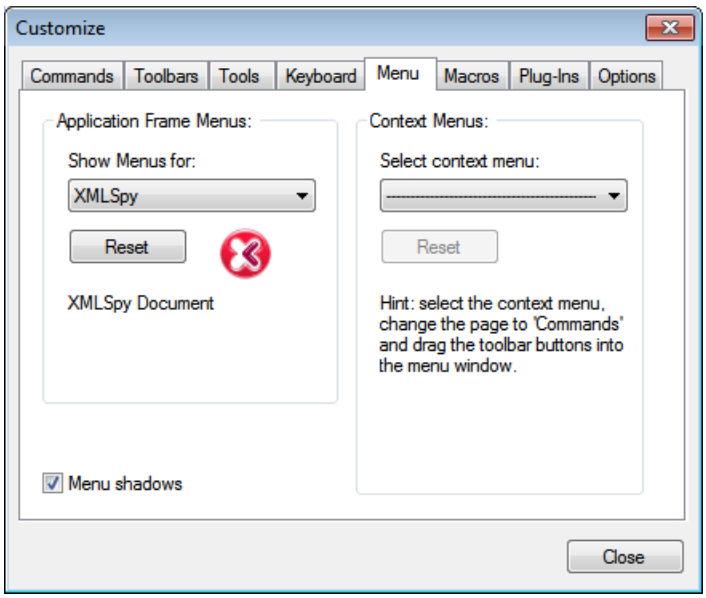
CTRL + Dbl-click Element	Display Element Definition

- *Debugger commands*

F9	Insert/Remove Breakpoint
F9 + Shift	Insert/Remove Tracepoint
F9 + CTRL	Enable/Disable Breakpoint
F9 + Shift + CTRL	Enable/Disable Tracepoint
F11	Step Into
F11 + Shift	Step Out
F11 + CTRL	Step Over
F11 + Alt	Start Debugger/Go

Menu

The **Menu** tab allows you to customize the two main menu bars (default and application menu bars) as well as the application's context menus.



Customizing the default menu bar and application menu bar

The default menu bar is the menu bar that is displayed when no document is open in the main window. The application menu bar is the menu bar that is displayed when one or more documents are open in the main window. Each menu bar can be customized separately, and customization changes made to one do not affect the other.

To customize a menu bar, select it in the *Show Menus For* combo box (*see screenshot above*). Then switch to the Commands tab of the Customize dialog and drag commands from the Commands list box to the menu bar or into any of the menus.

Deleting commands from menus and resetting the menu bars

To delete an entire menu or a command inside a menu, select that menu or menu command, and then either (i) right-click and select **Delete**, or (ii) drag away from the menu bar or menu, respectively.

You can reset each of these two menu bars (default and application menu bars) to its original installation state by selecting the menu in the *Show Menus For* combo box and then clicking the **Reset** button below the combo box.

Customizing the application's context menus

Context menus are the menus that appear when you right-click certain objects in the application's interface. Each of these context menus can be customized by doing the following:

1. Select the context menu you want in the *Select Context Menu* combo box. This pops up the context menu.
2. Switching to the Commands tab of the Customize dialog.
3. Drag a command from the *Commands* list box into the context menu.
4. If you wish to delete a command from the context menu, right-click that command in the context menu, and click **Delete**. Alternatively, you can drag the command you want to delete out of the context menu.

You can reset any context menu to its original installation state by selecting it in the *Select Context Menu* combo box and then clicking the **Reset** button below the combo box.

Menu shadows

Click the *Menu shadows* check box to give all menus shadows.

Macros

The **Macros** tab allows you to create application commands for macros that were created using XMLSpy's Scripting Editor. These application commands (which run the macros associated with them) can subsequently be made available in menus and toolbars, either from the Macros tab directly or by using the mechanisms available in the Commands tab of the Customize dialog. As application commands, they can also be assigned shortcuts in the Keyboard tab of the Customize dialog.

How macros work in XMLSpy

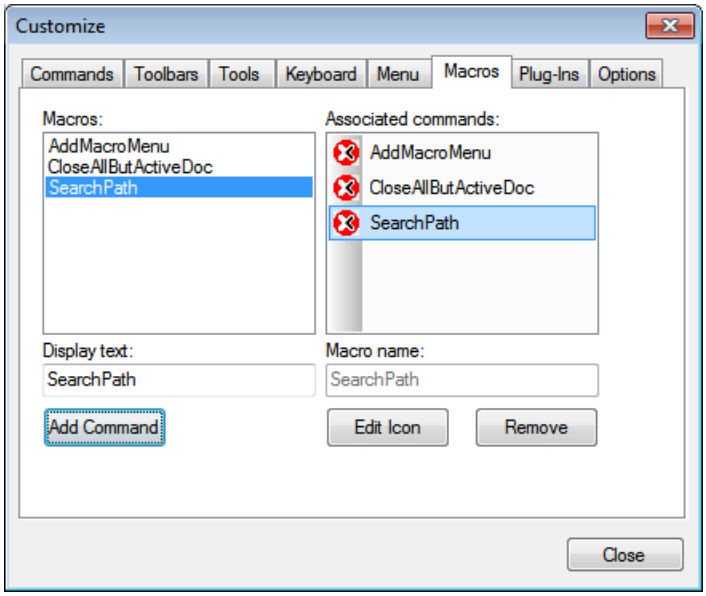
Macros in XMLSpy work as follows:

- Altova scripting projects (`.asprj` files) are created in XMLSpy's Scripting Editor. It is these scripting projects that can contain the macros used in XMLSpy.
- Two scripting projects can be active at a time in XMLSpy: (i) An application scripting project, which is specified in the Scripting tab of the Options dialog, and (ii) The scripting project of the active XMLSpy project, which is specified in the Script Settings dialog (**Project | Script Settings**).
- The macros in these two scripting projects are available in the application: in the **Project | Macros** menu (from where the macros can be run), and in the Macros tab of the Customize dialog (*screenshot below*), in which they can be set as application commands. After a macro has been set as an application command, the command can be placed in a menu and/or toolbar.

Creating an application command for a macro

In Scripting Editor (**Tools | Scripting Editor**) create the macro you wish and save it to a scripting project. Specify this file to be either the application scripting project (via the Scripting tab of the Options dialog) or the active application project's scripting project (via the application project's Script Settings dialog (**Project | Script Settings**)). The macros in the scripting project will now appear in the *Macros* pane of the Macros tab (*see screenshot below*).

To create an application command for a macro, select the macro in the *Macros* pane, set the text of the command in the *Display Text* text box, and click **Add Command** (*see screenshot below*). A command associated with the selected macro will be added to the *Associated Commands* list box.



To edit the icon of an associated command, select the command and click **Edit Icon**. To delete an associated command, click **Remove**.

Placing a macro-associated command in a menu or toolbar

There are two ways to place a macro-associated command in a menu or toolbar:

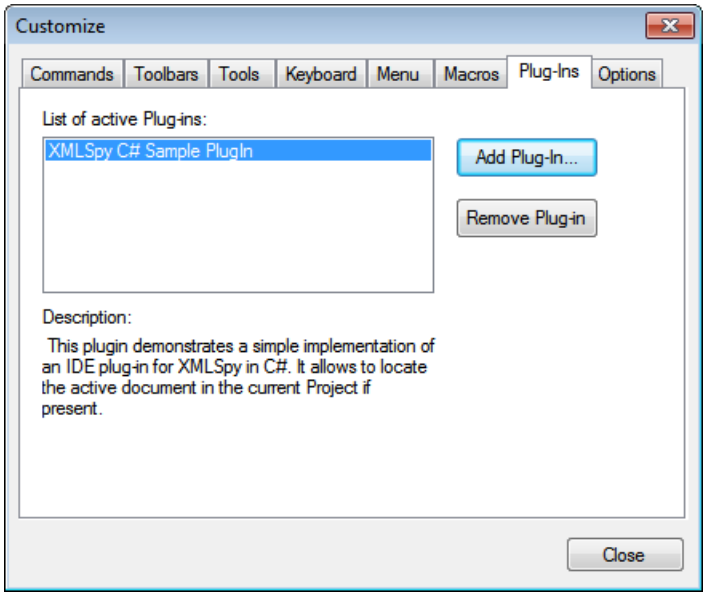
- Drag the command from the Associated Commands list box to the desired location in the menu or toolbar.
- Use the mechanisms available in the Commands tab of the Customize dialog.

In either case, the command will be created at the desired location. Clicking on the command in the menu or toolbar will execute the macro.

Note: If a macro has been set as an associated command, you can set a keyboard shortcut for it. In the Keyboard tab of the Customize dialog, select *Macros* in the *Category* combo box, then select the required macro, and set the shortcut. You must set a macro as an associated command in order for it to be available to be created as a keyboard shortcut.

Plug-Ins

The **Plug-Ins** tab allows you to integrate plug-ins and to place commands, where these have been so programmed, in an application menu and/or toolbar. In the Plug-In tab (*screenshot below*), click **Add Plug-In**, and browse for the plug-in's DLL file (*see 'Creating plug-ins' below*). Click **OK** to add the plug-in. Multiple plug-ins can be added.



After a plug-in has been added successfully, a description of the plug-in appears in the dialog and the **Remove Plug-In** button becomes enabled. If the plug-in code creates toolbars and menus, these will be immediately visible in the application interface. To remove a plug-in select it and click **Remove Plug-In**.

Creating plug-ins

Source code for sample plug-ins has been provided in the application's (My) Documents folder: `Examples\IDEPlugin` folder. To build a plug-in from such source code, do the following:

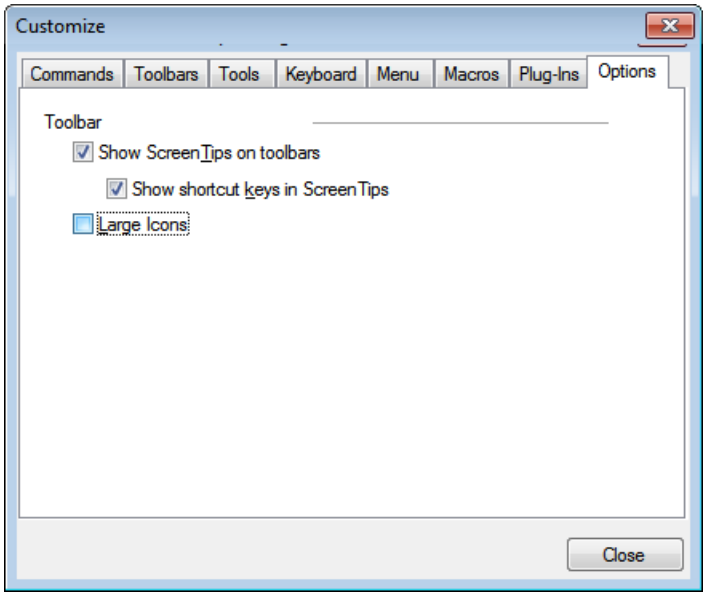
1. Open the solution you want to build as a plug-in in Visual Studio.

2. Build the plug-in with the command in the Build menu.
3. The plug-in's DLL file that will be created in the `Bin` or `Debug` folder. This DLL file is the file that must be added as a plug-in (*see above*).

For more information about plug-ins, see the section IDE Plugins.

Options

The **Options** tab allows you to define general environment settings.

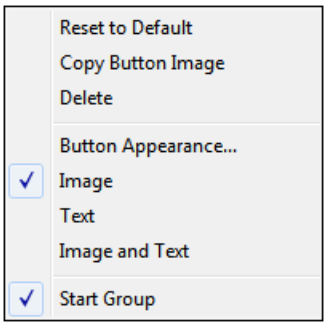


Click the check boxes to toggle on the following options:

- *Show ScreenTips on toolbar:* Displays a popup when the mouse pointer is placed over an icon in any toolbar. The popup contains a short description of the icon function, as well as the associated keyboard shortcut, if one has been assigned and if the *Show shortcut keys* option has been checked .
- *Show shortcut keys in Screen Tips:* Defines whether shortcut information will be shown in screen tips.
- *Large icons:* Toggles the size of toolbar icons between standard and large.

Customize Context Menu

The **Customize context menu** (*screenshot below*) is the menu that appears when you have the Customize dialog open and then right-click an application menu, menu command, or toolbar icon.

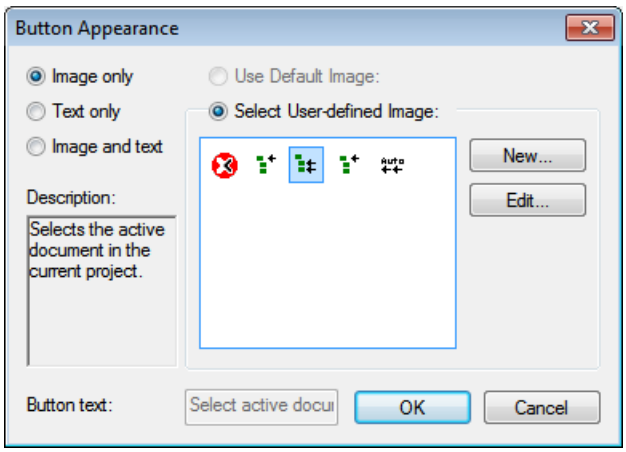


The following functionality is available:

- *Reset to Default:* Currently no function.
- *Copy Button Image:* Copies the icon you right-click to the clipboard.
- *Delete:* Deletes the selected menu, menu command, or toolbar icon. For information about how to restore deleted items, see below.
- *Button Appearance:* Pops up the Button Appearance dialog (*see screenshot below*), in which you can set properties that define the appearance of the selected toolbar icon. See the description below for details.
- *Image, Text, Image and Text*: Mutually exclusive options that determine whether the selected toolbar icon will be an icon only, text only, or both icon and text. You can select one of these options to make the change. Alternatively, you can make this change in the Button Appearance dialog.
- *Start Group:* Inserts a vertical group-divider to the left of the selected toolbar icon. This makes the selected toolbar icon the first of a group of icons.

The Button Appearance dialog

Right-click a toolbar icon and click **Button Appearance** to get the Button Appearance dialog (*screenshot below*). Via this dialog you can edit the toolbar icon image, as well as its text. Currently only toolbar icons for macros and from plug-ins can be edited using this dialog.

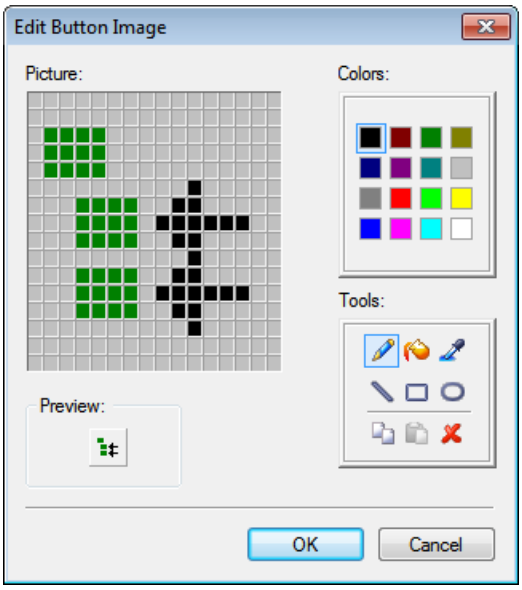


The following editing functionality is available for the selected toolbar icon (the one that was right-clicked to get the Customize context menu):

- *Image only, Text only, Image and text*: Select the desired radio button to specify what

form the toolbar icon will take.

- *Image editing:* When *Image only* or *Image and text* is selected, then the image editing options are enabled. Click **New** to create a new image that will be added to the user-defined images in the images pane. Select an image and click **Edit** to edit it.



- *Image selection:* Select an image from the Images pane and click OK to use the selected image as the toolbar icon.
- *Text editing and selection:* When *Text only* or *Image and text* is selected, then the *Button Text* text box is enabled. Enter or edit the text and click **OK** to make this the text of the toolbar icon.

Note: The Button Appearance dialog can also be used to edit the text of menu commands. Right-click the menu command (with the Customize dialog open), click **Button Appearance**, and then edit the menu command text in the *Button Text* text box.

Restoring deleted menus, menu commands, and toolbar icons

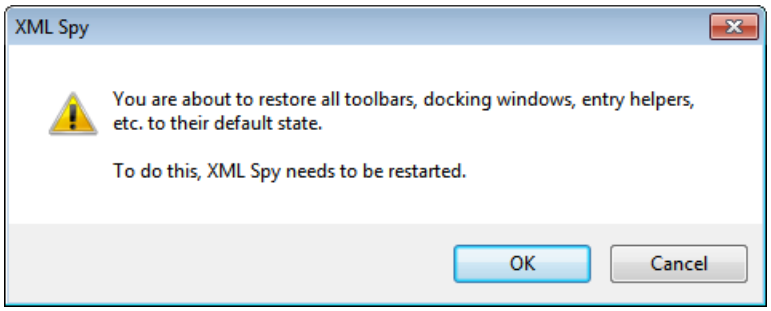
If a menu, menu command, or toolbar icon has been deleted by using the **Delete** command in the Customize context menu, these can be restored as follows:

- *Menus:* Go to **Tools | Customize | Menu**, and click the **Reset** button in the *Application Frame Menus* pane. Alternatively, go to **Tools | Customize | Toolbars**, select Menu Bar, and click the **Reset** button.
- *Menu commands:* Go to **Tools | Customize | Commands**, and drag the command from the Commands list box into the menu.
- *Toolbar icons:* Go to **Tools | Customize | Commands**, and drag the command from the Commands list box into the toolbar.

26.16.12 Restore Toolbars and Windows

The **Restore Toolbars and Windows** command closes down XMLSpy and re-starts it with the default settings. Before it closes down a dialog pops up asking for confirmation about whether

XMLSpy should be closed (*screenshot below*).



This command is useful if you have been resizing, moving, or hiding toolbars or windows, and would now like to have all the toolbars and windows as they originally were.

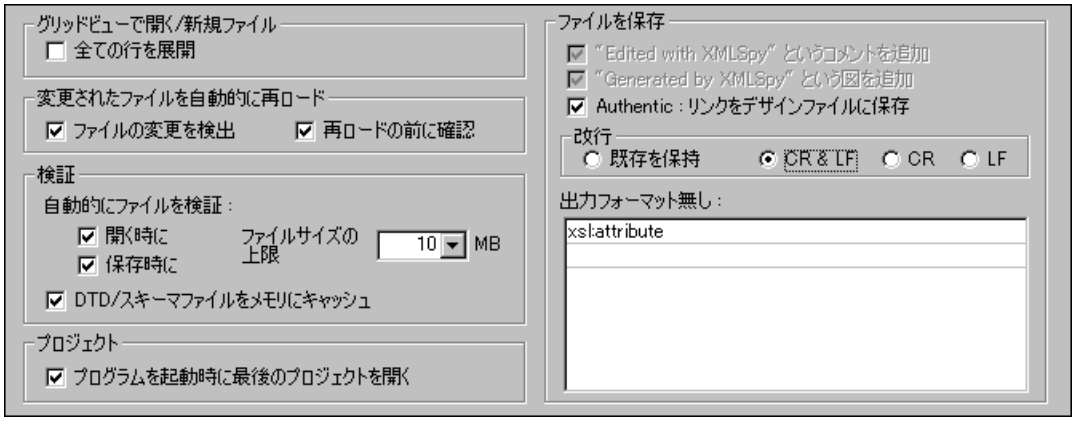
26.16.13 Options

The **Tools** | **Options** command enables you to define global application settings. These settings are specified in a tabbed dialog box and saved in the registry. They apply to all current and future document windows. The **Apply** button in the Options dialog displays the changes in the currently open documents and fixes the current settings. The changes are seen immediately in the background windows.

Each tab of the Options dialog is described in detail in this section.

File

The **File** tab defines the way XMLSpy opens and saves documents. Related settings are in the Encoding tab.



Open/New file in Grid view
You can choose to open an existing file or create a new file either in Grid View or in Text View. If you select Grid View, you can also choose to automatically expand all lines.

Automatic reload of changed files

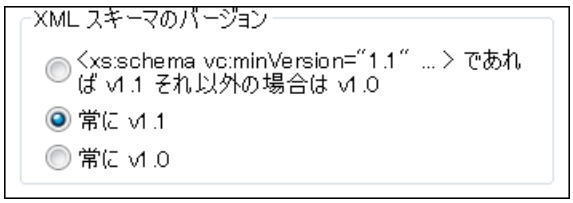
If you are working in a multi-user environment, or if you are working on files that are dynamically generated on a server, you can watch for changes to files that are currently open in the interface. Each time XMLSpy detects a change in an open document, it will prompt you about whether you want to reload the changed file.

Validation

If you are using DTDs or schemas to define the structure of your XML documents, you can automatically check the document for validity whenever it is opened or saved. During Open and Save operations, you have the option of validating files only if the file-size is less than a size you specify in MB. If the document is not valid, an error message will be displayed. If it is valid, no message will be displayed and the operation will proceed without any notification. XMLSpy can also cache these files in memory to save any unnecessary reloading (e.g. when the schema being referred to is accessed through a URL). If your schema location declaration uses an URL, disable the "cache DTD/Schema files in memory" option to have changes made to the schema appear immediately, and not use the cached version of the schema.

XML Schema Version

The XSD mode that is enabled in Schema View depends on both (i) the presence/absence—and, if present, the value—of the `/xs:schema/@vc:minVersion` attribute of the XSD document, and (ii) the XML Schema Version option selected in the File tab of the Options dialog (**Tools | Options**, *screenshot below*).



The following situations are possible. *XML Schema Version* in the table below refers to the selection in the XML Schema Version pane shown above. The `vc:minVersion` values in the table refer to the value of the `xs:schema/@vc:minVersion` attribute in the XML Schema document. For more details, see the section Editing Views | Schema View | XSDMode.

XML Schema Version	`vc:minVersion` attribute	XSD mode
Always v1.0	Is absent, or is present with any value	1.0
Always v1.1	Is absent, or is present with any value	1.1
Value of @vc:minVersion	Attribute has value of `1.1`	1.1
Value of @vc:minVersion	Attribute is absent, or attribute is present with a value other than `1.1`	1.0

Project
When you start XMLSpy, you can open the last-used project automatically.

Save File
When saving an XML document, XMLSpy includes a short comment `<!-- Edited with XMLSpy http://www.altova.com -->` near the top of the file. This option can only be deactivated by licensed users, and takes effect when editing or saving files in the Enhanced Grid or Schema Design View.

When saving a content model diagram (using the menu option **Schema design | Generate Documentation**), XMLSpy includes the XMLSpy logo. This option can only be deactivated by licensed users.

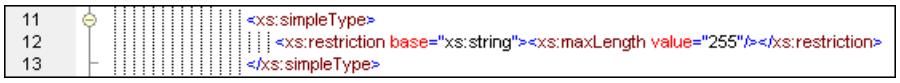
If a StyleVision Power Stylesheet is associated with an XML file, the 'Authentic: save link to design file' option will cause the link to the StyleVision Power Stylesheet to be saved with the XML file.

Line breaks
When you open a file, the character coding for line breaks in it are preserved if **Preserve old** is selected. Alternatively, you can choose to code line breaks in any of three codings: **CR&LF** (for PC), **CR** (for MacOS), or **LF** (for Unix).

No output formatting for
In Text View, the indentation of an element can be made to reflect its position in the element hierarchy (*see* **Save File**). You can, however, override this indentation for individual elements. To do this, enter the element name in the **No output formatting for** field. All elements entered in this field will be formatted such that their descendant elements have no whitespace between them (*see screenshots*).

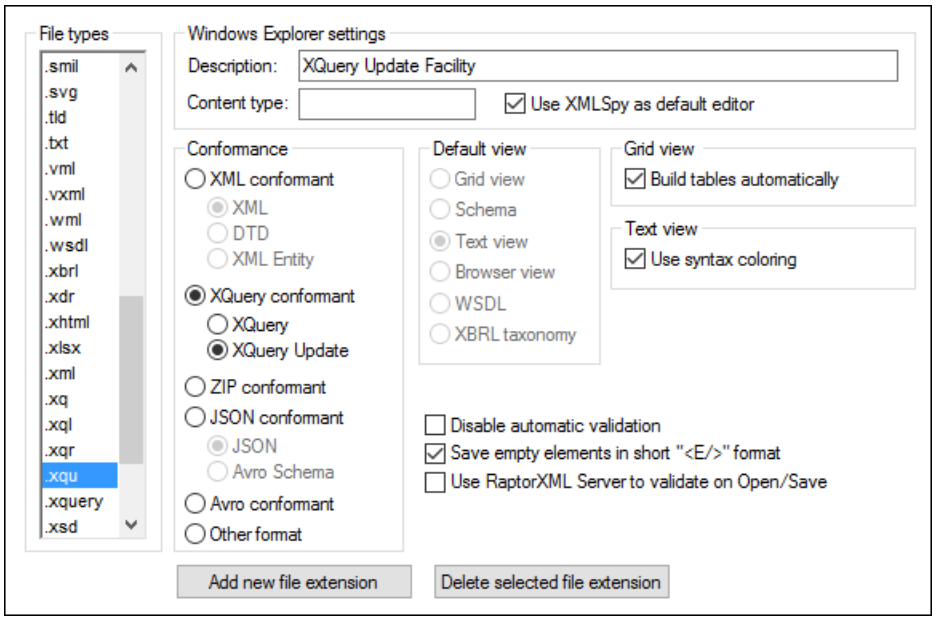
Hierarchical indentation for all elements:

No output formatting has been specified for element `xs:restriction`:



Save and exit
After making the settings, click **OK** to finish.

File Types
The **File types** tab allows you to customize the behavior of XMLSpy on a per-file-type basis.



Choose a file type from the File Types list box, and then customize the functions for that particular file type as described below.

Windows Explorer settings
You can define the file type description and MIME-compliant content type used by Windows Explorer and whether XMLSpy is to be the default editor for documents of this file type.

Conformance
XMLSpy provides specific intelligent editing features, as well as other features, for different file types. XMLSpy sets the features for a particular file type on the basis of the conformance you set in this option. For example, in the screenshot above, files with the `.xqu` file extension are set to be conformant to XQuery Update. XMLSpy will therefore open `.xqu` files with XQuery Update editing support. XMLSpy lets you set the following conformance options: XML, XQuery, ZIP, JSON, Avro, other formats. XML conformance is further differentiated between XML, DTD, and XML Entity file types. JSON conformant files are differentiated according to whether they are plain JSON or Avro Schema. The *Avro conformant* option invokes support for Avro binary files. A large number of file types are defined with a default conformance that is appropriate for the file type. We recommend that you do not modify these settings unless you are adding a new file type or deliberately wish to set a file type to another kind of conformance.

Default view
This group lets you define the default view to be used for each file type. If a particular conformance can be viewed in one view only, then that view is selected by default and view selection is disabled. For example, XQuery Update documents can only be viewed in Text View, so this view is selected by default and view selection is disabled; similarly, Avro conformant documents (Avro binaries) can be viewed only in Grid View.

Grid View
This check box lets you define whether the Grid View should automatically build tables.

Text View
This check box lets you set syntax-coloring for particular file types.

Disable automatic validation
This option enables you to disable automatic validation per file type. Automatic validation typically takes place when a file is opened or saved, or when a view is changed.

Save empty elements in short <E/> format
Some applications that use XML documents or output generated from XML documents may have problems understanding the short `<Element/>` form for empty elements defined in the XML 1.0 Specification. You can instruct XMLSpy to save elements in the longer (but also valid) `<Element></Element>` form.

Use RaptorXML Server to validate on Open/Save
Specifies whether RaptorXML Server should be used to validate files of the selected file type when the file is opened and saved. For this to work, a RaptorXML Server must be set up and configured.

Add new file extension
Adds a new file type to the File types list. You must then define the settings for this new file type using the other options in this tab.

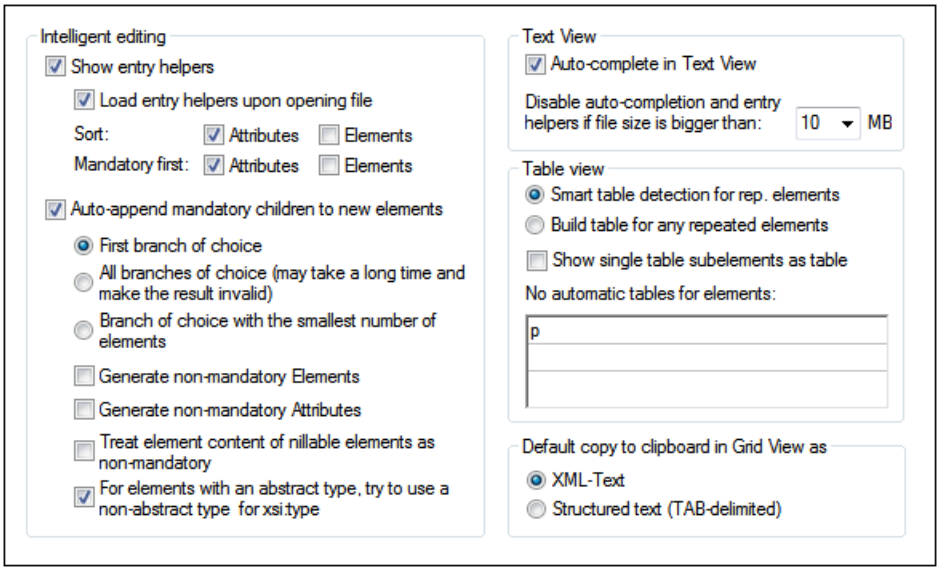
Delete selected file extension
Deletes the currently selected file type and all its associated settings.

Save and exit

After making the settings, click **OK** to finish.

Editing

The **Editing** tab enables you to specify editing behaviour in XMLSpy.



Intelligent editing

While editing documents, XMLSpy provides intelligent editing based on these settings. You can also customize various aspects of the Entry Helper behavior in this pane. The customization settings made here will be applied when relevant to the file type being edited. For example, the option to load entry helpers on opening the file and sorting attributes will not be applicable to DTD or XQuery documents.

Mandatory child elements of `choice` groups are auto-appended on the basis of the setting made in this pane. You can select whether (i) the first branch (element) of the `choice` group, (ii) all branches, or (iii) the branch with the smallest number of descendant elements is generated. Note that the *All branches* selection could generate an invalid document since only one branch from a `choice` group is allowed.

Text View

The *Auto-complete* option automatically adds unambiguous structural components. For example, when the closing angular bracket of the start tag of an element is entered, then the end tag of that element is automatically added if this option is enabled.

In Text View, Auto-completion and entry helpers can be disabled if a file is bigger than the size

specified in the *Disable Auto-completion...* combo box. This is useful if you wish to speed up the editing of large files and can do without the auto-completion feature and entry helpers. If the file size is bigger than that specified for this option, then the Text View context menu contains a toggle command for switching on and off Auto-completion and entry helper use. So you can always switch these editing aids on and off at any time during editing (in the event of files having a size greater than the size specified for this option). If the value specified for this option is smaller than the size of the opened file, locations indicated in error messages will not correctly correspond to the location in Text View.

Default copy to clipboard in grid view as

You can choose the format in which data will be exported to foreign applications using the clipboard. If you select XML-Text, the contents of the clipboard will be formatted and tagged just like the resulting XML file itself. The structured text mode attempts to format the clipboard contents as a table, for use in a spreadsheet or database application. This option does not affect the internal clipboard format that XMLSpy uses for copying and pasting.

Table view

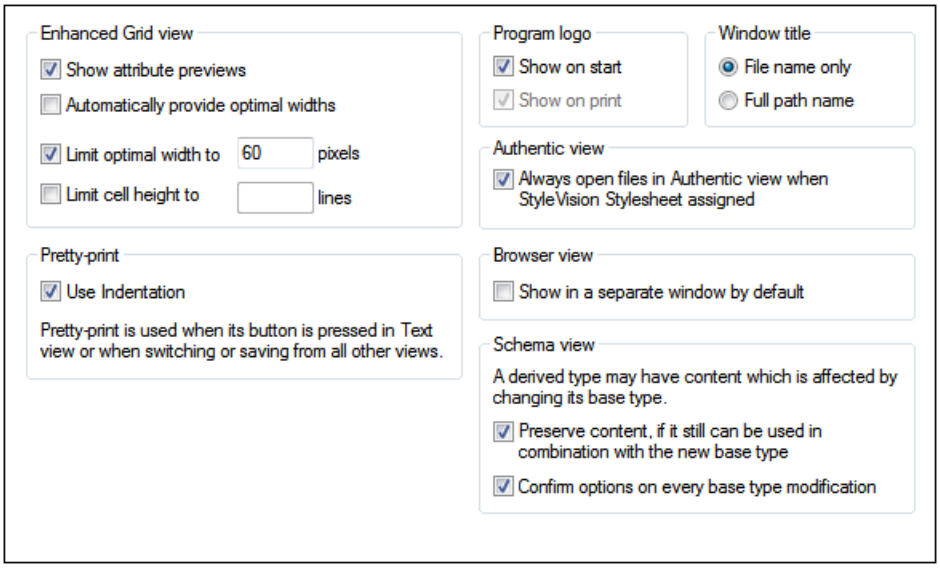
You can also control, how XMLSpy decides when to display repeating elements in the Table View.

Save and exit

After making the settings, click **OK** to finish.

View

The **View** tab enables you to customize the XML documents presentation in XMLSpy.



Grid View

XML elements in Grid View can be collapsed into a single line displaying the element name. When collapsed, the element's attributes can also be displayed in that line. If the *Show Attribute Previews* option is checked, attributes are displayed in gray with collapsed elements. Otherwise, attributes are not displayed with the collapsed element. Columns in the grid can be set to adjust automatically to optimal widths. Additionally, the maximum optimal width and cell height can be limited. If the content of a cell is more than can fit in a cell, this is indicated by an ellipsis.

Pretty-print

When you select **Edit | Pretty-Print XML Text** in Text View or switch from another view to Text View, the XML document will be "pretty-printed". The pretty-printing will be with or without indentation according to whether the *Use Indentation* option in this dialog is checked or not. The amount of indentation can be specified in the Tabs pane of the Text View Settings dialog.

Program logo

You can turn off the splash screen on program startup to speed up the application. Also, if you have a purchased license (as opposed to, say, a trial license), you will have the option of turning off the program logo, copyright notice, and registration details when printing a document from XMLSpy.

Window title

The window title for each document window can contain either the file name only or the full path name.

Authentic View

XML files based on a **StyleVision Power Stylesheet** are automatically opened in the Authentic View when this option is active.

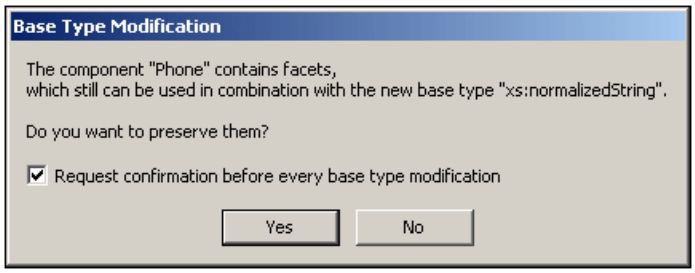
Browser View

You can choose to see the browser view in a separate window, enabling side-by-side placement of the edit and browser views.

Schema view

An XML Schema datatype can be derived from another datatype. For example, a datatype for E-mail elements can be derived from a base datatype of `xs:string` (for example, by restricting the `xs:string` datatype to a specific set of characters). If the base datatype is subsequently changed, you can set the following options:

- *Preserve content:* If the definitions used to define the derived type can be used with the new base type, checking this option will automatically preserve the definitions.
- *Confirm on every modification*: After changing the base type, a dialog (*see screenshot below*) will pop up asking whether the old definitions should be preserved and used with the new base type.

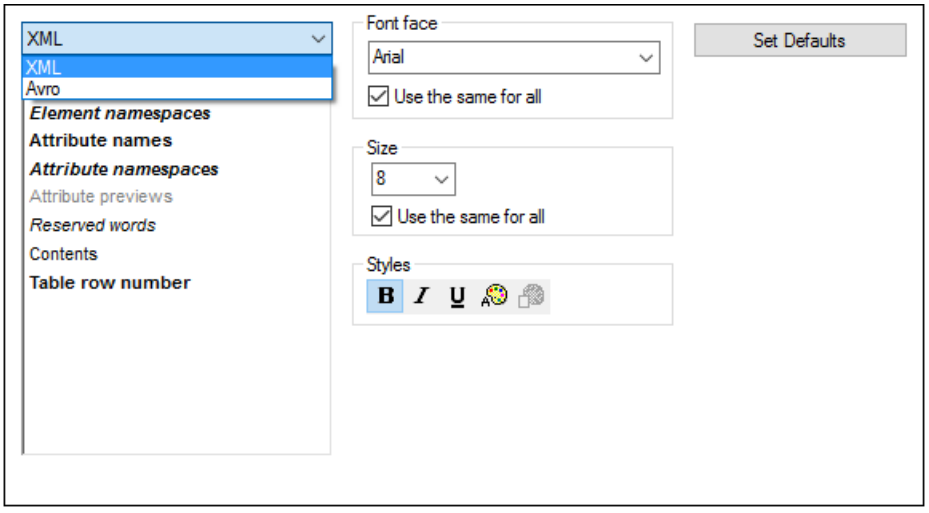


Save and exit

After making the settings, click **OK** to finish.

Grid Fonts

The **Grid fonts** tab allows you to customize the appearance of text in Grid View and Avro View. In the combo box (*see screenshot below*), select the document type for which you want to configure Grid View and then select the options for that document type.



Font face

You can select the font face and size to be used for displaying the various items in Grid View. The same fonts are also used for printing, so only TrueType fonts should be selected.

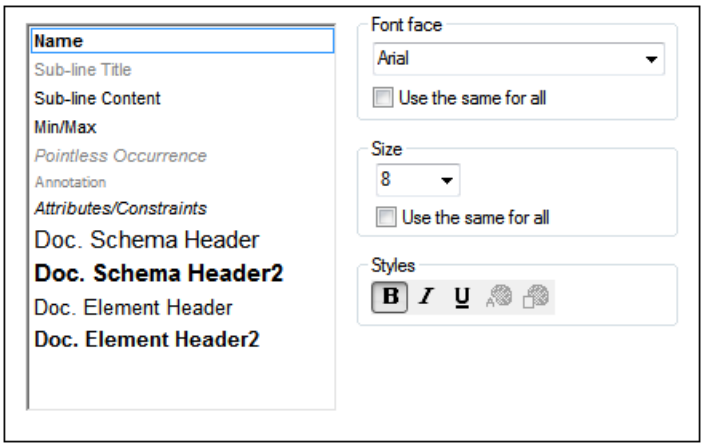
Size
Select the required size. If you want to use the same font size for all items, check the **Use the same for all** check box.

Styles
The style and color can be set using the options in this pane. The current settings are immediately reflected in the list in the left-hand pane, so you can preview the way your document will look.

Set Defaults
The **Set Defaults** button resets fonts to the original installation settings.

Save and exit
After making the settings, click **OK** to finish.

Schema Fonts
The **Schema fonts** tab enables you to customize the appearance of text in Schema View.



Font face
You can select the font face and size to be used for displaying the various items in the Schema Design view. The same fonts are used when printing and creating schema documentation, so only TrueType fonts should be selected. Components prefixed with "Doc." are used in the schema documentation.

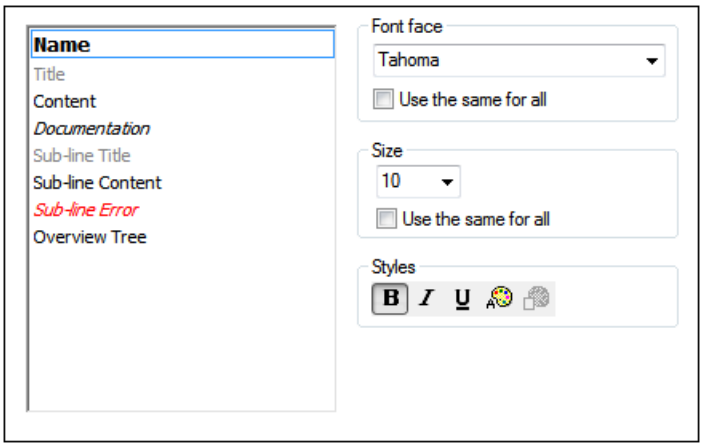
Size
Select the required size. If you want to use the same font size for all items, click on the Use The Same For All" check box.

Styles
The style and color can be set using the options in this pane. The current settings are immediately reflected in the list in the left pane, so you can preview the way your document will look.

Set Defaults
The **Set Defaults** button resets fonts to the original installation settings.

Save and exit
After making the settings, click **OK** to finish.

WSDL Fonts
The **WSDL fonts** tab you to customize the appearance of text in WSDL View.



Font face
You can select the font face and size to be used for displaying the various items in the WSDL Design view. The same fonts are used when printing and creating documentation, so only TrueType fonts should be selected.

Size
Select the required size. If you want to use the same font size for all items, click on the Use The

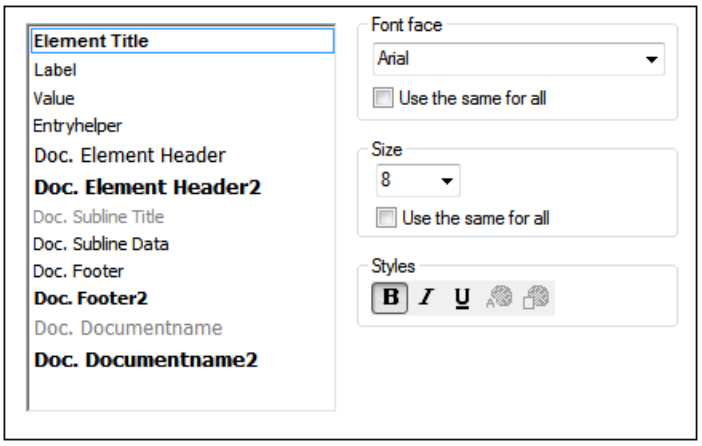
Same For All" check box.

Styles
The style and color can be set using the options in this pane. The current settings are immediately reflected in the list in the left pane, so you can preview the way your document will look.

Set Defaults
The **Set Defaults** button resets fonts to the original installation settings.

Save and exit
After making the settings, click **OK** to finish.

XBRL Fonts
The **XBRL fonts** tab you to customize the appearance of text in XBRL View.



Font face
You can select the font face and size to be used for displaying the various items in XBRL View. The same fonts are used when printing and creating documentation, so only TrueType fonts should be selected.

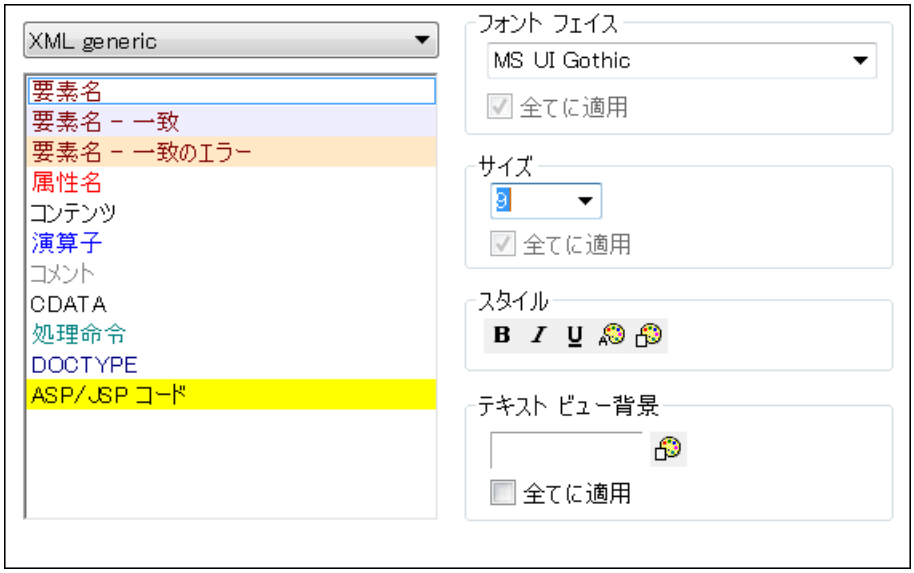
Size
Select the required size. If you want to use the same font size for all items, click on the *Use The Same For All* check box.

Styles
The style and color can be set using the options in this pane. The current settings are immediately reflected in the list in the left pane, so you can preview the way your document will look.

Set Defaults
The **Set Defaults** button resets fonts to the original installation settings.

Save and exit
After making the settings, click **OK** to finish.

Text Fonts
The **Text fonts** tab enables you to customize the appearance of text in Text View. You can customize the appearance of of text items according to the type of text item. For example, you can color element names and attribute names differently.



Customizing by file type
The text item types are categorized into the following groups:

- XML generic
- XQuery
- CSS
- JSON

- Miscellaneous
- Output

To customize text fonts, do the following:

1. In the combo box at top left, select the type of document for which you wish to customize text fonts. On doing this, the text item types for that document type appear in the box below the combo box. (*In the screenshot above,* `XML generic` *has been selected as the document type.*)
2. Select the text item type you wish to customize by clicking it. (*In the screenshot above,* `Element names` *has been selected.*)
3. Set the font properties using the options in the panes on the right-hand side. You can select the font-family, font-size, font-style, font-color, and background-color for the text. Additionally, you can also select a background color for the entire Text View.

Note the following points:

- The same font, style, and size is used for all text item types. Only the text color and background color can be changed for individual text types. This enables the syntax coloring feature.
- In the *Generic XML* category, the *Element names* text type consists of three subtypes: (i) *Element names* applies to element names that are not selected; (ii) *Element names - match* applies to those element names that are selected (names in which the cursor is placed) and where the start tag name matches the end tag name; (i) *Element names - match error* applies to those element names that are selected, but where the start tag name matches the end tag name. Element names that are being edited will therefore be highlighted with different background colors according to to whether the start tag names match the end tag names or not. These highlight colors can be changed by changing the respective background colors. Highlighting is turned on by default, and can be turned off by deselecting the *Highlight elements* option in the Text View Settings dialog.
- In the *Miscellaneous* category: (i) *Selection* refers to the currently selected text content; *inactive selection* refers to other occurrences in the document of the same text content; (ii) *Find active marker* refers to the currently selected occurrence of a search result, whereas *Find marker* refers to other (inactive) occurrences of the search result; (iii) *Back-mapping active* refers to the currently selected content in the back-mapping document-trio (**`XML–XSLT/XQuery–Result`**); *Back-mapping inactive* refers to back-mapped content in the other two documents.

Set defaults

The **Set Defaults** button resets fonts to the original installation settings.

Save and exit

After making the settings, click **OK** to finish.

Colors

The **Colors** tab enables you to customize the background colors used in the Table View of Grid View. In the screenshot below, the colors have been changed from the default colors by clicking

the palette icon next to each item and then selecting the preferred color.

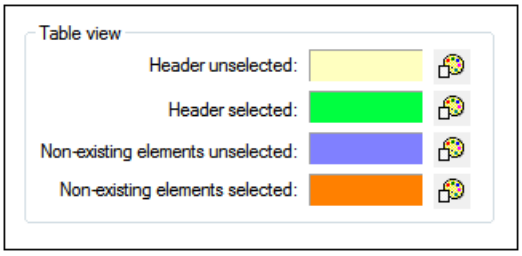
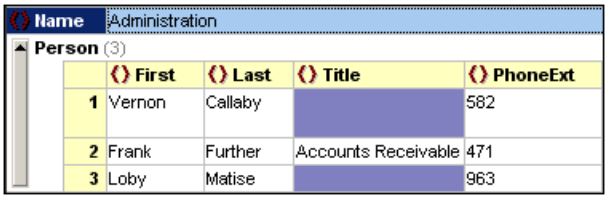
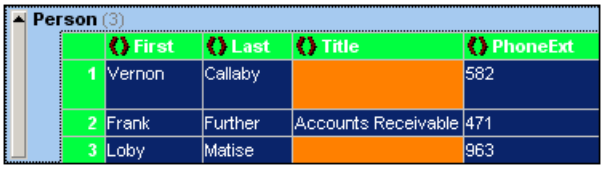


Table View

The Header unselected and Header selected options refer to the column and row headers. The screenshot below shows headers unselected; its color is as set in the dialog above.

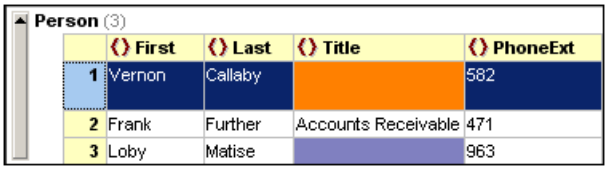


The Header Selected color is activated when all headers are selected (*screenshot below*)—not when individual headers are selected. The screenshot below shows this using the colors defined in the dialog shown above. All headers can be selected by clicking the cell that intersects both headers or by selecting the element created as the table—or any of its ancestors.



Non-existent Elements

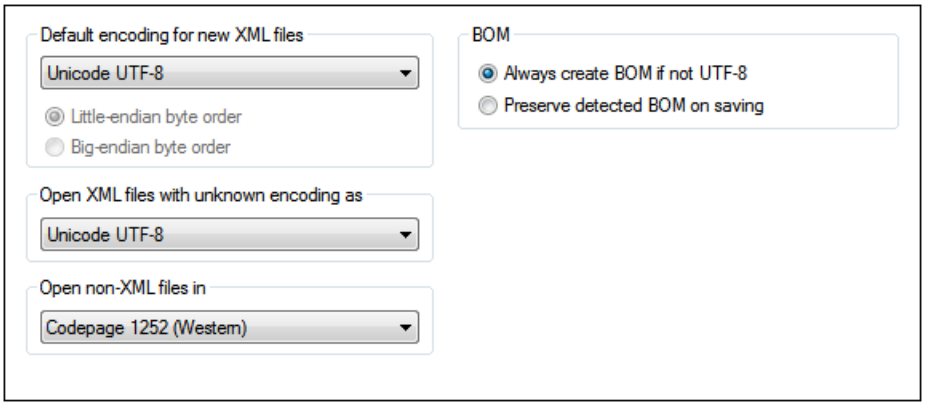
When an element or attribute does not exist in the XML document, then it can be given different background colors when selected and unselected. This is shown in the screenshot below, in which the first row is selected.



Note: In addition to the colors you define here, XMLSpy uses the regular selection and menu color preferences set in the Display Settings in the Control Panel of your Windows installation.

Save and exit
After making the settings, click **OK** to finish.

Encoding
The **Encoding** tab specifies options for file encodings.



Default encoding for new XML files
The default encoding for new XML files can be set by selecting an option from the dropdown list. A new document is created with an XML declaration containing the encoding value you specify here. If a two- or four-byte encoding is selected as the default encoding (i.e. UTF-16, UCS-2, or UCS-4) you can also choose between little-endian and big-endian byte-ordering.

The encoding of existing XML files will be retained and can only be changed with the **File | Encoding** command.

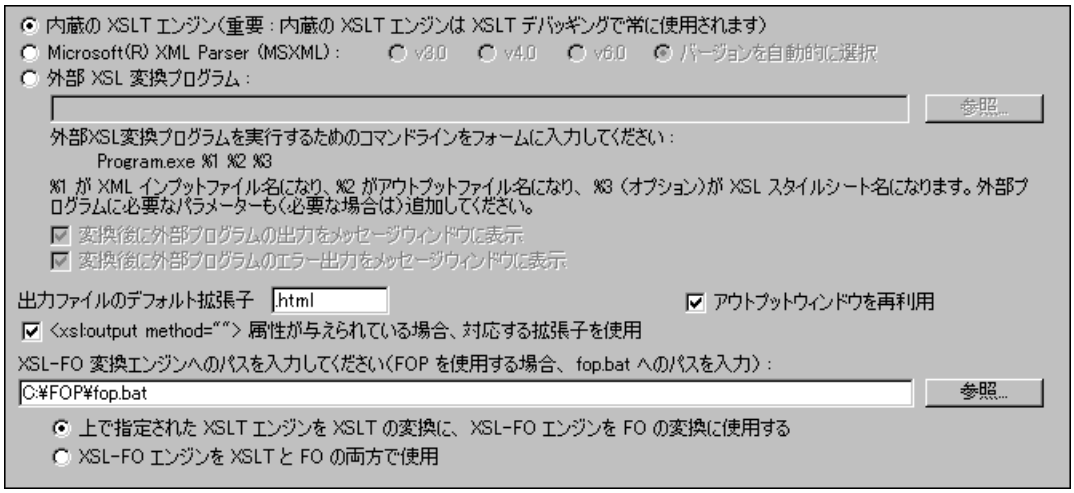
Open XML files with unknown encoding as
If the encoding of an XML file cannot be determined or if the XML document has no encoding specification, the file will be opened with the encoding you select in this combo box.

Open non-XML files in
Existing and new non-XML files are opened with the encoding you select in this combo box. You can change the encoding of the document by using the **File | Encoding** command.

BOM (Byte Order Mark)
When a document with two-byte or four-byte character encoding is saved, the document can be saved either with (i) little-endian byte-ordering and a little-endian BOM (*Always create BOM if not UTF-8*); or (ii) the detected byte-ordering and the detected BOM (*Preserve detected BOM on saving*).

Save and exit
After making the settings, click **OK** to finish.

XSL
The **XSL** tab (*screenshot below*) enables you to define options for XSLT transformations and XSL-FO transformations carried out from within the application.



XSLT transformations
XMLSpy contains the Altova RaptorXML XSLT 1.0, XSLT 2.0, and XSLT 3.0 engines, which you can use for XSLT transformations as well as for validating the XML files used in transformations. The appropriate XSLT engine (1.0, 2.0, or 3.0) is used (according to the value of the `version` attribute of the `xsl:stylesheet` or `xsl:transform` element). This applies both for XSLT transformations as well as for XSLT debugging using XMLSpy's XSLT/XQuery Debugger.

For transforming XML documents using XSLT, you could use one of the following:

- The built-in Altova XSLT Engines (XSLT 1.0, XSLT 2.0, and XSLT 3.0).
- The MSXML 3.0, 4.0, or 6.0 parser (which is pre-installed). If you know which version of the MSXML parser is running on your machine, you could select it; otherwise, you should let the application select the version automatically. (The *Choose version automatically* option is active by default.) In this case, the application tries to select the most recent available version.
- An external XSLT processor of your choice. You must specify the command line string that the external XSLT processor uses to run a transformation. The following variables are available for building the command line string:

  `%1` = XML document to process
  `%2` = Output file to generate
  `%3` = XSLT stylesheet to use (if the XML document does not contain a reference to a stylesheet)

For example, say you have a processor that uses the following command to run an XSLT transformation:

```
myxsltengine.exe -o output.xml input.xml stylesheet.xslt parameter-name=parameter-value
```

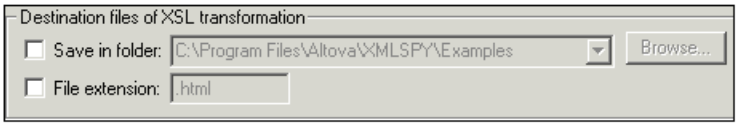
Then, in XMLSpy, you can use the variables listed above to generate this command. Select the *External XSL Transformation Program* radio button, and enter the following line in the text box:

```
c:\myxsltengine\myxsltengine.exe -o %2 %1 %3 parameter-name=parameter-value
```

Check the respective check boxes to show the output and error messags of the external program in the Messages Window in XMLSpy.

Note: The parameters set in XMLSpy's XSLT Input Parameters dialog are passed to the internal Altova XSLT Engines only. They are not passed to any other XSLT Engine that is set up as the default XSLT processor.

The *Reuse output window* option causes subsequent transformations to display the result document in the same output window. If the XML file belongs to a project and *Reuse output window* option is disabled, the setting only takes effect if the *Save in folder* output file path (*screenshot below*) in the relevant project properties is **also** disabled.



XSL-FO transformations

FO documents are processed using an FO processor, and the path to the executable of the FO processor must be specified in the text box for the XSL-FO transformation engine. The transformation is carried out using the **XSL/XQuery | XSL-FO Transformation** menu command. If the source file (the active document when the command is executed in the IDE) is an XSL-FO document, the FO processor is invoked for the transformation. If the source document is an XML document, an XSLT transformation is required to first convert the XML document to an XSL-FO document. This XSLT transformation can be carried out either by the XSLT engine you have specified as the default engine for the application (*see above*), or by the XSLT engine that might be built into the FO processor you have specified as the default FO processor for the application. To select between these two options, click the appropriate radio button.

After making the settings, click **OK** to finish.

Note: Unless you deselected the option to install the FOP processor of the Apache XML Project, it will have been installed in the folder `C:\ProgramData\Altova\SharedBetweenVersions`. If installed, the path to it will automatically have been entered in the XSL-FO Engine input box. You can set the path to any FO processor you wish to use. Note, however, that the same path will be used by other Altova products that use FO processors and have settings to select the FO processor (StyleVision and Authentic Desktop).

Save and exit
After making the settings, click **OK** to finish.

XSL Speed Optimizer

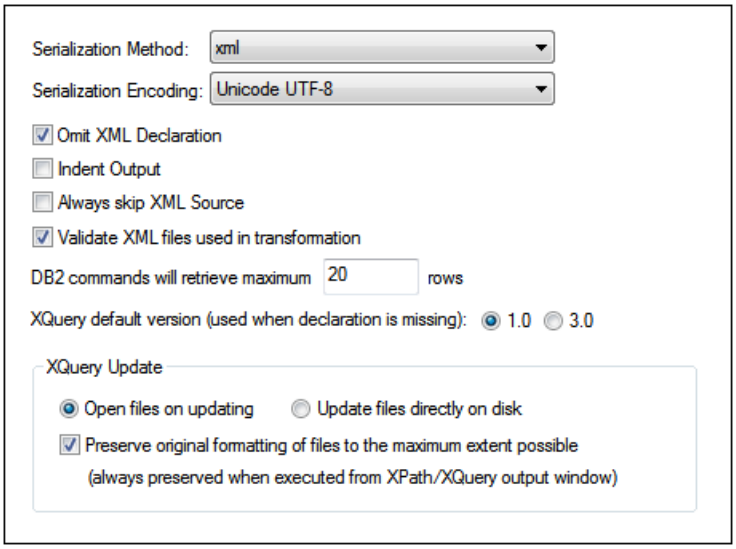
The **XSL Speed Optimizer** tab (*screenshot below*) enables you to define options for XSL Speed Optimizer.

A time threshold for single XSLT instructions in an XSLT stylesheet can be specified for the Optimizer. Values range from 0.1% of total transformation time to 99% of total time. If an instruction takes more time to execute than that specified as the threshold, then optimization analysis is invoked. Otherwise no analysis is carried out. If optimization analysis is unsuccessful, the reason might be that the time threshold is too high. Consider lowering it.

Save and exit
After making the settings, click **OK** to finish.

XQuery

The **XQuery** tab (*screenshot below*) defines options related to the editing and execution of XQuery and XQuery Update documents.



General XQuery options

The following options are available:

- *Serialization:* You can choose the serialization method (XML, XHTML, HTML, text) and serialization encoding. Serialization refers to the output document produced when an XQuery document is executed.
- *Omit XML declaration:* Omits the XML declaration in the serialized (output) document.
- *Indent output:* Indents the output document to show the document hierarchy.
- *Always skip XML source:* When an XQuery document is executed, XMLSpy can prompt for an XML source on which to execute the XQuery document. The prompt is a dialog that enables you to browse for the XML file. Select this option to skip this dialog and directly execute the XQuery document. If this option is selected, then the XQuery document should be able to execute correctly without being passed an XML document. This could be either because no XML document is required, or because XML data is accessed via functions within the XQuery document.
- *Validate XML files:* Validates XML files that are used in the execution of XQuery documents. Invalid XML files are flagged, and the XQuery document is not processed.
- *DB2 row retrieval:* In displays that show DB data, you can specify the maximum number of rows to be retrieved. XMLSpy recognizes `.xqr` file extensions as XQuery-for-DB files.
- *XQuery default version:* Specifies the XQuery engine version to use for execution of XQuery documents that do not have a `version` keyword. This applies to both XQuery and XQuery Update documents, and selects the default XQUery Engine to use.

XQuery Update options

The following XQuery Update options are available:

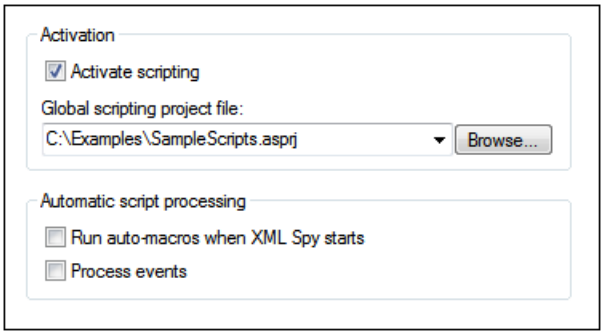
- *Updating:* When an XQuery Update file is executed, target XML files can either be updated directly on disk, or be opened in XMLSpy and updated in memory. The *Open Files on Updating* option enables you to review the updates and save the file to disk or reject the updates (by closing the file without saving).
- *Preserve original formatting:* Preserves the original formatting of the updated document as much as possible.

Save and exit

After making the settings, click **OK** to finish.

Scripting

The **Scripting** tab (*screenshot below*) allows you to enable the Scripting Environment on application startup. Check the *Activate Scripting* check box to do this. You can then specify the Global Scripting Project file (*see screenshot below*).



To set a global scripting project for XMLSpy, check the *Activate Scripting* check box and then browse for the Altova Scripting Project (`.asprj`) file you want. You can also specify: (i) whether Auto-Macros in the scripting project should be automatically executed when XMLSpy starts, and (ii) whether application event handler scripts in the project should be automatically executed or not; check or uncheck the respective check boxes accordingly.

Save and exit

After making the settings, click **OK** to finish. Macros in the Global Scripting Project will then be displayed in the submenu of the **Macros** command.

Taxonomy Packages

An XBRL Taxonomy Package is a zipped archive that contains an offline copy of a taxonomy. The package contains a catalog XML file that maps URIs to the taxonomy's file locations, and so makes the taxonomy available offline to applications. The rules that specify how taxonomy packages are to be structured and built are laid out in the Taxonomy Packages Recommendation of XBRL.org.

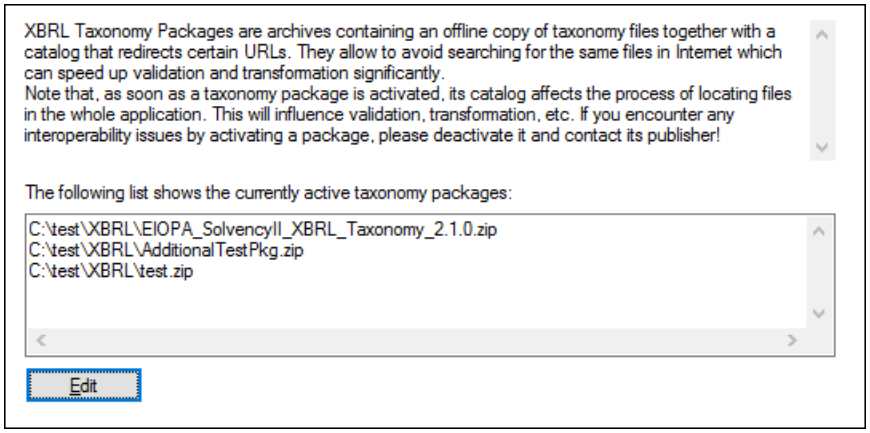
After you have downloaded a taxonomy package, you can set up XMLSpy to automatically identify and use the entry point catalog file of the package. Do this by adding the package to the list of active taxonomy packages. The catalog files of active packages will then be used to locate resources for operations such as XBRL validation.

Note: A resource pointed to by an active package's catalog file will be used for all XMLSpy operations that require that resource. If such a resource is different in some way than the resource that was previously used by XMLSpy, then errors might result when operations are run. For more information, see the caution at the bottom of this topic.

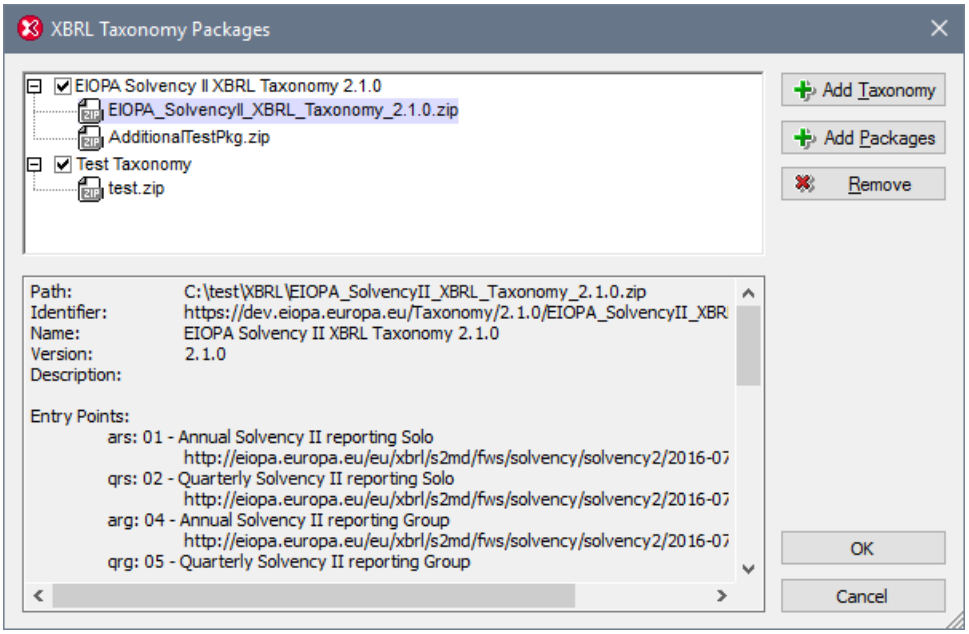
Adding and managing taxonomy packages

To add a taxonomy package, do the following:

1. Select **Tools | Options | Taxonomy Packages** to display the Taxonomy Packages pane (*screenshot below*).



2. Click **Edit** (*see screenshot above*) to display the XBRL Taxonomy Packages dialog (*screenshot below*).



3. Click **Add Taxonomy**, then browse to the location of the taxonomy package, select it, and click **Open**. (You can also select multiple packages to add at one time.) The package will be added to the taxonomy package list in the dialog. The list is displayed as a tree of two levels. The first level indicates the taxonomy; the second level shows the packages of that taxonomy. The check box to the left of a taxonomy entry indicates whether that taxonomy is active or not. A newly added taxonomy will be active by default.
4. Click **OK** to finish. The newly added packages will be displayed in the Taxonomy Packages pane of the Options dialog (*first screenshot above*).

Note the following points:

- If you wish to add an additional package to a taxonomy, do this: Select the taxonomy in the XBRL Taxonomy Packages dialog (*screenshot above*), then add the additional package/s via the **Add Packages** button. The added package/s will be displayed at the second level of that taxonomy.
- When a taxonomy package is selected in the list in the upper pane of the XBRL Taxonomy

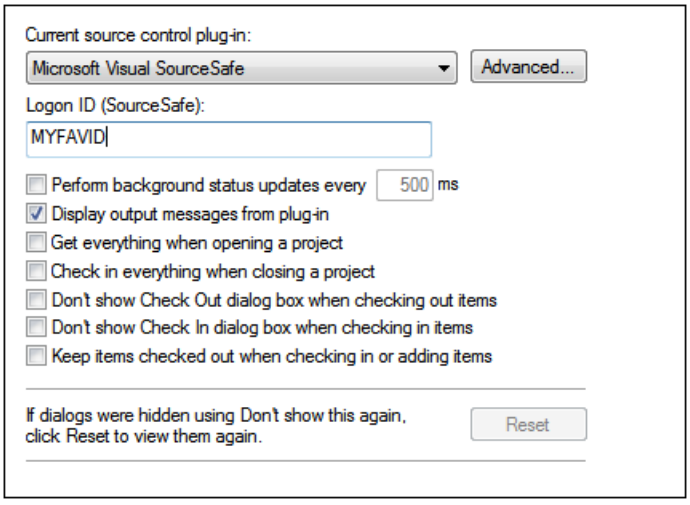
Packages dialog, its details (including its offline location) are displayed in the dialog's lower pane (*see screenshot above*).

- To deactivate a taxonomy, uncheck its check box. If you deactivate a taxonomy, its catalog file/s will not be used. Deactivation is useful if, say, you wish to switch between two versions of a taxonomy.
- You can remove a package by selecting it and clicking **Remove**.
- The following Altova applications support Taxonomy Package Registration: XMLSpy, MapForce, and StyleVision. The taxonomy package list is common to all these applications. If you edit the list in one application, then the modified list will be displayed in the other applications as well. If you edit the package list in one application, and another application is open at the same time, then the other application will display an alert asking whether you wish to reload the package list to reflect the modification.

Caution: Package catalogs might redirect to incompatible resources
A resource pointed to by an active package's catalog file/s will be used for all XMLSpy operations that require that resource. An example of such a resource would be XML Schema, which is used for for both XML validation as well as XBRL validation. If the offline resource located by the package's catalog file is incompatible with your existing environment, then errors might result. In this case, deactivate the taxonomy package and contact the creators of the package with the error information.

Source Control

The **Source Control** tab (*screenshot below*) enables you to specify the source control provider, and the settings and default logon ID for each source control provider.



Source Control Plugin

The current source control plugin can be selected from among the currently installed source control systems. These systems are listed in the dropdown list of the combo box. After selecting the required source control, specify the login ID for it in the next text box. The **Advanced** button pops up a dialog specific to the selected source control plugin, in which you can define settings

for that source control plugin. These settings are different for different source control plugins.

User preferences

A range of user preferences is available, including the following:

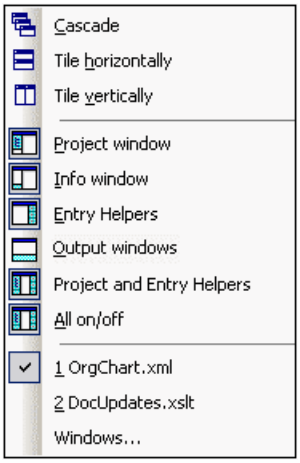
- Status updates can be performed in the background after a user-defined interval of time, or they can be switched off entirely. Very large source control databases could consume considerable CPU and network resources. The system can be speeded up, however, by disabling background status updates or increasing the interval between them..
- When opening and closing projects, files can be automatically checked out and checked in, respectively.
- The display of the Check Out and Check In dialogs can be suppressed.
- The **Reset** button is enabled if you have checked/activated the *Don't show this again* option in one of the dialog boxes. On clicking the **Reset** button, the *Don't show this again* prompt is re-enabled.

Save and exit

After making the settings, click **OK** to finish.

26.17 Window Menu

To organize the individual document windows in an XMLSpy session, the **Window** menu contains standard commands common to most Windows applications.



You can cascade the open document windows, tile them, or arrange document icons once you have minimized them. You can also switch the various Entry Helper windows on or off, or switch to an open document window directly from the menu.

26.17.1 Cascade

This command rearranges all open document windows so that they are all cascaded (i.e. staggered) on top of each other.

26.17.2 Tile Horizontally

This command rearranges all open document windows as **horizontal tiles**, making them all visible at the same time.

26.17.3 Tile Vertically

This command rearranges all open document windows as **vertical tiles**, making them all visible at the same time.

26.17.4 Project Window

This command lets you switch the Project Window on or off.

This is a dockable window. Dragging on its title bar detaches it from its current position and makes it a floating window. Click right on the title bar, to allow docking or hide the window.

26.17.5 Info Window

This command lets you switch the Info Window on or off.

This is a dockable window. Dragging on its title bar detaches it from its current position and makes it a floating window. Click right on the title bar, to allow docking or hide the window.

26.17.6 Entry Helpers

This command lets you switch all three Entry-Helper Windows on or off.

All three Entry helpers are dockable windows. Dragging on a title bar detaches it from its current position and makes it a floating window. Click right on the title bar to allow docking or hide the window.

26.17.7 Output Windows

The Output Windows are a set of tabbed output windows, such as the Messages window (which displays messages like validation results), the Find in Files window, and the XPath window (which shows XPath evaluation results). The initial setting is for them to open at below the Main Window. The Output Windows command lets you switch the Output Windows on or off.

The Output Windows window is dockable. Dragging on its title bar detaches it from its current position and makes it a floating window. Click right on the title bar to allow docking or to hide the window.

For a complete description of Output Windows see Output Windows in the section, Text View.

26.17.8 Project and Entry Helpers

This command toggles on and off the display of the Project Window and the Entry Helpers together.

26.17.9 All On/ Off

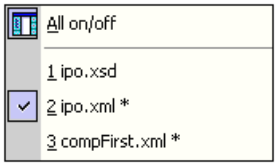
This command lets you switch all dockable windows on, or off:

- the Project Window
- the Info Window
- the three Entry-Helper Windows
- the Output Windows

This is useful if you want to hide all non-document windows quickly, to get the maximum viewing area for the document you are working on.

26.17.10 Currently Open Window List

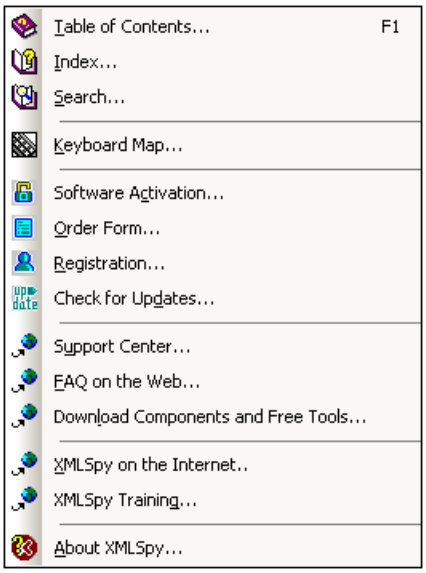
This list shows all currently open windows, and lets you quickly switch between them.



You can also use the Ctrl-TAB or CTRL F6 keyboard shortcuts to cycle through the open windows.

26.18 Help Menu

The **Help** menu contains commands required to get help or more information about XMLSpy, as well as links to information and support pages on the Altova web server.



The **Help** menu also contains the Registration dialog, which lets you enter your license key-code once you have purchased the product.

26.18.1 Table of Contents, Index, Search

Table of Contents

Description

Opens the onscreen help manual of XMLSpy with the Table of Contents displayed in the left-hand-side pane of the Help window. The Table of Contents provides an overview of the entire Help document. Clicking an entry in the Table of Contents takes you to that topic.

Index

Description

Opens the onscreen help manual of XMLSpy with the Keyword Index displayed in the left-hand-side pane of the Help window. The index lists keywords and lets you navigate to a topic by double-clicking the keyword. If a keyword is linked to more than one topic, a list of these topics is displayed.

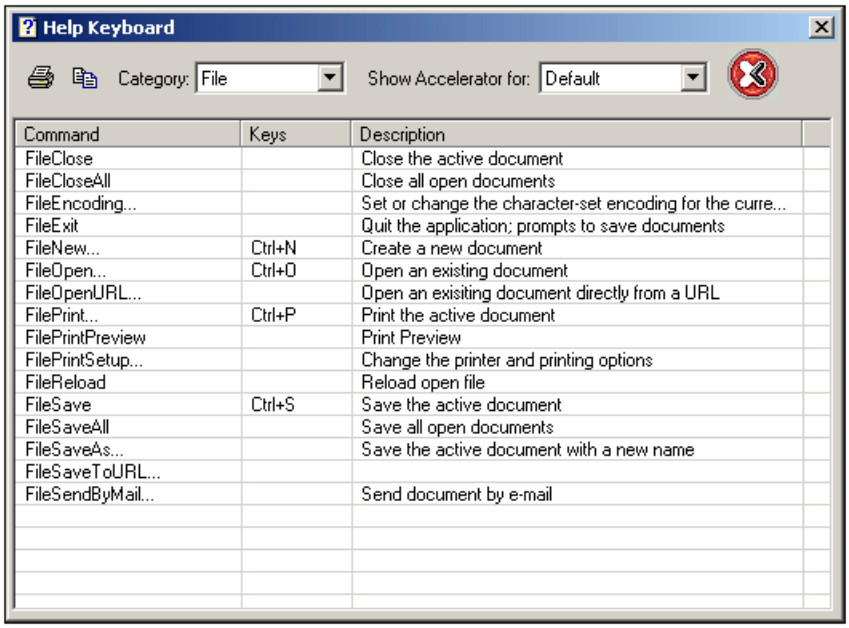
Search

Description

Opens the onscreen help manual of XMLSpy with the Search dialog displayed in the left-hand-side pane of the Help window. To search for a term, enter the term in the input field, and press **Return**. The Help system performs a full-text search on the entire Help documentation and returns a list of hits. Double-click any item to display that item.

26.18.2 Keyboard Map

The **Help | Keyboard Map** command causes an information box to be displayed that contains a menu-by-menu listing of all commands in XMLSpy. Menu commands are listed with a description and shortcut keystrokes for the command.



To view commands in a particular menu, select the menu name in the Category combo box. You can print the command by clicking the printer icon.

You should note the following points about shortcuts:

- Certain commands (and their shortcuts) are applicable only within a certain view. For example, most of the commands in the XML menu are applicable only in Grid View. Other commands (such as **File | Save** or **XML | Check Well-Formedness**) are available in multiple views.
- Other cool shortcuts: For example, **Shift+F10** brings up the context menu in Text View and Schema View; **Ctrl+E** when the cursor is inside an element start or end tag in Text View moves the cursor to the end or start tag, respectively.
- In the Keyboard tab of the Customize dialog, you can also set your own shortcuts for various menu commands.

26.18.3 Activation, Order Form, Registration, Updates

Software Activation

Description

After you download your Altova product software, you can activate it using either a free evaluation key or a purchased permanent license key.

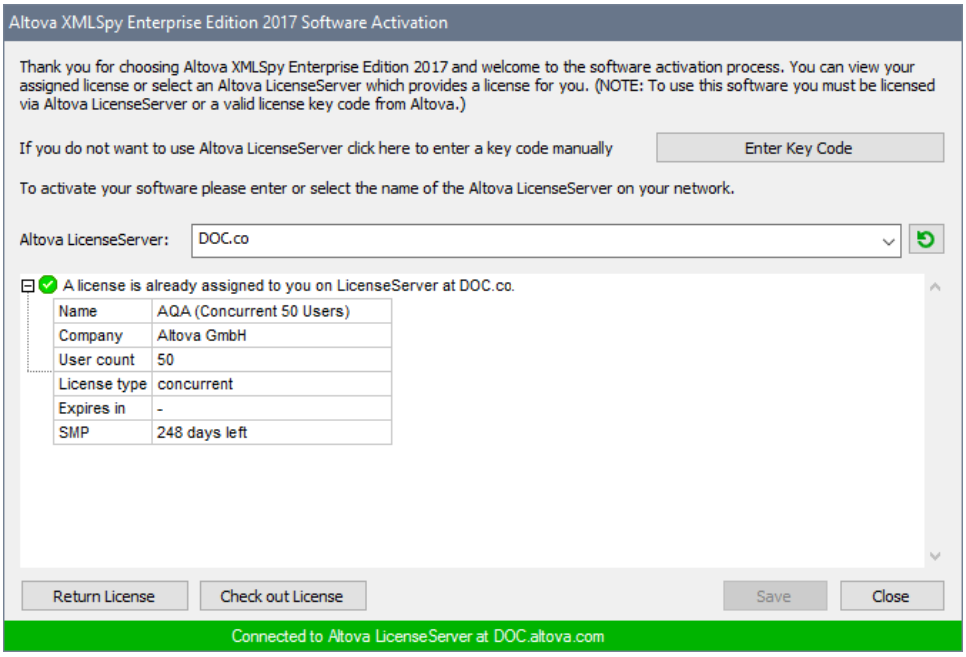
- ***Free evaluation key.*** When you first start the software after downloading and installing it, the Software Activation dialog will pop up. In it is a button to request a free evaluation key-code. Enter your name, company, and e-mail address in the dialog that appears, and click Request Now! The evaluation key is sent to the e-mail address you entered and should reach you in a few minutes. Now enter the key in the key-code field of the Software Activation dialog box and click **OK** to start working with your Altova product. The software will be unlocked for a period of 30 days.
- ***Permanent license key.*** The Software Activation dialog contains a button to purchase a permanent license key. Clicking this button takes you to Altova's online shop, where you can purchase a permanent license key for your product. There are two types of permanent license: single-user and multi-user. Both will be sent to you by e-mail. A *single-user license* contains your license-data and includes your name, company, e-mail, and key-code. A *multi-user license* contains your license-data and includes your company name and key-code. Note that your license agreement does not allow you to install more than the licensed number of copies of your Altova software on the computers in your organization (per-seat license). Please make sure that you enter the data required in the registration dialog exactly as given in your license e-mail.

Note: When you enter your license information in the Software Activation dialog, ensure that you enter the data exactly as given in your license e-mail. For multi-user licenses, each user should enter his or her own name in the Name field.

The Software Activation dialog (*screenshot below*) can be accessed at any time by clicking the **Help | Software Activation** command.

You can activate the software by either:

- Entering the license key information (click **Enter a New Key Code**), or
- Acquiring a license via an Altova LicenseServer on your network (click **Use Altova LicenseServer**, located at the bottom of the Software Activation dialog). The Altova LicenseServer must have a license for your Altova product in its license pool. If a license is available in the LicenseServer pool, this is indicated in the Software Activation dialog (*screenshot below*), and you can click **Save** to acquire the license.



Note that, once a license has been acquired, it cannot be returned to the LicenseServer for a period of seven days. After that time, you can return the license (click **Return License**) so that the license can be acquired by another client. A LicenseServer administrator, however, can unassign an acquired license via the administrator's Web UI of LicenseServer at any time.

Check out license

You can check out a license from the license pool for a period of up to 30 days so that the license is stored on the product machine. This enables you to work offline, which is useful, for example, if you wish to work in an environment where there is no access to your Altova LicenseServer (such as when your Altova product is installed on a laptop and you are traveling). While the license is checked out, LicenseServer displays the license as being in use, and the license cannot be used by any other machine. The license automatically reverts to the checked-in state when the check-out period ends. Alternatively, a checked-out license can be checked in at any time via the **Check in** button of the Software Activation dialog.

To check out a license, do the following: (i) In the Software Activation dialog, click **Check out License** (*see screenshot above*); (ii) In the License Check-out dialog that appears, select the check-out period you want and click **Check out**. The license will be checked out. The Software Activation dialog will display the check-out information, including the time when the check-out period ends. The **Check out License** button in the dialog changes to a **Check In** button. You can check the license in again at any time by clicking **Check In**. Because the license automatically reverts to the checked-in status, make sure that the check-out period you select adequately covers the period during which you will be working offline.

Note: For license check-outs to be possible, it must be enabled on the LicenseServer. If this functionality has not been enabled, you will get an error message to this effect. In this event, contact your LicenseServer administrator.

Altova LicenseServer provides IT administrators with a real-time overview of all Altova

licenses on a network, together with the details of each license, as well as client assignments and client usage of licenses. The advantage of using LicenseServer therefore lies in administrative features it offers for large-volume Altova license management. Altova LicenseServer is available free of cost from the Altova website. For more information about Altova LicenseServer and licensing via Altova LicenseServer, see the Altova LicenseServer documentation.

Order Form

Description

When you are ready to order a licensed version of the software product, you can use either the **Order license key** button in the Software Activation dialog (*see previous section*) or the **Help | Order Form** command to proceed to the secure Altova Online Shop.

Registration

Description

Opens the Altova Product Registration page in a tab of your browser. Registering your Altova software will help ensure that you are always kept up to date with the latest product information.

Check for Updates

Description

Checks with the Altova server whether a newer version than yours is currently available and displays a message accordingly.

26.18.4 Other Commands

Support Center

Description

A link to the Altova Support Center on the Internet. The Support Center provides FAQs, discussion forums where problems are discussed, and access to Altova's technical support staff.

FAQ on the Web

Description

A link to Altova's FAQ database on the Internet. The FAQ database is constantly updated as Altova support staff encounter new issues raised by customers.

Download Components and Free Tools

Description

A link to Altova's Component Download Center on the Internet. From here you can download a variety of companion software to use with Altova products. Such software ranges from XSLT and XSL-FO processors to Application Server Platforms. The software available at the Component Download Center is typically free of charge.

XMLSpy on the Internet

Description

A link to the Altova website on the Internet. You can learn more about XMLSpy and related technologies and products at the Altova website.

XMLSpy Training

Description

A link to the Online Training page at the Altova website. Here you can select from online courses conducted by Altova's expert trainers.

About XMLSpy

Description

Displays the splash window and version number of your product. If you are using the 64-bit version of XMLSpy, this is indicated with the suffix (x64) after the application name. There is no suffix for the 32-bit version.

26.19 Command Line

Certain XMLSpy actions can be carried out from the command line. These commands are listed below:

Open a file

Command: `xmlspy.exe file.xml`
Action: Opens the file, `file.xml`, in XMLSpy

Note: If an XML file has an SPS file already assigned to it, then the XML file is opened in Authentic View. Otherwise, the XML file is opened in Text View. If an SPS file is not assigned, one can be assigned with the `/sps` flag (*see below*).

Open multiple files

Command: `xmlspy.exe file1.xml file2.xml`
Action: Opens the files, `file1.xml` and `file2.xml`, in XMLSpy

Assign an SPS file to an XML file for Authentic View editing

Command: `xmlspy.exe myxml.xml /sps mysps.sps`
Action: Opens the file, `myxml.xml` in Authentic View with `mysps.sps` as its SPS file. The `/sps` flag specifies that the SPS file that follows is to be used with the XML file that precedes the `/sps` flag (for Authentic View editing).

Open a new XML template file via an SPS file

Command: `xmlspy.exe mysps.sps`
Action: Opens a new XML file in Authentic View. The display will be based on the SPS and the new XML file will have a skeletal structure based on the SPS schema. The name of the newly created XML file must be assigned when saving the XML file.

Open an SPS file as an XML document in Text View

Command: `xmlspy.exe /raw mysps.sps`
Action: Opens the file `mysps.sps` as an XML document in Text View. The `/raw` flag specifies that the SPS file that follows is to be edited as an XML file.

Altova XMLSpy 2017 Enterprise Edition

プログラマーのレファレンス

プログラマーのレファレンス

XMLSpy は、自動化されたサーバーです：プログラムすることのできるオブジェクトを自動化クライアントと呼ばれる他のアプリケーションに公開します。自動化クライアントは直接にオブジェクトおよび自動化サーバーにより使用することのできる機能にアクセスすることができます。ですから、XMLSpy の自動化クライアントは、例えば、XMLSpyの XML 検証機能を使用することができます。この結果、開発者は、XMLSpy の既存の機能を使用してアプリケーションを拡張することができます。

XMLSpy のプログラムすることのできるオブジェクトは、XMLSpy のアプリケーション API を使用して、自動化クライアントで使用することができます。XMLSpy のアプリケーション API は、以降省略してアプリケーション API と呼ばれます。アプリケーション API のオブジェクトモデル、および使用可能なすべてのオブジェクトの詳細はこのドキュメントで説明されています（Application API のセクションを参照してください）。

実行環境

アプリケーション API は、以下の環境の内部からアクセスすることができます：

- スクリプトエディター
- IDE プラグイン
- 外部プログラム
- ActiveX 統合

これらの環境について下で簡単に説明がされています。

スクリプトエディター：XMLSpy 機能のカスタム化と変更

XMLSpy のインストール機能を変更および追加することでカスタム化することができます。ユーザー入力のためのフォームを作成し、新しいメニューコマンドおよびツールバーショートカットを含むユーザーインターフェイスを変更することもできます。これらすべての機能は、アプリケーション API のオブジェクトとの作動するスクリプトを書くことにより達成することができます。これらのタスクを効率的に実行するために、XMLSpy は、ビルドインのスクリプトエディターを搭載しています。機能の完全な説明と使用方法はこのドキュメントのスクリプトエディターセクションで説明されています。サポートされるプログラミング言語は、**JScript** および **VBScript** です。

IDE プラグイン：XMLSpy のためのプラグインの作成

XMLSpy により自身のプラグインを作成し、XMLSpy に統合することができます。これを行うには、XMLSpy のプラグインのための特別なインターフェイスを使用します。プラグイン作成に関する詳細は、XMLSpy IDE プラグインのセクションで説明されています。

アプリケーションオブジェクトは、IDE プラグインにより実装され、アプリケーションにより呼び出される必要があるメソッドにパスされます。IDE プラグインの実装のために使用される 典型的な言語は、**C#** および **C++** です。詳細に関しては、XMLSpy IDE プラグインのセクションを参照してください。

外部プログラム

さらに、外部スクリプトともにXMLSpy を操作することができます。例えば、XMLSpy を指定した時刻に起動し、XMLSpy 内のファイルを開き、ファイルを検証し、印刷するスクリプトを作成することができます。外部スクリプトは、アプリケーション API を使用して、これらのタスクを実行します。 アプリケーション API の詳細に関しては、アプリケーション API のセクションを参照してください。 .

XMLSpy の外部からアプリケーションAPI を使用する場合は、XMLSpy のインスタンスが最初に開始されることが必要です これがどのように行われるかは、使用されるプログラミング言語によります。個別の言語に関する情報は、プログラミング言語のセクションを参照してください。

基本的には、XMLSpy はCOM 登録を使用して起動されます。そして、XMLSpy インスタンスと関連する `Application` オブジェクトを返します。COM 設定により、既に実行中のXMLSpy に関連したオブジェクトが返されます。 COM オブジェクトの作成と呼び出しをサポートするプログラミング言語を使用することができます。共通する項目は、下にリストされています。

- JScript およびVBScript スクリプト ファイルは、シンプルな構文を持ち、COM オブジェクトにアクセスするためにデザインされています。DOS コマンド ラインから直接、またはWindows Explorer をダブルクリックして実行することができます。これらのスクリプトはシンプルな 自動化 タスクのために使用されます。
- C# は、広範囲の既存の機能を搭載した完全なプログラミング言語です。Access to COM オブジェクトへのアクセスは、自動的にラップされたC# を使用します。
- C++ は、COM への直接的なコントロールを与えますが、他の言語よりも多量のコードが必要です。
- Java: Altova 製品には、アプリケーションAPI をラップして、Java の完全な外観と使用感を与える、ネイティブのJava クラスが搭載されています。
- 役に立つ代替の他のプログラミング言語は、次のとおりです :アプリケーション、Perl、およびPython のための Visual Basic

ActiveX 統合

XMLSpy ActiveX コントロールを使用してアプリケーションAPI にアクセスする特別の場合、この機能は、XMLSpy 統合パッケージがインストールされている場合のみ使用することができます。すべてのActiveX コントロールには、元になる機能のための対応するCOM オブジェクトを返すプロパティがあります。マネージャーコントロールは`Application` オブジェクト、ドキュメントコントロール、`Document` オブジェクト、およびプレースホルダオブジェクトを与え、プロジェクトツリーを含む箇所では、`Project` オブジェクトを返します。これらのオブジェクトによりサポートされるメソッドは、アプリケーションAPI のインターフェイスのセクションで説明されています。ActiveX コントロール統合 のコンテキストでは、このメソッドを使用する必要がなくこの点を考慮する必要があります。 詳細に関してはActiveX 統合を参照してください。

プログラマーのレファレンスについて

XMLSpy のプログラマーのレファレンスは、次のセクションから構成されています :

- スクリプトエディター: XMLSpy で使用することのできるスクリプト環境のためのユーザー参照。
- IDE プラグイン: XMLSpy のための プラグイン作成の説明
- アプリケーションAPI: アプリケーションAPI のためのレファレンス
- ActiveX 統合: ActiveX コントロールを使用した、XMLSpy GUI およびXMLSpy 機能の統合の方法の案内および参照。

1 Scripting Editor

The Scripting Editor of XMLSpy uses the Form Editor components of the Microsoft .NET Framework, and thus provides access to the Microsoft .NET Framework. This means that JScripts and VBScripts not only work with the XMLSpy API—which is a COM API and the API of XMLSpy—but can also access and use classes of the Microsoft .NET framework.

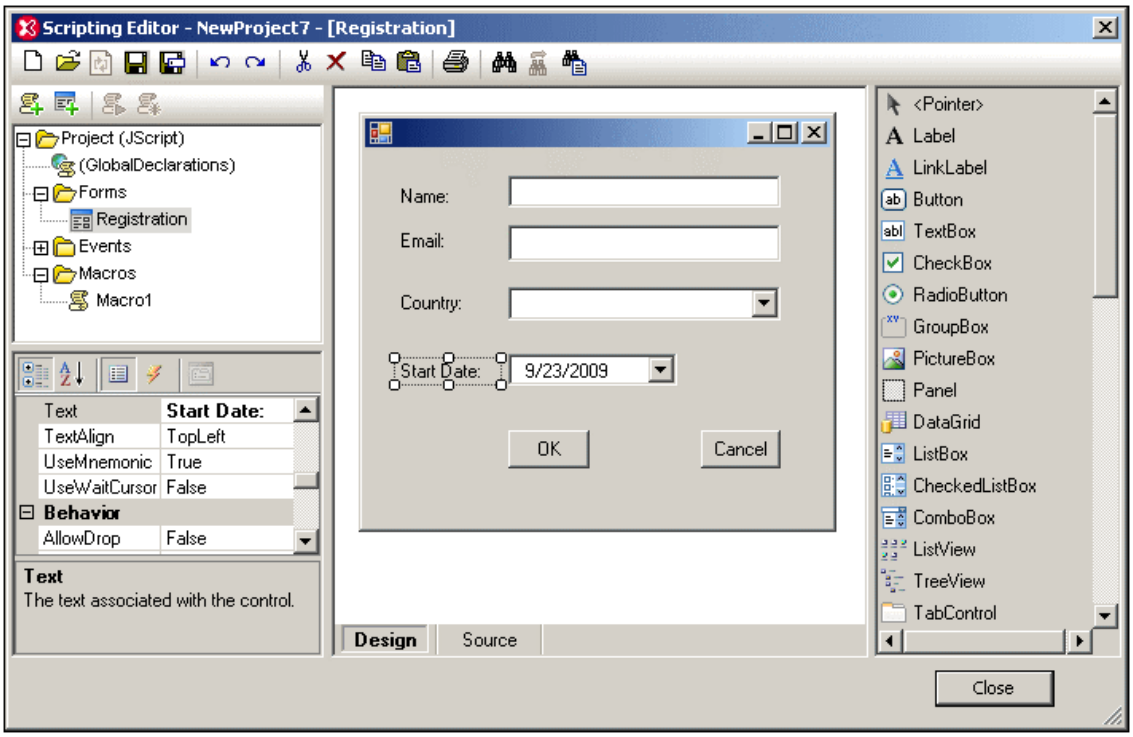
You can therefore create and use your own macros and forms within XMLSpy, and thus add to and modify the functionality of your installation of XMLSpy.

Note: Visual Basic is **not supported** as a language in the scripting environment. Only VBScript and JavaScript are. Ensure that you use VBScript syntax and not Visual Basic syntax in the scripting environment.

Note: Microsoft's **.NET Framework 2.0 or higher** is a system prerequisite for Scripting Editor, and it must be installed before XMLSpy is installed.

The Scripting Editor

The Scripting Editor (*screenshot below*) opens in a separate window and is accessed via the **Tools | Scripting Editor** menu command in the XMLSpy GUI. The programming languages that can be used in the Scripting Environment are **JScript** and **VBScript**. The scripting language can be changed by right-clicking the Project item in the Project window, selecting **Scripting Language**, and selecting the language you want.



What you can do with the Scripting Editor

In the Scripting Editor, you can create Forms, Event Handlers, and Macros to build up a

Scripting Project. A Scripting Project can then be set as the Global Scripting Project for XMLSpy, thus enabling scripts in the Scripting Project to be used in the application. Additionally, different Scripting Projects can be assigned to different XMLSpy projects, thus allowing different scripts to be used for different XMLSpy projects.

Every script project can define the .NET runtime version it wants to use. An application can handle multiple scripting projects with different .NET runtime versions simultaneously, but the appropriate .NET version must be installed. For example, script projects with .NET 4.0 will only run on computers having .NET 4.0 installed.

Documentation about the Scripting Editor

The documentation describing the Scripting Environment (this section) is organized into the following parts:

- An overview, which provides a high level description of the Scripting Editor and Scripting Projects.
- A list of steps required to create a Scripting Project.
- An explanation of Global Declarations, together with an example.
- A description of how to create Forms.
- A discussion of XMLSpy-specific event handlers.
- An explanation of how to use macros in the Scripting Editor and in XMLSpy.

1.1 Overview

The Scripting Editor provides an interface in which you can: (i) graphically design Forms while assigning scripts for components in the Form; (ii) create Event Handlers, and (iii) create Macros.

These Forms, Event Handlers, and Macros are organized into scripting projects, which are then assigned to XMLSpy application projects and can be used in the application.

Variables and functions can be defined in a Global Declarations script, which is always executed before Macro or Event Handler scripts.

This section gives an overview of the Scripting Editor and Scripting Projects. It is organized into the following sections:

- Scripting Projects in XMLSpy, which describes how the scripting projects you create with the Scripting Editor will be used in XMLSpy.
- The Scripting Editor GUI, which provides a detailed look at the different parts of the Scripting Editor GUI and how they are to be used.
- Components of a Scripting Project, which explains the different components that go to make up a scripting project.

The details about the creation of the various components (Global Declarations, Forms, Event Handlers, and Macros) are described in their respective sections.

.NET assemblies

Every scripting project can have references to .NET assemblies—in addition to the default references. .NET assemblies can be added for the whole scripting project or for individual macros (by using the new `CLR.LoadAssembly` command in the source code; see Built-in Commands). Assemblies can be added, for example, from the Global Assembly Cache.

To add an assembly, right-click the project or macro, and, from the context menu that pops up, select **Add .NET Assembly | Assembly from Global Cache (GAC)**.

This works in the same way as with Visual Studio and allows access not only to the complete Microsoft .NET Framework but also to any user-defined assembly.

1.1.1 Scripting Projects in XMLSpy

All scripts and scripting information created in the Scripting Editor are stored in **Altova Scripting Projects** (**`asprj` files**).

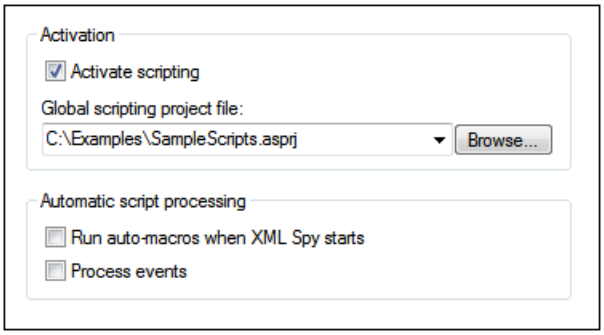
You can create any number of Altova Scripting Projects. After a scripting project has been created, it can be used in the following ways:

- It can be set as the global scripting project for XMLSpy. Scripts in the global scripting project can then be called from within the application, and macros of the Global Scripting Project can be used for all XMLSpy projects.
- It can be assigned to an XMLSpy project (as an application project). When an XMLSpy project is open in XMLSpy, scripts in the associated scripting project can be called.

Your XMLSpy package contains a sample scripting project called `SampleScripts.asprj`. This file is located in the folder: `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\` and contains global declarations for a few standard tasks.

Setting the global scripting project of an application

The global scripting project of an application is set in the Scripting tab of the Options dialog of XMLSpy (*screenshot below*, **Tools | Options**).



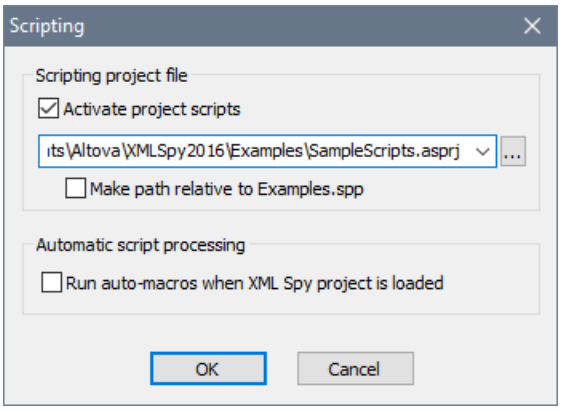
To set a global scripting project for XMLSpy, check the *Activate Scripting* check box and then browse for the Altova Scripting Project (`.asprj`) file you want. You can also specify: (i) whether Auto-Macros in the scripting project should be automatically executed when XMLSpy starts, and (ii) whether application event handler scripts in the project should be automatically executed or not; check or uncheck the respective check boxes accordingly.

Note: Nested script execution is possible, i.e. Macros can call other macros, and events are received during macro, or event, execution.

Assigning a scripting project to an XMLSpy project

A scripting project is assigned to an XMLSpy project as follows:

1. In the XMLSpy GUI, open the required application project.
2. Select the menu command **Project | Script Settings**. The Scripting dialog (*screenshot below*) opens.



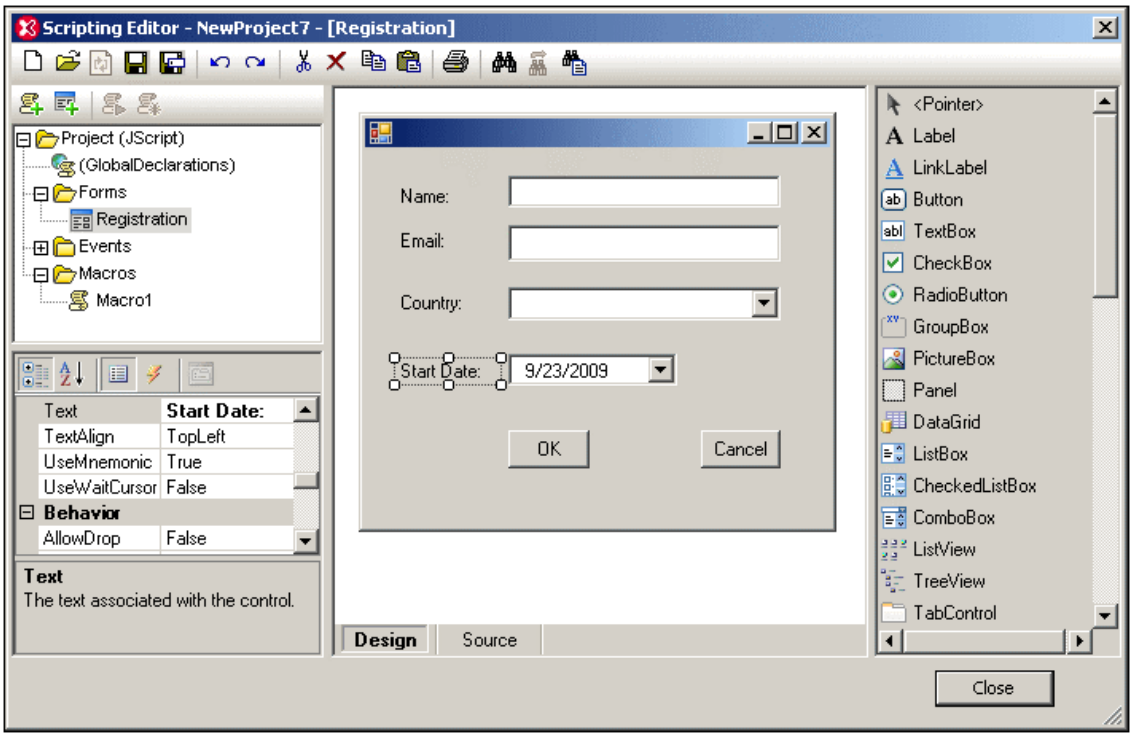
3. Check the *Activate Project Scripts* check box and select the required scripting project (`asprj` file). If you wish to run Auto-Macros when the XMLSpy project is loaded, check the *Run Auto-Macros* check box.
4. Click **OK** to finish.

Note: To deactivate (that is, unassign) the scripting project of an XMLSpy project, uncheck the *Activate Project Scripts* check box.

1.1.2 The Scripting Editor GUI

The Scripting Editor GUI is shown below. It has the following parts:

- A toolbar
- A Scripting Project Tree pane (top left-hand side)
- A Properties and Events pane (bottom left)
- A Main Window with Design and Source tabs
- A Form Object Palette (right-hand side)



Scripting Editor toolbar

The Scripting Editor toolbar contains icons for:

- Standard file commands such as **New**, **Open**, **Save**, and **Print**. These commands are used to create new scripting projects, open existing scripting projects, and save and print scripting projects.
- Standard editing commands such as **Copy**, **Paste**, **Undo**, **Redo**, **Find**, and **Replace**. Note that the **Find** and **Replace** commands are applied to code in the Source tab of the Scripting Editor.

Scripting Project Tree

The Scripting Project Tree (*screenshot below*) shows the various components of the scripting project, structured along four main branches: (i) Global Declarations, (ii) Forms, (iii) Events, and (iv) Macros.

The Scripting Project Tree provides access to each component of the scripting project. For example, in order to display and edit a particular Form, expand the Forms folder in the tree (*see screenshot above*), right-click the Form you wish to display or edit, and click **Open** from the context menu that pops up.

A quicker way to open a Form, Event, macro, or the Global Declarations script, is to double-click the respective icon, or text. To delete a Form or Macro from the scripting project, right-click the component and select the **Delete** command from the context menu.

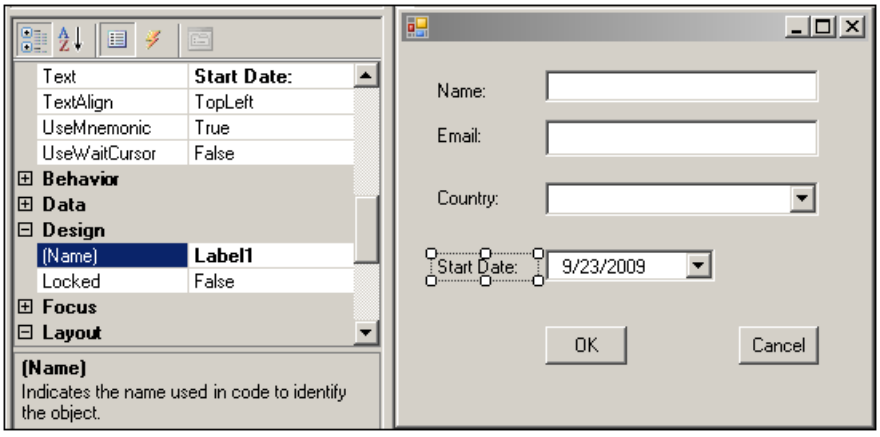
The Scripting Project Tree pane contains a toolbar with icons (*screenshot below*).

The icons, from left to right, are for: (i) creating a new macro, (ii) creating a new form, (iii) running a macro, and (iv) debugging a macro. These commands are also available in the context menu that appears when you right-click any component in the Scripting Project Tree.

Properties and Events

The Properties and Events pane (*screenshot below*) displays the following:

- Form properties, when the Form is selected
- Object properties, when an object in a Form is selected. (The screenshot below shows, at left, the properties of the object selected in the Form at right.)
- Form events, when a Form is selected
- Object events, when an object in a Form is selected



To switch between the properties and events of the selected component, click, respectively, the **Properties** icon (third from left in the Properties and Events toolbar, *see screenshot above*) and the **Events** icon (fourth from left).

The first and second icons from left in the toolbar are, respectively, the **Categorized** and **Alphabetical** icons. These display the properties or events either organized by category or organized in ascending alphabetical order.

When a property or event is selected, a short description of it is displayed at the bottom of the Properties and Events pane.

Main Window

The Main Window displays one component at a time and has one or two tabs depending on what is being displayed. If a Global Declarations script, an Event, or a Macro is being displayed, then a single tab, the Source tab, displays the source code of the selected component.

The Source tab supports:

- syntax coloring
- source code folding
- setting/deleting bookmarks using **CTRL+F2**
- autocompletion entry helper with parameter info
- Goto Brace, Goto Brace Extend
- Zoom In / Zoom Out
- full method/property signature shown next to the autocompletion entry helper
- brace highlighting during code entry

```
if ( x == y.GetName( a, b, c() ) )
```

- mouse over popups; placing the mouse over a known method or property, displays its signature (and documentation if available)

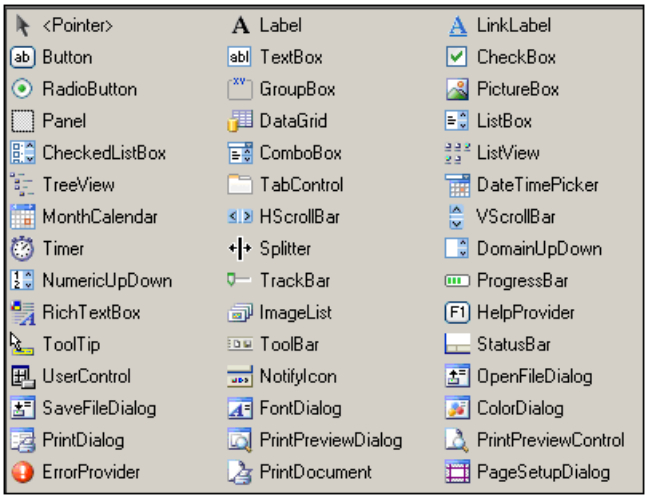
If a **Form** is being displayed, then the Main Window has two tabs: a Design tab showing and enabling the layout of the Form, and a Source tab containing the source code for the Form. Content in both the Design tab and Source tab can be edited.

Note: Since JScript and VB Script are untyped languages, entry helpers and auto-completion is supported only in cases of "fully qualified constructs" and "predefined" names.

If names start with `objDocument`, `objProject`, `objXMLData`, or `objAuthenticRange`, members of the corresponding interface will be shown. Auto-completion entry helper and parameter info are shown during editing, but can also be obtained on demand by pressing **Ctrl+Space**.

Form Object Palette

The Form Object Palette contains all the objects that are available for designing Forms and looks something like the screenshot below. Registered ActiveX controls can be added to the Form Object Palette by right-clicking the pane and selecting the **Add ActiveX Control** command



To insert an object from the Form Object Palette click the object you want in the palette, then click at the location in the Form where you wish to insert the object. The object will be placed at this location. In many cases you will need to supply some properties of the object via the Properties and Events pane. You can drag the object to other locations as well as resize it. Further, a number of editing commands, such as centering and stacking objects, can be accessed via the context menu of the selected Form object.

Some Form objects, such as `Timer`, are not added to the Form but are created as Tray Components in a tray at the bottom of the Main Window. You can select the object in the tray and set properties and event handlers for the object via the Properties and Events pane. For an example of how Tray Components are handled, see Form usage and commands.

1.1.3 Components of a Scripting Project

An Altova Scripting Project consists of the following four major components:

- *Global Declarations*, a component which contains definitions of variables and functions that are available to, and can be used by, all Forms, Macros, and Event Handler scripts in the scripting project.
- *Forms*, a component which contains all the Forms defined in the scripting project.
- *Events*, a component which contains Event Handler scripts for all application-based—as opposed to Form-based—events.
- *Macros*, a component which contains all the Macros defined in the scripting project.

These components are displayed in and accessed via the Scripting Project Tree of the Scripting Editor (*screenshot below*).

Given below is a brief description of each of these components.

Global Declarations

The Global Declarations component is a script that contains variables and functions that can be used by Forms, Event Handlers, and Macros. The functions make use of the XMLSpy API to access XMLSpy functionality. Creating a variable or function in the Global Declarations module enables it to be accessed from all the Forms, Event Handlers and Macros in the scripting project.

To add a variable or function, open the Global Declarations component (by right-clicking it in the Scripting Project Tree and selecting **Open**) and edit the Global Declarations script in the Main Window. In this script, add the required variable or function.

Forms

In the Scripting Editor, you can build a Form graphically using a palette of Form objects such as text input fields and buttons. For example, you can create a Form to accept the input of an element name and to then remove all occurrences of that element from the active XML document.

For such a Form, a function script can be associated with a text box so as to take an input variable, and an Event Handler can be associated with a button to start execution of the delete functionality, which is available in the XMLSpy API. A Form is invoked by a call to it either within a function (in the Global Declarations script) or directly in a Macro. For details of how to create and edit Forms, see the Forms section.

Event handling

Event Handler scripts can be associated with a variety of available events. You can control events that occur both within Forms (Form events) and within the general application interface (application events). The script associated with an event is executed immediately upon the triggering of that event.

Most events have parameters which provide detailed information about the event. The return value from the script typically instructs the application about how to continue its processing (for example, the application may not allow editing).

An Event Handler runs when the relevant event occurs in the Form or in XMLSpy. For details about how to create event handlers, see Event Handlers.

Macros

Macros are used to implement complex or repetitive tasks. Macros do not use either parameters or return values.

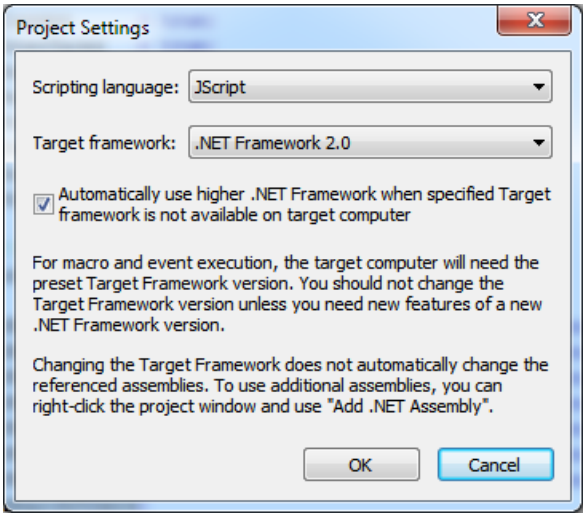
In a Macro, it is possible to access all variables and functions declared in the Global Declarations and to display Forms for user input.

For a simple example of creating a Macro, see Writing a Macro. Also see Running Macros for a description of the ways in which a Macro can be called. A Macro is run from within the XMLSpy interface by clicking **Tools | Macros | *[MacroName]***

1.2 Creating a Scripting Project

The broad steps for creating a Scripting Project are as follows:

1. Open the Scripting Editor by clicking the command **Tools | Scripting Editor**.
2. In the Scripting Editor, open a new scripting project by clicking the **New** icon in the Scripting Editor toolbar. The Project Settings dialog (*screenshot below*) pops up. You can also access this dialog by right-clicking a project in the Scripting Project Tree pane (in the top left part of the Scripting Editor window) and clicking the command **Project Settings**.



   Select either JScript or VBScript in the first combo box and the .NET Framework in the second combo box. To enable higher .NET Frameworks (such as .NET Framework 4.5 on Windows 8), check the *Automatically use higher .NET Framework* check box. Then click **OK**. The new Scripting Project is created.
3. Click the **Save** icon in the Scripting Editor toolbar to save the Scripting Project as a `.asprj` file.
4. A Scripting Project can be considered to be made up of several components that work together. These components will typically be a combination of: Global Declarations, Forms, Events, and Macros. They can be created in any order, but you should clearly understand how they work together. The way each type of component is called and executed is described below. How to create each type of component is described in the respective sections about the component type.
5. After you have finished creating all the required components, save the Scripting Project (by clicking the **Save** icon in the Scripting Editor toolbar).
6. Close the Scripting Editor.

Note: Right-clicking the Project folder and selecting **Project Settings** lets you change the scripting language at any time.

How Forms, Event Handlers, and Macros are called and executed

Forms, Event Handlers, and Macros are all created in the Scripting Editor. However, the way they are called and executed is different for each and has a bearing on how you create your scripting projects.

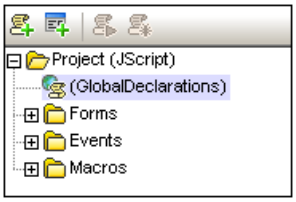
- A Form is invoked by a call to it either within a function in the Global Declarations script or directly in a Macro.
- An Event Handler runs when the relevant event occurs in XMLSpy. If an Event Handler for a single event is defined in both the Global Scripting Project and the XMLSpy-project-

specific Scripting Project, then the event handler for the project-specific Scripting Project is executed first and that for the Global Scripting Project immediately afterwards.

- A Macro is executed from within the XMLSpy interface by clicking **Tools | Macros | *[MacroName]***. In a Macro, it is possible to access all variables and functions declared in the Global Declarations and to display Forms for user input.

1.3 Global Declarations

The Global Declarations component is present by default in every Scripting Project (*see screenshot below*), and therefore does not have to be created. In order to add variables and functions to the Global Declarations script of a Scripting Project, you need to open the Global Declarations script and add the code fragment to the Global Declarations script. See Components of a Scripting Project and Creating a Scripting Project for more information.



To open the Global Declarations script of a Scripting Project, right-click the *Global Declarations* item in the Scripting Project Tree (*screenshot above*), and select **Open**. The Global Declarations script opens in the Main Window.

Note: Every time a macro is executed or an event handler is called, global declarations are re-initialized.

Given below is an example function. Remember that creating a variable or function in the Global Declarations script makes this variable or function accessible to all Forms, Event Handlers, and Macros.

Example function

A function called `RemoveAllNamespaces` would have code like this:

```
function RemoveAllNamespaces(objXMLData)
{
      if(objXMLData == null)
   return;

  if(objXMLData.HasChildren) {
    var objChild;

    // spyXMLDataElement := 4
    objChild = objXMLData.GetFirstChild(4);

    while (objChild)  {
    RemoveAllNamespaces(objChild);

          try {
             var nPos,txtName;
             txtName = objChild.Name;

             if((nPos = txtName.indexOf(":")) >= 0)      {
                objChild.Name = txtName.substring(nPos+1);
             }

    objChild = objXMLData.GetNextChild();
          }
          catch (Err)              {
             objChild = null;
          }
      }
    }
```

```
}
```

Note:

- It is possible to define local variables and helper functions within macros and event handlers. Example:

```
//return value: true allows editing
//return value: false disallows editing
var txtLocal;
function Helper()
{
  txtMessage = txtLocal;
  Application.ShowForm ("MsgBox");
}
function On_BeforeStartEditing(objXMLData)
{
  txtLocal= "On_BeforeStartEditing()";
  Helper();
}
```

- Recursive functions are supported.

1.4 Forms

Creating and editing Forms in the Scripting Editor consists of the following steps:

1. Creating a New Form. The new Form is created and named, and has properties defined for it.
2. Designing the Form. A Form is designed by adding Form Objects to it and assigning values for the different Form Objects.
3. Scripting Form Events. Scripts are assigned to Form-related events.

1.4.1 Creating a New Form

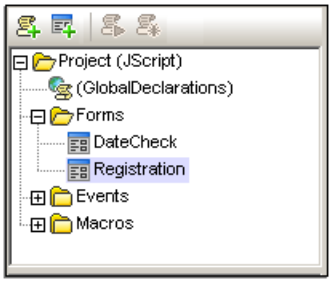
Creating a new Form in the Scripting Editor involves the following steps:

1. Creating a new Form and naming it
2. Specifying the properties of the Form

Creating a new Form and naming it

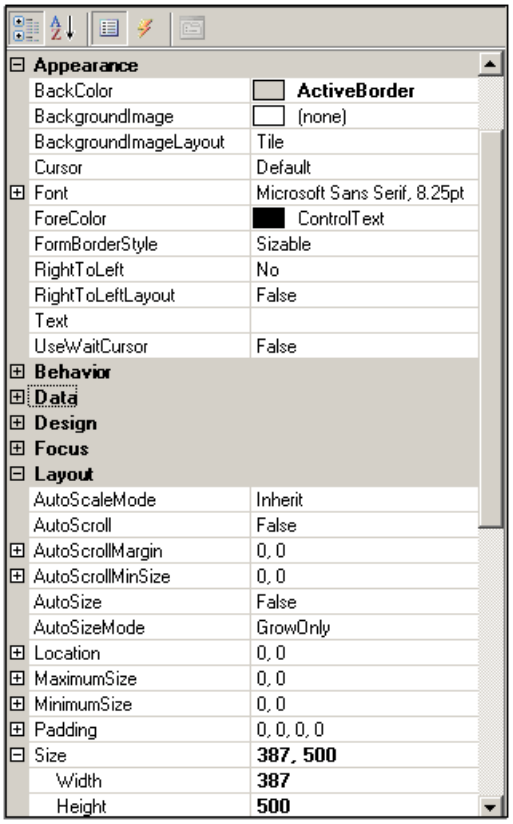
To add a new Form to a scripting project, click the **Add Form** icon (*highlighted in screenshot below*) in the toolbar of the Project Overview pane. Enter the name of the new Form.

A new Form is added to the project. It appears in the Main Window and an entry for it is created in the Scripting Project Tree pane, under the Forms heading. Press the F2 function key to rename the form, or right click the form name and select Rename from the context menu. In the screenshot below, we have named the new Form *Registration*.



Form properties

The properties of the Form, such as its size, background color, and font properties, can be set in the Properties pane. The screenshot below shows the size and background-color property values in bold, in the *Layout* and *Appearance* categories, respectively.



Testing a Form

You can test a form in the Scripting Editor by right-clicking it in the Project Overview pane and selecting the **Test Form** Command.

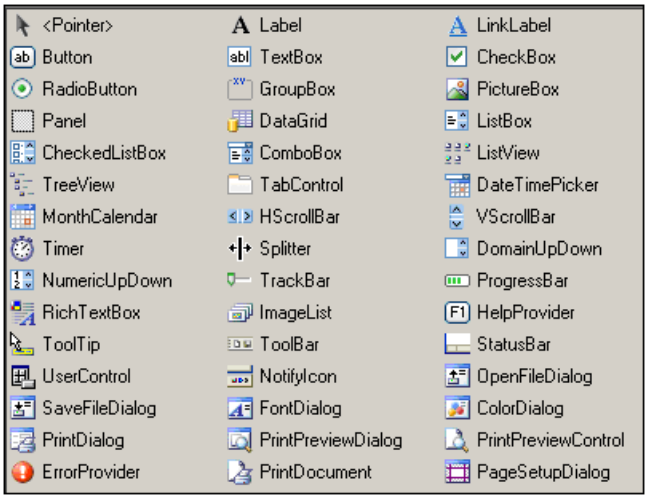
1.4.2 Form Design and Form Objects

Designing a Form consists of the following steps:

- Placing an object from the Form Object Palette in the Form design.
- Assigning values for the properties of individual Form Objects.
- Assigning scripts for Form-based events.

The Form Object Palette

The Form Object Palette contains all the objects that are available for designing Forms and looks something like the screenshot below. Registered ActiveX controls can be added to the Form Object Palette by right-clicking the pane and selecting the **Add ActiveX Control** command



To insert an object from the Form Object Palette click the object you want in the palette, then click at the location in the Form where you wish to insert the object. The object will be placed at this location. In many cases you will need to supply some properties of the object via the Properties and Events pane. You can drag the object to other locations as well as resize it. Further, a number of editing commands, such as centering and stacking objects, can be accessed via the context menu of the selected Form object.

Some Form objects, such as `Timer`, are not added to the Form but are created as Tray Components in a tray at the bottom of the Main Window. You can select the object in the tray and set properties and event handlers for the object via the Properties and Events pane. For an example of how Tray Components are handled, see Form usage and commands.

Some of the most commonly used objects are described below:

Label: Adds text fields such as captions or field descriptions.

Button: Adds a button. It is possible to assign bitmaps as background images for these buttons.

Check Box: Adds a check box, which enables `Yes/No` type selections.

Combo Box: Adds a combo box, which allows the user to select an option from a drop-down menu.

List Box: Adds a list box, which displays a list of items for selection.

TextBox: Enables the user to enter a single line of text.

Rich TextBox: Enables the user to enter multiple lines of text.

Creating objects and setting their properties

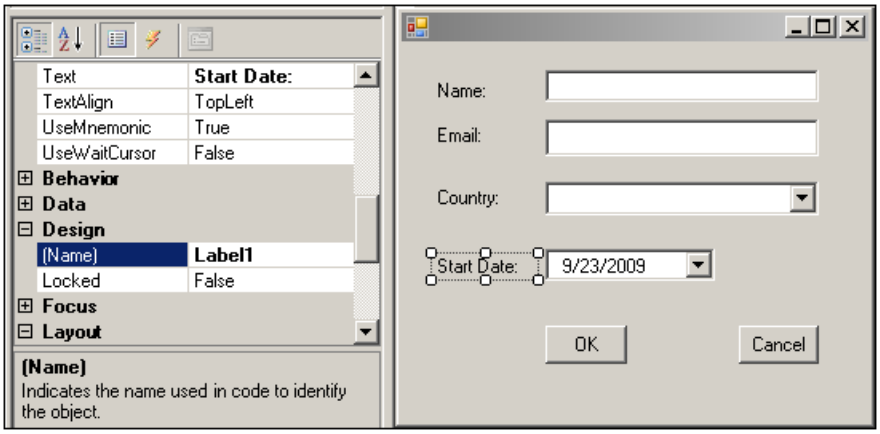
To create an object in the Form, first select the required object in the Form Object Palette and then click the location in the Form where you want to insert it. After the object has been inserted, you can resize it as well as drag it to another location in the Form.

When an object is selected in the design, you can specify its properties in the Properties and

Events pane. In the toolbar of the Properties and Events pane, click the Properties icon to display a list of the object's properties.

For example, in the screenshot below, the Label object with the text *Start Date* has been selected in the design. In the Properties and Events pane, the name of the object (which is the name that is to be used to identify the object in code, `Label1` in the screenshot below) is given in the *Design* category of properties; in this case, the name of the object is `Label1`.

The text of the label (which is what appears in the Form) must be entered as the value of the *Text* property in the *Appearance* category of properties.



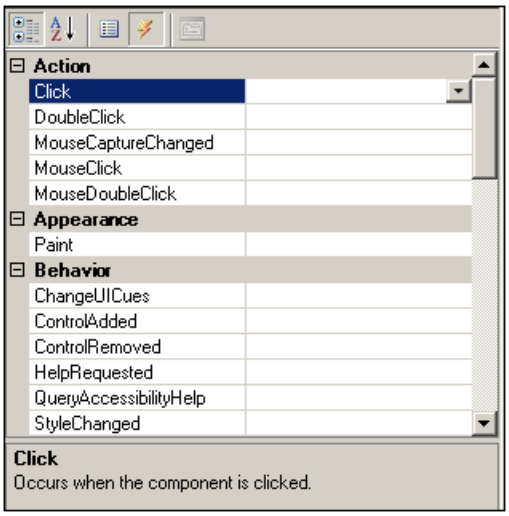
To assign other object properties, enter values for them in the Properties and Events pane.

Testing a Form

You can test a form in the Scripting Editor by right-clicking it in the Project Overview pane and selecting the **Test Form** Command.

1.4.3 Form Events

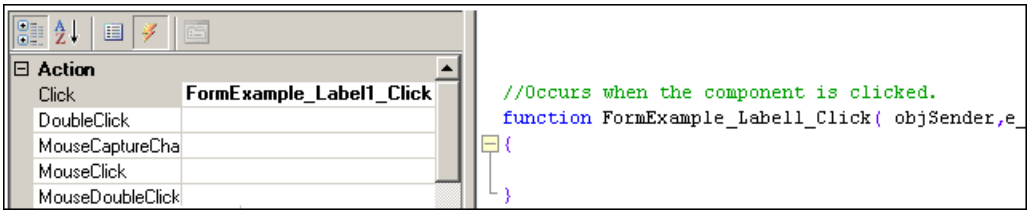
When an object is selected in the design, clicking on the Events icon in the toolbar of the Properties and Events pane (*fourth icon from left*), displays all the events available for that object (*see screenshot below*). These can be displayed either by category (*screenshot below*) or alphabetically.



For each event, you can enter the name of an existing event handler or function. Alternatively:

- you can double click on an event to create: (i) an empty function script in the *Source* tab of the Main Window, and (ii) an association of the newly created function with the selected event.
- double click a button in the design tab, to directly generate the handler stub in the code window.

The screenshot below was taken after the *Click* event was double-clicked. Notice that an empty event handler function called `FormExample_Label1_Click` has been created in the Main Window and that, in the Properties and Events pane, this function has been associated with the *Click* event.



Enter the required scripting code and save the project.

Writing the required scripts

After the visual design of the form is complete, form objects will typically be associated with suitable scripts. The example below is a script that adds colors when a button is clicked. The script is inserted as an event handler for the `Click` event of the button `Button1` (the event is available in the Properties and Events pane when the button is selected in the design):

```
function FormExample_Button1_Click( objSender, e_EventArgs )
{
	// Sets the ForeColor (red) of the button.
	objSender.ForeColor = CLR.Static( "System.Drawing.Color" ).Red;
	// Sets the BackColor (blue) of the button.
	objSender.BackColor = CLR.Static( "System.Drawing.Color" ).Blue;
	// Sets the form BackColor (green).
	objSender.FindForm().BackColor =
CLR.Static( "System.Drawing.Color" ).Green;
}
```

1.5 Events

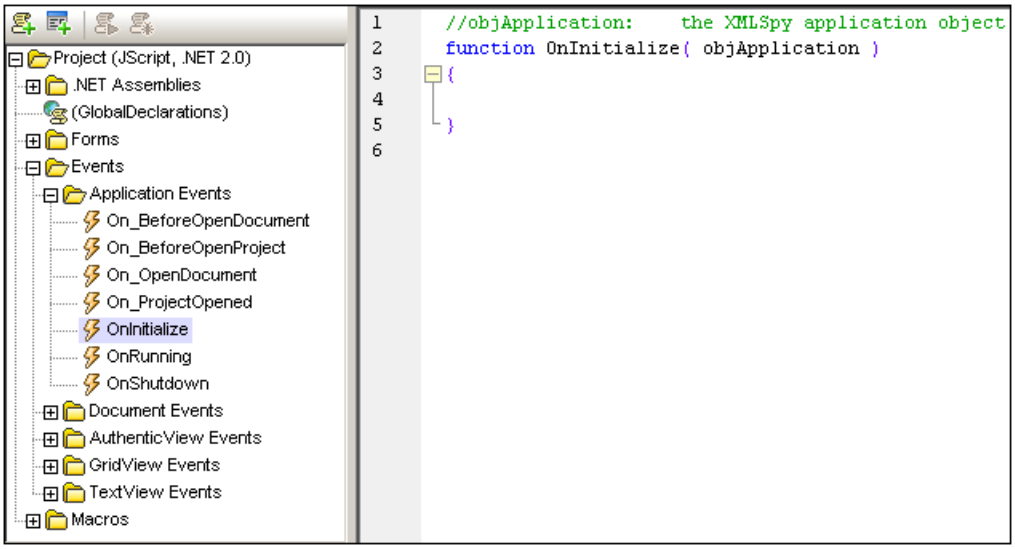
The Events folder of the scripting project (*see screenshot below*) contains folders for the following type of events:

- Application Events
- Document Events
- Authentic View Events
- Grid View Events
- Text View Events

Note that these events are XMLSpy-specific, as opposed to Form-based events. Each of the folders listed above contains a set of events for which Event Handler scripts can be written.

Application Events, for example, are shown in the screenshot below.

To access the event handler script of any of these events, right-click the event and select **Open** from the context menu. The script will be displayed in the Main Window (*see screenshot below*) and can be edited there. After you have finished editing the script, save changes by clicking the **Save** command in the toolbar of the Scripting Editor.



Note the following points:

- Event Handlers need function headers with the correct spelling of the event name. Otherwise the Event Handler will not be called.
- It is possible to define local variables and helper functions within Macros and Event Handlers. Example:

```
//return value: true allows editing
//return value: false disallows editing
var txtLocal;
function Helper()
{
  txtMessage = txtLocal;
  Application.ShowForm ("MsgBox");
}
function On_BeforeStartEditing(objXMLData)
{
  txtLocal= "On_BeforeStartEditing()";
```

```
  Helper();
}
```

- In order for events to be processed, the Process Events options must be toggled on in the Scriptings options of XMLSpy. See Scripting Projects in XMLSpy for details.
- Also see Programming Points.

Application Events

OnInitialize
The `OnInitialize` event is raised after the main window becomes visible but before any project is loaded. This event is not raised if the application can't be loaded at all.

OnRunning
If the application is completely loaded and after the `OnInitialize` event occurs, the `OnRunning` event is raised.

OnShutdown
The event is raised after any open project and all documents have been closed on shutdown of the application. The main window is no longer visible.

Example

The following script is an Event Handler for the `On_BeforeOpenProject` event. It allows you to add a script that will be executed each time before XMLSpy opens a project. The example script below sequentially opens all XML files located in the XML folder of the project and validates them. If the validation fails, the script shows the validation error and stops. If a file passes the validity test, it will be closed and the next file will be opened.

Enter the following script for the `On_BeforeOpenProject()` event, and then save the scripting project.

```
function On_BeforeOpenProject()
{
 var bOK;
 var nIndex,nCount;
 var objItems,objXMLFolder = null;

 objItems = Application.CurrentProject.RootItems;
 nCount = objItems.Count;

 // search for XML folder
 for(nIndex = 1;nIndex <= nCount;nIndex++)      {
  var txtExtensions;
  txtExtensions = objItems.Item(nIndex).FileExtensions;

 if(txtExtensions.indexOf("xml") >= 0)    {
  objXMLFolder = objItems.Item(nIndex);
 break;
 }
}

// does XML folder exist?
if(objXMLFolder) {
 var objChild,objDoc;

 nCount = objXMLFolder.ChildItems.Count;

 // step through associated xml files
 for(nIndex = 1;nIndex <= nCount;nIndex++)      {
```

```
    objChild = objXMLFolderChildItems.Item(nIndex);

    try {
     objDoc = objChild.Open();

     // use JScript method to access out-parameters
     var strError = new Array(1);
     var nErrorPos = new Array(1);
     var objBadData = new Array(1);

     bOK = objDoc.IsValid(strError,nErrorPos,objBadData);

     if(bOK){
      // if the validation fails, we should display the
      // message from XMLSpy
      // of course we have to create the form "MsgBox" and
      // define the gbbal txtMessage variable
      //
      // txtMessage = Position:" + nErrorPos[0] + "\n" + // strError[0];
      // txtMessage += "\n\nXML:\n" + objBadData[0].Name + "," +
      //    objBadData[0].TextValue;
      //
      // Application.ShowForm("MsgBox");

       break;
     }

     objDoc.Close(true);
     objDoc = null;
    }
    catch(Err) {
     // displaying the error description here is a good idea

     // txtMessage = Err.Description;
     // Application.ShowForm("MsgBox");

     break;
     }
    }
   }
  }
```

Testing the Event Handler

Switch to XMLSpy, and open a project to see how the `BeforeOpenProject` event is handled.

1.6 Macros

Macros automate repetitive or complex tasks. In the Scripting Environment, you can create a script that calls application functions as well as custom functions that you have defined. This flexibility provides you with a powerful method of automating tasks within XMLSpy. This section about macros is organized as follows:

- Creating and Editing a Macro describes how to create a new macro and edit an existing one.
- Running a Macro explains how a macro can be run from the Scripting Editor and from the broader XMLSpy environment as well.
- Debugging describes how macros can be debugged.

Key points about macros

Given below is a summary of important points about macros.

- Any number of macros can be added to the active scripting project. These macros are saved in the Altova Scripting Project file (`.asprj` file).
- Functions that are used in a macro can be saved as a Global Declaration. All Global Declarations are also saved in the Altova scripting project file (`.asprj` file).
- The macro can be tested by running it from within the Scripting Editor, and it can be debugged from within the Scripting Editor.
- XMLSpy can have one global Scripting Project, and a second scripting project, assigned to the currently loaded project, active at any one time; the macros are available to both of them. See Running a Macro for details.

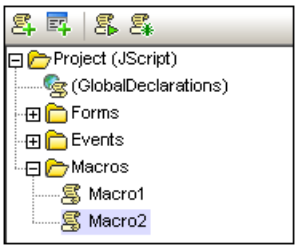
1.6.1 Creating and Editing a Macro

The following operations enable you to create a new macro and edit an existing macro.

Creating a new macro

Right-click the Macro folder in the Scripting Projects tree and select **Add Macro** from the context menu. (The **Add Macro** command can also be selected from the context menu of any item in the Scripting Projects tree.) Alternatively, click the **New Macro** icon in the toolbar of the Scripting Projects tree.

The newly created (and empty) macro document is displayed in the Main Window, and the name of the macro is displayed in the title bar of the Scripting Editor (*screenshot below*).



Naming or renaming a macro

To name or rename a macro, click the macro name in the Scripting Project tree and press the **F2** function key, or right click the name and select **Rename** from the context menu.

Opening a macro

To open a macro, right-click the macro in the Macros folder of the Scripting Project tree (*see screenshot above*), and select the **Open** command. The macro is displayed in the Main Window and its name is displayed in the title bar of the Scripting Editor (*screenshot below*). Alternatively, double-clicking a macro in the Scripting Project tree opens it in the Main Window.

Editing the macro

To edit a macro, enter or edit its code in the Main Window. For example, the following code creates the Form named `Form1` in memory and then shows it. `Form1` must already have been created (using the Scripting Editor's Form creation) before this macros is run.

```
objForm = CreateForm('Form1');
objForm.ShowDialog();
```

The following macro uses the `RemoveAllNamespaces` function to remove all namespaces in the active XML document.

```
if(Application.ActiveDocument != null)        {
                RemoveAllNamespaces(Application.ActiveDocument.RootElement);
                Application.ActiveDocument.UpdateViews();
}
```

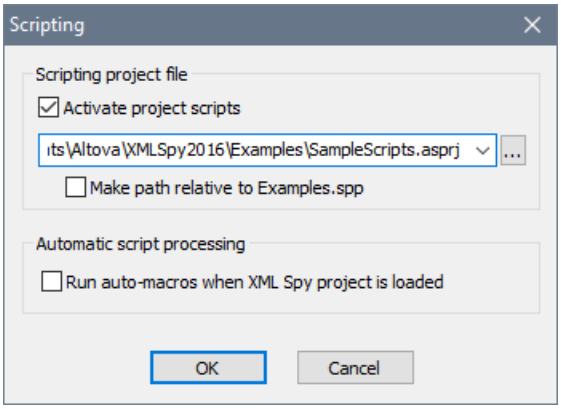
The `RemoveAllNamespaces` function itself will have to be defined in the Global Declarations script. After the `RemoveAllNamespaces` function has been defined, the macro is complete and can be run.

Note: Macros do not support parameters or return values.

Setting a macro as an Auto-Macro

When a macro is set as an Auto-Macro it can be run automatically when: (i) XMLSpy is started, or (ii) an Altova XMLSpy project is loaded in XMLSpy. To specify whether Auto-Macros should be run in each of these two events, check the *Run Auto-Macros* option in the Automatic Script Processing pane of the relevant dialogs:

- *When XMLSpy is started:* the Scripting tab of the XMLSpy Options dialog (**Tools | Options** menu command).
- *When an XMLSpy project is loaded into XMLSpy:* the Scripting dialog (*screenshot below*, **Project | Scripting Settings** menu command).



To set a macro as an Auto-Macro, right-click the macro in the Scripting Project tree and select

the command **Set as Auto-Macro**. This is a toggle command; so to remove the Auto-Macro setting of a macro, select the command again.

1.6.2 Running a Macro

To run a macro in the Scripting Editor, right-click the macro in the Scripting Project tree and select the command **Run Macro**.

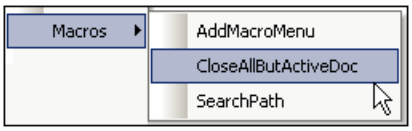
There are different ways to run a macro from XMLSpy:

- Via the Tools | Macros menu of XMLSpy.
- By creating and using a toolbar button for a macro.
- By creating and using a menu item for a macro.

Note that only one macro can be run at a time. After a macro (or event) is executed, the script is closed and global variables lose their values.

The XMLSpy command to run Macros

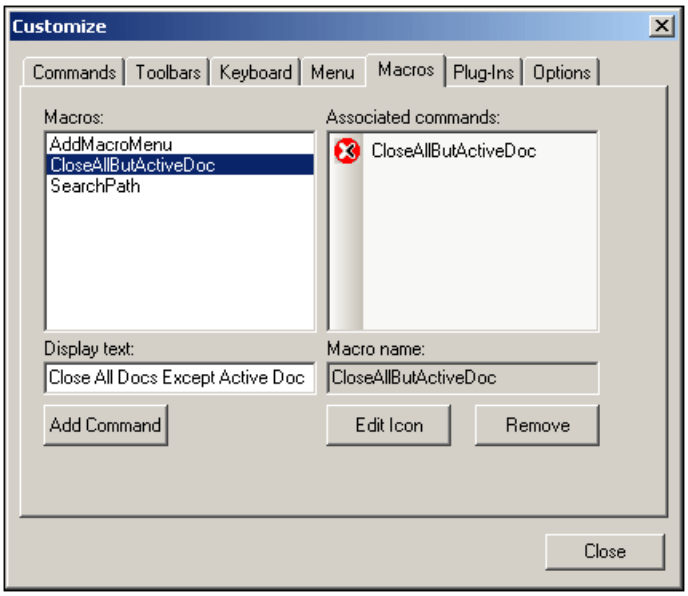
The **Tools | Macros** menu command (*screenshot below*) opens a submenu containing the macros defined in the Scripting Project that is currently active in XMLSpy. The active Scripting Projects are specified in the Scripting tab of the Options dialog, or in the Scripting tab of the project settings.



From the submenu of available macros, select the macro to run. The macro will be executed.

Toolbar icon

You can create an icon in the toolbar or a menu item that runs a selected macro. To do this, click **Tools | Customize | Macros**. This causes the Customize dialog to be displayed (*screenshot below*).



Now do the following:

1. In the Macros tab of the Customize dialog, select the required macro from the Macros pane. The macros in the Macros pane are those in the active Scripting Project (which is specified in the Scripting tab of the Options dialog).
2. In the *Display Text* input field enter the name of the icon. This name will appear when the cursor is placed over the icon when it is in the toolbar.
3. Click **Add Command** to add it to the list of commands.
4. Select the command and click **Edit Icon** to create a new icon.
5. Drag the finished icon from the *Associated Commands* pane and drop it on to the toolbar or menu when the cursor changes from an arrow to an I-beam or line.
6. Macros can even be assigned their own shortcuts in the Keyboard tab of the Customize dialog (*see screenshot above*).

To remove the toolbar icon, open the Macros tab of the Customize dialog and drag the icon out of the toolbar and into the *Associated Commands* pane. Select the command in the *Associated Commands* pane and click **Remove** to remove the command from the pane.

Item in the Tools menu

The XMLSpy API includes a function, `AddMacroMenuItem()`, to add macros as menu items to the **Tools** menu. This function can be used to add one or more macros to the **Tools | Macros** list of macros. Typically, you should do this as follows:

1. Add the macro menu item by calling the XMLSpy API function, `AddMacroMenuItem()`.

```
Application.AddMacroMenuItem("DeleteElements","Delete Elements
Dialog");
```

   - The function's first parameter (`DeleteElements` in the example listing above) is the name of the macro. If you run the macro and there is an open project having scripts associated with it, XMLSpy searches for the macro in the project scripts first. If there are no project scripts, or if XMLSpy cannot find the macro, then it looks for the macro in the global scripts.
   - The second parameter (`Delete Elements Dialog`) is the display text for the menu item.

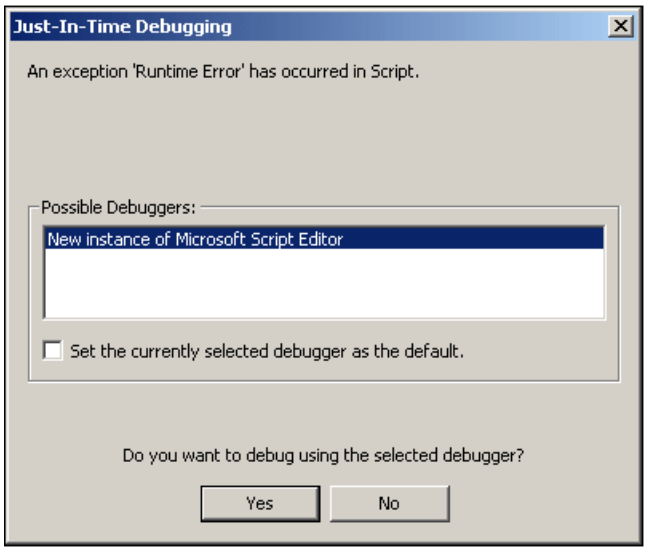
2. Reset the **Tools** menu by calling `ClearMacroMenu()`. This removes all previously added menu items

The best way to call these two functions is with the `Autorun` macro of the global scripting project or the `On_OpenProject` event.

1.6.3 Debugging a Macro

You can debug a macro using an installed debugger. To do this, right-click the macro in the Scripting Project tree and select the command **Debug Macro**.

This pops up the Just-In-Time Debugging dialog (*screenshot below*), which lists the debuggers available on the machine. Select the debugger you wish to use and click **Yes**.



The selected debugger starts.

1.7 Programming Points

The following programming points should be noted:

- All namespaces and types of the following .NET assemblies can be accessed in the Microsoft .NET Framework per default:

```
System
System.Data
System.Design
System.Drawing
System.Windows.Forms
System.XML
```

  Additional assemblies can be added to the scripting project via the project's context menu, or dynamically (at runtime) in the source code by using `CLR.LoadAssembly`.

- Out-parameters from methods of the XMLSpy API require special variables in JScript. Given below are some examples.

```
                // use JScript method to access out-parameters
var strError = new Array(1);
var nErrorPos = new Array(1);
var objBadData = new Array(1);
bOK = objDoc.IsValid(strError,nErrorPos,objBadData);END
```

- Out-parameters from methods of the.NET Framework require special variables in JScript. For example:

```
var dictionary = CLR.Create("System.Collections.Generic.Dictionary< System.String,
System.String >");
dictionary.Add("1","A");
dictionary.Add("2","B");

// use JScript method to access out-parameters
var strOut = new Array(1);
if (dictionary.TryGetValue("1",strOut)) // TryGetValue will set the out parameter
        alert(strOut[0]);// use out parameter
```

- .NET Methods that require integer arguments should not be called directly with JScript Number Objects which are Floating Point Values.

  For example, instead of:

```
var objCustomColor = CLR.Static( "System.Drawing.Color" ). FromArgb( 128,
128, 128 );
```

use:

```
var objCustomColor = CLR.Static( "System.Drawing.Color" ).
FromArgb( Math.floor( 128 ), Math.floor( 128 ), Math.floor( 128 ) );
```

- To iterate .NET collections the JScript Enumerator as well as the .NET iterator technologies can be used:

For example:

```
// iterate using the JScript iterator
var itr = new Enumerator(coll);
for (; !itr.atEnd(); itr.moveNext())
        alert(itr.item());

// iterate using the .NET iterator
var itrNET = coll.GetEnumerator();
while (itrNET.MoveNext())
        alert(itrNET.Current);
```

- .NET templates can be instantiated as shown below:

```
var coll = CLR.Create("System.Collections.Generic.List<System.String>");
```

or

```
CLR.Import("System");
CLR.Import("System.Collections.Generic");
var dictionary = CLR.Create("Dictionary<String,Dictionary<String,String>>");
```

- .NET Enum values are accessed as shown below:

```
var enumValStretch =
CLR.Static("System.Windows.Forms.ImageLayout").Stretch;
```

- Enumeration literals, as defined in the Altova type libraries, can now be used instead of numerical values.

```
objExportXMIFileDlg.XMIType = eXMI21ForUML23;
```

1.7.1 Built-in Commands

This section lists:

- Built-in commands
  alert
  conform
  doevents
  CreateForm
  lastform
  prompt
  ShowForm
  watchdog

- .NET interoperability commands
  CLR.Create
  CLR.Import
  CLR.LoadAssembly
  CLR.ShowImports
  CLR.ShowLoadedAssemblies
  CLR.Static

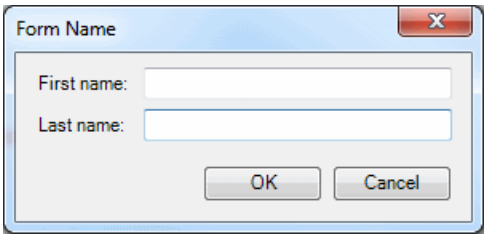
Built-in commands

The following built-in commands are available.

ShowForm(strFormName : String)
Instantiates a New Form object from the given form name and immediately shows it as Dialog.
Return Value: A Number that represents the generated DialogResult (System.Windows.Forms.DialogResult).

```
Example:
    var dialogResult = ShowForm("FormName");
```

Shows Form "FormName" as Dialog:



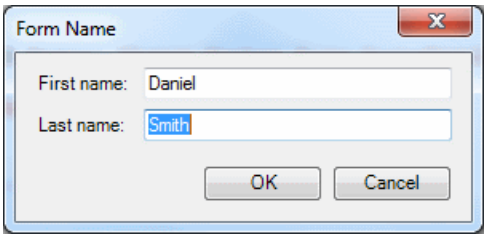
The DialogResult can be evaluated e.g. by:

```
if (dialogResult == CLR.Static("System.Windows.Forms.DialogResult").OK)
  alert("ok");
else
  alert("cancel");
```

CreateForm(strFormName : String)
Instantiates a New Form object from the given Form name.
Return Value: The Form object (System.Windows.Forms.Form) of the given name, or `null` if no Form with such name exists.

```
Example:
    var myForm = CreateForm("FormName");
    if (myForm != null)
    {
      myForm.textboxFirstName.Text = "Daniel";
      myForm.textboxLastName.Text = "Smith";
      var dialogResult = myForm.ShowDialog();
    .
    .
```

Shows Form "FormName" as Dialog - TextBoxes are initialized:



The DialogResult can be evaluated e.g. by:

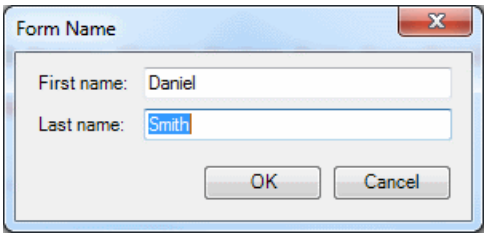
```
if (dialogResult == CLR.Static("System.Windows.Forms.DialogResult").OK)
  alert("ok");
else
  alert("cancel");
```

lastform
This global field can be used to conveniently access the last form object that was created.
Return Value: Returns a reference to the last form object (`System.Windows.Forms.Form`) that was successfully instantiated via `CreateForm()` or `ShowForm()`.

Example:

```
CreateForm("FormName");
if (lastform != null)
{
  lastform.textboxFirstName.Text = "Daniel";
  lastform.textboxLastName.Text = "Smith";
  var dialogResult = lastform.ShowDialog();
}
```

Shows Form "FormName" as Dialog - TextBoxes are initialized (similar to the CreateForm example above):



doevents()
Processes all Windows messages currently in the message queue.
Return Value: None

Example:

```
for (i=0;i< nLongLastingProcess;++i)
{
  // do long lasting process

  doevents();// process windows messages; give UI a chance to update
}
```

watchdog(bEnable : boolean)
Long running CPU-intensive scripts cause the watchdog to ask the user if the script should be terminated. The `watchdog()` method is used to disable or enable this behavior.
Per default the watchdog is enabled.
Return Value: None

Example:

```
watchdog(false),   // disable watchdog - we know the next statement is CPU intensive but it will
terminate for sure
doCPUIntensiveScript();
watchdog(true),    // re-enable watchdog
```

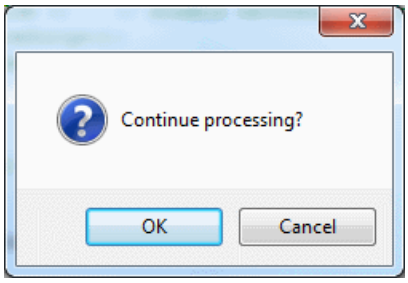
Usage tip:
Calling watchdog(true) can also be used to reset the watchdog. This can be useful before executing long running (CPU intensive) tasks to ensure they have the maximum allowed script processing quota.

alert(strMessage : String) or MsgBox(strMessage : String)
An alert box is used to show a given message. The user will have to click "OK" to proceed.
Return Value: None

```
Example:
      alert("Hello World");
```

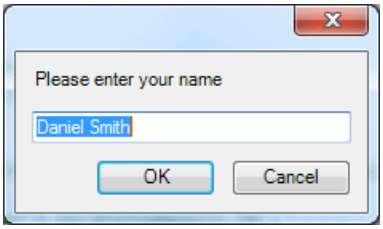
confirm(strMessage : String)
Opens a dialog that shows the given confirm message.
A confirm box is often used to verify or accept something. The user will have to click either "OK" or "Cancel" to proceed.
Return Value: A Boolean that represents the users answer. If the user clicks "OK", the dialog returns true, if the user clicks "Cancel", the dialog returns false.

```
Example:
      if (confirm("Continue processing?") == false)
        return;
```



prompt(strMessage : String, strDefault : String)
Opens a dialog that shows the given prompt message and a TextBox control with a default answer.
A prompt box is often used to input a simple string value.
Return Value: A String that contains the TextBox value or null if the user selected "Cancel".

```
Example:
      var name = prompt("Please enter your name","DanielSmith");
      if (name != null)
        alert("Hello " + name + "!");
```



.NET interoperability commands

To allow further interoperability with the .NET Framework additional functions are provided under CLR.

`CLR.Import(strNamespaceCLR : String)`

This is the scripting equivalent to the C# *using* / VB.Net *imports* keyword. This allows to leave out the given namespaces in successive calls like `CLR.Create()` and `CLR.Static()`.
Return Value: None

`Example:`

Instead of always having to use full qualified names:

```
if (ShowForm("FormName") == CLR.Static("System.Windows.Forms.DialogResult").OK)
{
  var sName = lastform.textboxFirstName.Text + " " + lastform.textboxLastName.Text;
  CLR.Static("System.Windows.Forms.MessageBox").Show("Hello " + sName);
}
```

one can import namespaces and use the short form:

```
CLR.Import("System.Windows.Forms");
if (ShowForm("FormName") == CLR.Static("DialogResult").OK)
{
  var sName = lastform.textboxFirstName.Text + " " + lastform.textboxLastName.Text;
  CLR.Static("MessageBox").Show("Hello " + sName);
}
```

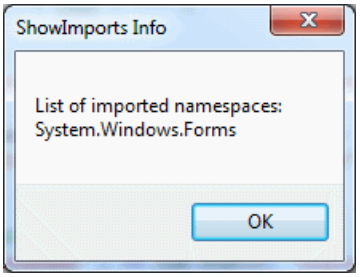
Please note:
Importing a namespace does not add or load the corresponding assembly to the scripting project! Assemblies can be added to the scripting project dynamically (at runtime) in the source code by using `CLR.LoadAssembly`.

`CLR.ShowImports()`

Opens a MessageBox dialog that shows the currently imported namespaces. The user will have to click "OK" to proceed.
Return Value: None

`Example:`

```
CLR.ShowImports();
```



CLR.LoadAssembly(strAssemblyNameCLR : String)
Loads the .NET assembly with the given long assembly name or file path.
Return Value: A Boolean value. True if the assembly could be loaded, false otherwise.

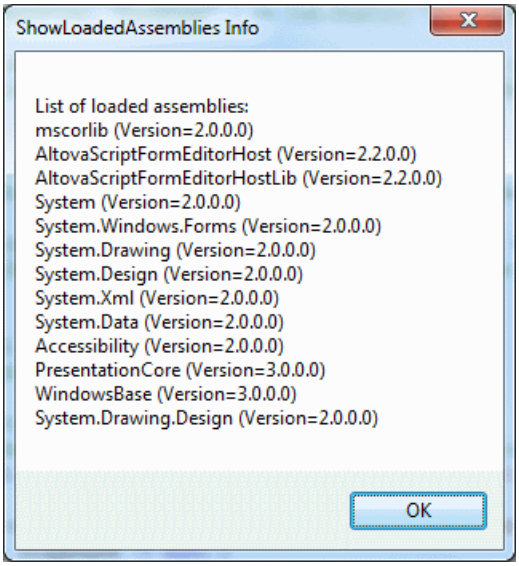
Example:

```
// set clipboard text (if possible)
// System.Windows.Clipboard is part of the PresentationCore assembly, so load this assembly first:
if (CLR.LoadAssembly("PresentationCore, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35", true))
{
  var clipboard = CLR.Static("System.Windows.Clipboard");
  if (clipboard != null)
          clipboard.SetText("HelloClipboard");
}
```

CLR.ShowLoadedAssemblies()
Opens a MessageBox dialog that shows the currently loaded assemblies. The user will have to click "OK" to proceed.
Return Value: None

Example:

```
CLR.ShowLoadedAssemblies();
```



CLR.Create(strTypeNameCLR : String, constructor arguments ...)
Creates a new .NET object instance for the given typename. If more than one argument is

passed the successive arguments are interpreted as the arguments for the constructor of the .NET object.
Return Value: A reference to the created .NET object

```
Examples:
      var objArray = CLR.Create("System.Collections.ArrayList");

      var newItem = CLR.Create("System.Windows.Forms.ListViewItem","NewItemText");

      var coll= CLR.Create("System.Collections.Generic.List<System.String>");

      CLR.Import("System");
      CLR.Import("System.Collections.Generic");
      var dictionary = CLR.Create("Dictionary<String,Dictionary<String,String>>");
```

CLR.Static(strTypeNameCLR : String)
Gives access to .NET types that have no instances and contain only static members.
Return Value: A reference to the static .NET object

```
Examples:
      var enumValStretch =
      CLR.Static( "System.Windows.Forms.ImageLayout" ).Stretch

      var clipboard = CLR.Static("System.Windows.Clipboard");
      clipboard.SetText("HelloClipboard");

      if (ShowForm("FormName") == CLR.Static("System.Windows.Forms.DialogResult").OK)
        alert("ok");
      else
        alert("cancel");
```

Form usage and commands

Form usage is as follows:

With Form objects, the Form Component Tree can be accessed naturally via field access:

For example, suppose there is a Form designed as follows:

```
MyForm
      ButtonPanel
            OkButton
            CancelButton
      TextEditor
      AxMediaPlayer1

TrayComponents:
      MyTimer
```

The Form can then be instantiated from script as:

```
var objForm = CreateForm("MyForm");
```

To access one its components the field access can be used:

```
objForm.ButtonPanel.OkButton.Enabled = false;
```

or

```
objForm.TextEditor.Text = "Hello World";
```

To access Tray Components use the following method on the Form object:

```
var objTrayComponent = <A form object>.GetTrayComponent(strComponentName
: String);
```

In our example to get a reference to the Timer Component to enable it use the following:

```
var objTimer = objForm.GetTrayComponent("MyTimer");
objTimer.Enabled = true;
```

For ActiveX Controls the underlying COM object can be accessed via the OCX property:

```
var ocx = lastform.AxMediaPlayer1.OCX; // get underlying COM object
ocx.enableContextMenu = true;
ocx.URL = "mms://apasf.apa.at/fm4_live_worldwide";
```

1.8 Migrating to Scripting Editor 2010 and Later

The Scripting Editor in XMLSpy from version 2010 onwards uses a different underlying technology than earlier versions used. Consequently, scripting projects that were created with versions of XMLSpy prior to version 2010 might need to be modified. The following points need to be noted.

- If a previous Scripting Projects (`.prj` file) is opened with the new Scripting Editor (version 2010 and later), the visual layout of Forms will be migrated as faithfully as possible and scripts will be copied as they are in the `.prj` file. You will then need to modify the scripts to be in accordance with the new technology used by the Scripting Editor, and which is described in this documentation.
- `TheView` object: The old Scripting Environment provided an artificial property named `TheView` that was only accessible from inside event handlers. It was used to access the Form that trigged the event (either directly or from one of its child controls). The new Scripting IDE does **not** provide this artificial property but instead provides the same functionality, and much more, with orthogonal built-in scripting helper functions combined with the power of the .NET framework.
- Since all event handlers in the new Scripting Environment get a sender object as a first parameter, the source that triggered the event is always available. By calling the .NET function `FindForm()` on the sender object one can access the Form object easily. Alternatively (if only one Form is involved) the built-in property `lastform` can be used. Note that the use of `lastform` is not constrained to event handlers (as was the case with `TheView`). It can be used everywhere in script code.

Given below is a list of methods and properties of the `TheView` object, each accompanied by an alternative mechanism offered by the new Scripting Environment.

Methods

The following methods were provided by the `TheView` object and must be migrated as explained:

`Cancel()`

In the new scripting environment the same can be achieved with: `lastform.Close(); // Use .NET Form.Close()`

`IsFormOpen(Name as String) as Boolean`

Since for .NET Forms there is a distinction between showing a Form and instantiating a Form, the previous concept does not directly translate. Instead the user can ask if a certain Form is currently shown. For example:

```
var objFormPencilSelector = CreateForm("PencilSelector");
var objFormColorSelector = CreateForm("ColorSelector");
      ...
// Anywhere in code ...

if(objFormColorSelector.Visible)
{
      ...
}
```

`FormFind(Name as String) as Object`

The new Scripting Environment allows you to instantiate more Forms of the same kind. In the old Scripting Environment each Form could only exist once (as a Singleton). Thus there is no equivalent of `FormFind()`. In the new Scripting Environment.

OpenDoc(File as String)
The same can be achieved with: `Application.OpenDocument( File as String )`

PumpData()
This corresponds to the built-in function `doevents()` which processes all Windows messages currently in the message queue.

RunClick(), RunInitialize(), RunTerminate()
There is no direct replacement for these methods. Call the corresponding handlers directly instead.

Properties

The following properties were provided by the `TheView` object and must be migrated as explained:

ToolTipText as String
To use tooltips in the new scripting environment, the .NET infrastructure can be used. This allows fine-grained control of tooltip behaviour (adjusting delays, when to show, etc). For example, to provide tooltips for a Form with two controls, the following code could to be added to the Form's `Load` event handler:

```
            //Occurs whenever the user loads the form.
            function MyForm_Load( objSender, e_EventArgs )
            {
                  // Create the ToolTip and associate with the Form
container.
                  var toolTip = CLR.Create("System.Windows.Forms.ToolTip");

                  // Set up the delays for the ToolTip.
                  toolTip.AutoPopDelay = 3000;
                  toolTip.InitialDelay = 1000;
                  toolTip.ReshowDelay = 500;

                  // Force the ToolTip text to be displayed whether or not
                  // the form is active.
                  toolTip.ShowAlways = true;

                  // Set up the ToolTip text for several Controls.
                  toolTip.SetToolTip(objSender.ProgressBar1,
                              "Shows the progress of the operation");
                  toolTip.SetToolTip(objSender.Button1,
                              "Click Button to start the processing");
            }
```

Color as Long
Since all Form/controls in the new Scripting Environment are .NET controls from the `System.Windows.Forms` namespace, the possibilities to modify colors, background image, fonts, and all other visual aspects are numerous. For example, every Visual Component has the properties `BackColor` and `ForeColor` to modify the visual appearance. The following handler could be used to change the color of a button at runtime:

```
      function TestForm_Button1_Click( objSender, e_EventArgs )
      {
            objSender.BackColor =
CLR.Static( "System.Drawing.Color" ).SlateBlue;
```

```
}
```

Please refer to the .NET documentation to find out more about this topic:
http://msdn.microsoft.com/en-us/library/system.windows.forms.aspx

2 IDE Plugins

XMLSpy allows you to create your own IDE plug-ins and integrate them into XMLSpy.

Use plug-ins to:

- Configure your version of XMLSpy, add commands through menus, icons, buttons etc.
- React to events from XMLSpy.
- Run your specific code within XMLSpy with access to the complete XMLSpy API

XMLSpy expects your plug-in to implement the IXMLSpyPlugIn interface. C# and C++ examples are included with your installation package and are located in the `XMLSpy2017\Examples\IDEPlugin` folder of your XMLSpy installation.

Windows XP	`C:/Documents and Settings/<username>/My Documents`
Windows Vista, Windows 7, 8, 10	`C:/Users/<username>/Documents`

See ATL sample files for an example using C++.

2.1 Registration of IDE PlugIns

XMLSpy maintains a specific key in the Registry where it stores all registered IDE plug-ins:

HKEY_CURRENT_USER\Software\Altova\XML Spy\PlugIns

All values of this key are treated as references to registered plug-ins and must conform to the following format:

Value name:	ProgID of the plug-in
Value type:	must be REG_SZ
Value data:	CLSID of the component

Each time the application starts the values of the "PlugIns" key is scanned, and the registered plug-ins are loaded.

Register plug-in manually

To register a plug-in manually, use the "Customize" dialog box of the XMLSpy "Tools" menu. Use the "Add Plug-In..." button to specify the DLL that implements your plug-in. XMLSpy registers the DLL as a COM server and adds the corresponding entry in its "PlugIns" key.

If you experience problems with manual registration you can check if the CLSID of your plug-in is correctly registered in the "PlugIns" key. If this is not the case, the name of your plug-in DLL was probably not sufficiently unique. Use a different name or perform direct registration.

Register plug-in directly

A plug-in can be directly registered as an IDE plug-in by first registering the DLL and then adding the appropriate value to the "PlugIns" key of XMLSpy during plug-in setup for example. The new plug-in will be activated the next time XMLSpy is launched.

Creating plug-ins

Source code for sample plug-ins has been provided in the application's (My) Documents folder: `Examples\IDEPlugin` folder. To build a plug-in from such source code, do the following:

1. Open the solution you want to build as a plug-in in Visual Studio.
2. Build the plug-in with the command in the Build menu.
3. The plug-in's DLL file that will be created in the `Bin` or `Debug` folder. This DLL file is the file that must be added as a plug-in (*see above*).

2.2 ActiveX Controls

ActiveX controls are supported. Any IDE PlugIn which is also an ActiveX control will be displayed in a Dialog Control Bar. A sample PlugIn that is also an ActiveX control is included in the `XMLSpyPlugInActiveX` folder in the `Examples` folder of your application folder.

2.3 Configuration XML

The IDE plug-in allows you to change the user interface (UI) of XMLSpy. This is done by describing each separate modification using an XML data stream. The XML configuration is passed to XMLSpy using the GetUIModifications method of the IXMLSpyPlugIn interface.

The XML file containing the UI modifications for the IDE PlugIn, must have the following structure:

```
<ConfigurationData>
 <ImageFile>path To image file</ImageFile>
 <Modifications>
  <Modification>
   ...
  </Modification>
  ...
 </Modifications>
</ConfigurationData>
```

You can define icons or toolbar buttons for the new menu items which are added to the UI of XMLSpy by the plug-in. The path to the file containing the images is set using the ImageFile element. Each image must be 16 x 16 pixels using max. 256 colors. The image references must be arranged from left to right in a single (<ImageFile>...) line. The rightmost image index value, is zero.

The Modifications element can have any number of Modification child elements. Each Modification element defines a specific change to the standard UI of XMLSpy. Starting with version 4.3, it is also possible to remove UI elements from XMLSpy.

Structure of Modification elements
All Modification elements consist of the following two child elements:

```
<Modification>
 <Action>Type of action</Action>
 <UIElement Type="type of UI element">
 </UIElement>
</Modification>
```

Valid values for the Action element are:

Add - to add the following UI element to XMLSpy
Hide - to hide the following UI element in XMLSpy
Remove - to remove the UI element from the "Commands" list box, in the customize dialog

You can combine values of the Action element e.g. "Hide Remove"

The UIElement element describes any new, or existing UI element for XMLSpy. Possible elements are currently: new toolbars, buttons, menus or menu items. The **type** attribute, defines which UI element is described by the XML element.

Common UIElement children
The ID and Name elements are valid for all different types of XML UIElement fragments. It is however possible, to ignore one of the values for a specific type of UIElement e.g. Name is ignored for a separator.

```
<ID></ID>
<Name></Name>
```

If UIElement describes an existing element of the UI, the value of the ID element is predefined by XMLSpy. Normally these ID values are not known to the public. If the XML fragment describes a new part of the UI, then the ID is arbitrary and the value should be less than 1000.
The Name element sets the textual value. Existing UI elements can be identified just by name, for e.g. menus and menu items with associated sub menus. For new UI elements, the Name element sets the caption e.g. the title of a toolbar, or text for a menu item.

Toolbars and Menus
To define a toolbar its necessary to specify the ID and/or the name of the toolbar. An existing toolbar can be specified using only the name, or by the ID if it is known. To create a **new** toolbar both values must be set. The **type** attribute must be equal to "ToolBar".

```
<UIElement Type="ToolBar">
 <ID>1</ID>
 <Name>TestPlugIn</Name>
</UIElement>
```

To specify an XMLSpy menu you need two parameters:

- The ID of the menu bar which contains the menu. If no XML documents are open in the main window, the menu bar ID is 128. If one or more XML documents are open, the menu bar ID is 129.
- The menu name. Menus do not have an associated ID value. The following example defines the "Edit" menu of the menu bar which is active, when at least one XML document is open:

```
<UIElement Type="Menu">
 <ID>129</ID>
 <Name>Edit</Name>
</UIElement>
```

An additional element is used if you want to create a new menu. The Place element defines the position of the new menu in the menu bar:

```
<UIElement Type="Menu">
 <ID>129</ID>
 <Name>PlugIn Menu</Name>
 <Place>12</Place>
</UIElement>
```

A value of -1 for the Place element sets the new button or menu item at the end of the menu or toolbar.

Commands
If you add a new command, through a toolbar button or a menu item, the UIElement fragment can contain any of these sub elements:

```
<MacroName></MacroName>
<Info></Info>
<ImageID></ImageID>
```

If MacroName is specified, XMLSpy searches for a macro with the same name in the scripting environment and executes it each time this command is processed. The Info element contains a short description string which is displayed in the status bar, when the mouse pointer is over the associated command (button or menu item). ImageID defines the index of the icon the external image file. Please note that all icons are stored in one image file.

To define a toolbar button create an UIElement with this structure:

```
<UIElement Type="ToolBarItem">
 <!--don't reuse localIDs even the commands do the same-->
 <ID>5</ID>
 <Name>Open file from repository...</Name>
 <!--Set Place To -1 If this is the first button To be inserted-->
 <Place>-1</Place>
 <ImageID>0</ImageID>
 <ToolBarID>1</ToolBarID>
 <!--instead of the toolbar ID the toolbar name could be used-->
 <ToolBarName>TestPlugIn</ToolBarName>
</UIElement>
```

Additional elements to declare a toolbar button are Place, ToolBarID and ToolBarName. ToolBarID and ToolBarName are used to identify the toolbar which contains the new or existing button. The textual value of ToolBarName is case sensitive. The (UIElement) **type** attribute must equal "ToolBarItem".

To define a menu item, the elements MenuID, Place and Parent are available in addition to the standard elements used to declare a command. MenuID can be either 128 or 129. Please see "Toolbars and Menus" for more information on these values.

The Parent element is used to identify the **menu** where the new menu entry should be inserted. As sub menu items have no unique Windows ID, we need some other way to identify the parent of the menu item.

The value of the Parent element is a path to the menu item.
The text value of the Parent element, must equal the **parent menu name** of the submenu, where the submenu name is separated by a colon. If the menu has no parent, because its not a submenu, add a colon to the beginning of the name. The **type** attribute must be set to "MenuItem". Example for an UIElement defining a menu item:

```
<UIElement Type="MenuItem">
 <!--the following element is a Localcommand ID-->
 <ID>3</ID>
 <Name>Open file from repository...</Name>
 <Place>-1</Place>
 <MenuID>129</MenuID>
 <Parent>:PlugIn Menu</Parent>
 <ImageID>0</ImageID>
</UIElement>
```

XMLSpy makes it possible to add toolbar separators and menus if the value of the ID element is set to 0.

2.4 ATL sample files

The following pages show how to create a simple XMLSpy IDE plug-in DLL using ATL. To build the DLL it is necessary to know about ATL, the wizards that generate new ATL objects, as well as MS VisualStudio.

To access the API the implementation imports the Type Library of XMLSpy. The code reads various properties and calls methods using the smart pointers provided by the #import statement.

In addition, the sample code uses the MFC class CString and the ATL conversion macros such as W2T.

At a glance the steps to create an ATL DLL are as follows:

1. Open VisualStudio and select "New..." from the "File" menu.
2. Select the "Projects" tab.
3. Select "ATL COM AppWizard" and type in a project name.
4. Select "Support for MFC" if you want to use MFC classes, or if you want to create a project for the sample code.

   Having created the project files you can add an ATL object to implement the IXMLSpyPlugIn interface:

1. Select "New ATL Object..." from the "Insert" menu.
2. Select "Simple Object" from the wizard and click "Next".
3. Type in a name for the object.
4. On the "Attributes" tab, select "Custom" for the type of interface, and disable Aggregation.

These steps produce the skeleton code for the implementation of the IDE plug-in interface. Please see the following pages on how to modify the code and achieve some basic functionality.

2.4.1 Interface description (IDL)

The IDL of the newly created ATL object contains a declaration for one COM interface.

- This interface declaration must be replaced by the declaration of IXMLSpyPlugIn as shown below.
- The IDL must also contain the definition of the SPYUpdateAction enumeration.
- Replace the generated default interface name, (created by the wizard) with "IXMLSpyPlugIn" in the coclass declaration. The IDL should then look something like the example code below:

Having created the ATL object, you then need to implement the IDE plug-in interface of XMLSpy.

```
import "oaidl.idl";
import "ocidl.idl";

// ----- please insert the folbwing bbck into your IDL file -----
 typedef enum {
  spyEnable = 1,
  spyDisable = 2,
  spyCheck = 4,
  spyUncheck = 8
 } SPYUpdateAction;

// ----- end insert bbck ----
```

```
// ----- E.g. Interface entry automatically generated by the ATL wizard -----
// [
//    object,
//    uuid(AB7CD86A-8145-429A-A1F3-270692E08AFC),

//    helpstring("IXMLSpyPlugIn Interface")
//    pointer_default(unique)
// ]
// interface IXMLSpyPlugIn : IUnknown
// {
// };

// ----- end automatically generated Interface Entry


// ----- replace the Interface Entry (shown above) generated for you by the ATL wizard, with the
following block -----

 [
  odl,
  uuid(88F2A622-4B7E-42CD-8D04-3C0E5389DD85),
  helpstring("IXMLSpyPlugIn Interface")
 ]
 interface IXMLSpyPlugIn : IUnknown
  {
   HRESULT _stdcallOnCommand([in] long nID, [in] IDispatch* pXMLSpy);

   HRESULT _stdcallOnUpdateCommand([in] long nID, [in] IDispatch* pXMLSpy, [out, retval]
SPYUpdateAction* pAction);

   HRESULT _stdcallOnEvent([in] long nEventID, [in] SAFEARRAY(VARIANT)* arrayParameters, [in]
IDispatch* pXMLSpy, [out, retval] VARIANT* pReturnValue);

   HRESULT _stdcallGetUIModifications([out, retval] BSTR* pModificationsXML);

   HRESULT _stdcallGetDescription([out, retval] BSTR* pDescription);
  };

// ----- end replace block -----


// ----- The code below is automatically generated by the ATL wizard and will look slightly different in your
case -----

 [
 uuid(24FE0D1B-3FC0-494E-B36E-1D4CE412B014),
 version(1.0),
 helpstring("XMLSpyIDEPlugInDLL 1.0 Type Library")
 ]
 library XMLSPYIDEPLUGINDLLLib
 {
 importlib("stdole32.tlb");
 importlib("stdole2.tlb");

 [
  uuid(3800E791-7F6B-4ACD-9E32-2AC184444501),
  helpstring("XMLSpyIDEPlugIn Class")
 ]
 coclass XMLSpyIDEPlugIn
 {
  [default] interface IXMLSpyPlugIn;  // ----- define IXMLSpyPlugIn as the default interface -----
 };
};
```

2.4.2 Class definition

In the class definition of the ATL object, several changes must be made. The class has to derive from IXMLSpyPlugIn, the "Interface Map" needs an entry for IXMLSpyPlugIn, and the methods of the IDE plug-in interface must be declared:

```
#ifndef _XMLSPYIDEPLUGIN_H_
#define _XMLSPYIDEPLUGIN_H_

#include "resource.h"     // main symbols

/////////////////////////////////////////////////////////////////////////////
// CXMLSpyIDEPlugIn
class ATL_NO_VTABLE CXMLSpyIDEPlugIn :
 public CComObjectRootEx<CComSingleThreadModel>,
 public CComCoClass<CXMLSpyIDEPlugIn, &CLSID_XMLSpyIDEPlugIn>,
 public IXMLSpyPlugIn
{
public:
 CXMLSpyIDEPlugIn()
 {
 }

DECLARE_REGISTRY_RESOURCEID(IDR_XMLSPYIDEPLUGIN)
DECLARE_NOT_AGGREGATABLE(CXMLSpyIDEPlugIn)

DECLARE_PROTECT_FINAL_CONSTRUCT()

BEGIN_COM_MAP(CXMLSpyIDEPlugIn)
 COM_INTERFACE_ENTRY(IXMLSpyPlugIn)
END_COM_MAP()

// IXMLSpyIDEPlugIn
public:
 virtual HRESULT _stdcall OnCommand(long nID, IDispatch* pXMLSpy);

 virtual HRESULT _stdcall OnUpdateCommand(long nID, IDispatch* pXMLSpy, SPYUpdateAction* pAction);

 virtual HRESULT _stdcall OnEvent(long nEventID, SAFEARRAY **arrayParameters, IDispatch* pXMLSpy,
VARIANT* pReturnValue);

 virtual HRESULT _stdcall GetUIModifications(BSTR* pModificationsXML);

 virtual HRESULT _stdcall GetDescription(BSTR* pDescription);
};

#endif //__XMLSPYIDEPLUGIN_H_
```

2.4.3 Implementation

The code below shows a simple implementation of an XMLSpy IDE plug-in. It adds a menu item and a separator (available with XMLSpy) to the Tools menu. Inside the OnUpdateCommand() method, the new command is only enabled when the active document is displayed using the Grid View. The command searches for the XML element which has the current focus, and opens any URL starting with "http://", from the textual value of the element.

```
/////////////////////////////////////////////////////////////////////////////
// CXMLSpyIDEPlugIn

#import "XMLSpy.tlb"
using namespace XMLSpyLib;

HRESULT CXMLSpyIDEPlugIn::OnCommand(long nID, IDispatch* pXMLSpy)
```

```
{
 USES_CONVERSION;

 if(nID == 1) {
  IApplicationPtr ipSpyApp;

  if(pXMLSpy) {
   if(SUCCEEDED(pXMLSpy->QueryInterface(__uuidof(IApplication),(void **)&ipSpyApp))) {
    IDocumentPtr ipDocPtr = ipSpyApp->ActiveDocument;

    // we assume that grid view is active
    if(ipDocPtr) {
     IGridViewPtr ipGridPtr = ipDocPtr->GridView;

     if(ipGridPtr){
      IXMLDataPtr  ipXMLData = ipGridPtr->CurrentFocus;

      CString strValue = W2T(ipXMLData->TextValue);

      if(!strValue.IsEmpty() && (strValue.Left(7) == _T("http://")))
       ::ShellExecute(NULL,_T("open"),W2T(ipXMLData->TextValue),NULL,NULL,SW_SHOWNORMAL);
     }
    }
   }
  }
 }

 return S_OK;
}


HRESULT CXMLSpyIDEPlugIn::OnUpdateCommand(long nID, IDispatch* pXMLSpy, SPYUpdateAction*
pAction)
{
 *pAction = spyDisable;

 if(nID == 1) {
  IApplicationPtr ipSpyApp;

  if(pXMLSpy) {
   if(SUCCEEDED(pXMLSpy->QueryInterface(__uuidof(IApplication),(void **)&ipSpyApp)))  {
    IDocumentPtr ipDocPtr = ipSpyApp->ActiveDocument;

    // only enable if grid view is active
    if((ipDocPtr != NULL) && (ipDocPtr->CurrentViewMode == spyViewGrid))
     *pAction = spyEnable;
   }
  }
 }

 return S_OK;
}


HRESULT CXMLSpyIDEPlugIn::OnEvent(long nEventID, SAFEARRAY **arrayParameters, IDispatch* pXMLSpy,
VARIANT* pReturnValue)
{
 return S_OK;
}

HRESULT CXMLSpyIDEPlugIn::GetUIModifications(BSTR* pModificationsXML)
{
 CComBSTR bstrMods = _T(" \
    <ConfigurationData> \
```

```
    <Modifications> ");
 // add "Open URL..." to Tools menu
 bstrMods.Append (_T(" \
      <Modification> \
       <Action>Add</Action> \
       <UIElement type=\"MenuItem\"> \
        <ID>1</ID> \
        <Name>Open URL...</Name> \
        <Place>0</Place> \
        <MenuID>129</MenuID> \
        <Parent>Tools</Parent> \
       </UIElement> \
      </Modification> "));
 // add Seperator to Tools menu
 bstrMods.Append (_T(" \
      <Modification> \
       <Action>Add</Action> \
       <UIElement type=\"MenuItem\"> \
        <ID>0</ID> \
        <Place>1</Place> \
        <MenuID>129</MenuID> \
        <Parent>Tools</Parent> \
       </UIElement> \
      </Modification> "));
 // finish modification description
 bstrMods.Append (_T(" \
    </Modifications> \
   </ConfigurationData>"));

 return bstrMods.CopyTo(pModificationsXML);
}


HRESULT CXMLSpyIDEPlugIn::GetDescription(BSTR* pDescription)
{
 CComBSTR bstrDescr = _T("ATL C++ XMLSpy IDE PlugIn,This PlugIn demonstrates the implementation of a
simple ATL DLL as a IDE PlugIn for XMLSpy.");
 return bstrDescr.CopyTo(pDescription);
}
```

2.5 IXMLSpyPlugIn

See also

Methods
OnCommand
OnUpdateCommand
OnEvent
GetUIModifications
GetDescription

Description
If a DLL is added to XMLSpy as an IDE plug-in, it is necessary that it registers a COM component that answers to an IXMLSpyPlugIn interface with the reserved uuid(88F2A622-4B7E-42CD-8D04-3C0E5389DD85), for it to be recognized as a plug-in.

2.5.1 OnCommand

See also

Declaration: OnCommand(*nID* as long, pXMLSpy as IDispatch)

Description
The OnCommand() method of the interface implementation, is called each time a command added by the the IDE plug-in (menu item or toolbar button) is processed. nID stores the command ID defined by the ID element of the respective UIElement.

pXMLSpy holds a reference to the dispatch interface of the Application object of XMLSpy.

Example

```
Public Sub IXMLSpyPlugIn_OnCommand(ByVal nID As Long, ByVal pXMLSpy As Object)
   If (Not (pXMLSpy Is Nothing)) Then
      Dim objDlg
      Dim objDoc As XMLSpyLib.Document
      Dim objSpy As XMLSpyLib.Application
      Set objSpy = pXMLSpy

      If nID = 3 Or nID = 5 Then
         Set objDlg = CreateObject("MSComDlg.CommonDialog")
         objDlg.Filter = "XML Files (*.xml)|*.xml|All Files (*.*)|*.*||"
         objDlg.FilterIndex = 1
         objDlg.ShowOpen

         If Len(objDlg.FileName) > 0 Then
            Set objDoc = objSpy.Documents.OpenFile(objDlg.FileName, False)
            Set objDoc = Nothing
         End If
      End If

      If nID = 4 Or nID = 6 Then
         Set objDlg = CreateObject("MSComDlg.CommonDialog")
         objDlg.Filter = "All Files (*.*)|*.*||"
         objDlg.Flags = cdlOFNPathMustExist
         objDlg.ShowSave

         If Len(objDlg.FileName) > 0 Then
            Set objDoc = objSpy.ActiveDocument

            If Not (objDoc Is Nothing) Then
               objDoc.SetPathName objDlg.FileName
               objDoc.Save
               Set objDoc = Nothing
            End If
```

```
      End If
    End If

    Set objSpy = Nothing
  End If
End Sub
```

2.5.2 OnUpdateCommand

See also

Declaration: `OnUpdateCommand(nID as long, pXMLSpy as IDispatch) as SPYUpdateAction`

Description

The `OnUpdateCommand()` method is called each time the visible state of a button or menu item needs to be set. `nID` stores the command ID defined by the `ID` element of the respective `UIElement`.

`pXMLSpy` holds a reference to the dispatch interface of the `Application` object.

Possible return values to set the update state are:

```
spyEnable       = 1
spyDisable    = 2
spyCheck        = 4
spyUncheck      = 8
```

Example

```
Public Function IXMLSpyPlugIn_OnUpdateCommand(ByVal nID As Long, ByVal pXMLSpy As Object) As
SPYUpdateAction
  IXMLSpyPlugIn_OnUpdateCommand = spyDisable

  If (Not (pXMLSpy Is Nothing)) Then
    Dim objSpy As XMLSpyLib.Application
    Set objSpy = pXMLSpy

    If nID = 3 Or nID = 5 Then
      IXMLSpyPlugIn_OnUpdateCommand = spyEnable
    End If
    If nID = 4 Or nID = 6 Then
      If objSpy.Documents.Count > 0 Then
        IXMLSpyPlugIn_OnUpdateCommand = spyEnable
      Else
        IXMLSpyPlugIn_OnUpdateCommand = spyDisable
      End If
    End If
  End If
End Function
```

2.5.3 OnEvent

See also

Declaration: `OnEvent(nEventID as long, arrayParameters as SAFEARRAY(VARIANT), pXMLSpy as IDispatch) as VARIANT`

Description

`OnEvent()` is called each time an event is raised from XMLSpy.

Possible values for `nEventID` are:

```
On_BeforeStartEditing          = 1
```

On_EditingFinished	= 2
On_FocusChanged	= 3
On_Beforedrag	= 4
On_BeforeDrop	= 5
On_OpenProject	= 6
On_OpenDocument	= 7
On_CloseDocument	= 8
On_SaveDocument	= 9

Events available since XMLSpy 4r4:

On_DocEditDragOver	= 10
On_DocEditDrop	= 11
On_DocEditKeyDown	= 12
On_DocEditKeyUp	= 13
On_DocEditKeyPressed	= 14
On_DocEditMouseMove	= 15
On_DocEditButtonUp	= 16
On_DocEditButtonDown	= 17
On_DocEditContextMenu	= 18
On_DocEditPaste	= 19
On_DocEditCut	= 20
On_DocEditCopy	= 21
On_DocEditClear	= 22
On_DocEditSelectionChanged	= 23

Events available since XMLSpy 2004:

On_DocEditDragOver	= 10

Events available since XMLSpy 2004r4 (type library version 1.4):

On_BeforeOpenProject	= 25
On_BeforeOpenDocument	= 26
On_BeforeSaveDocument	= 27
On_BeforeCloseDocument	= 28
On_ViewActivation	= 29
On_DocEditKeyboardEvent	= 30
On_DocEditMouseEvent	= 31

Events available since XMLSpy 2006 SP1 (type library version 1.5):

On_BeforeValidate	= 32

Events available since XMLSpy 2007 (type library version 1.6):

On_BeforeShowSuggestions	= 33
On_ProjectOpened	= 34
On_Char	= 35

Events available since XMLSpy 2009 (type library version 2.2):

On_Initialize	= 36
On_Running	= 37
On_Shutdown	= 38

Events available since XMLSpy 2012 (type library version 2.8):

On_AuthenticBeforeSave	= 39
On_AuthenticContextMenuActivated	= 40
On_AuthenticLoad	= 41
On_AuthenticToolbarButtonClicked	= 42
On_AuthenticToolbarButtonExecuted	= 43
On_AuthenticUserAddedXMLNode	= 44

The names of the events are the same as they appear in the Scripting Environment of XMLSpy. For IDE plug-ins the names used are immaterial. The events are identified using the `ID` value.

`arrayParameters` is an array which is filled with the parameters of the currently raised event. Order, type and meaning of the single parameters are available through the scripting environment of XMLSpy. The events module of a scripting project, contains predefined functions for all events prior to version 4.4. The parameters passed to the predefined functions are identical to the array elements of the `arrayParameters` parameter.

Events raised from the Authentic View of XMLSpy do not pass any parameters directly. An "event" object is used instead. The event object can be accessed through the Document object of the active document.

`pXMLSpy` holds a reference to the dispatch interface of the `Application` object of XMLSpy.

If the return value of `OnEvent()` is set, then neither the IDE plug-in, nor an event handler inside of

the scripting environment will get this event afterwards. Please note that all IDE plug-ins get/process the event before the Scripting Environment does.

2.5.4 GetUIModifications

See also

Declaration: `GetUIModifications()` as String

Description

The `GetUIModifications()` method is called during initialization of the plug-in, to get the configuration XML data that defines the changes to the UI of XMLSpy. The method is called when the plug-in is loaded for the first time, and at every start of XMLSpy.

See also Configuration XML for a detailed description how to change the UI.

Example

```
Public Function IXMLSpyPlugIn_GetUIModifications() As String
   'GetUIModifications() gets the XML file with the specified modifications of
   'the UI from the config.xml file in the plug-in folder
   Dim strPath As String
   strPath = App.Path

   If Len(strPath) > 0 Then
      Dim fso As New FileSystemObject
      Dim file As file

      Set file = fso.GetFile(strPath & "\config.xml")

      If (Not (file Is Nothing)) Then
         Dim stream As TextStream
         Set stream = file.OpenAsTextStream(ForReading)

         'this replaces the token "**path**" from the XML file with
         'the actual installation path of the plug-in to get the image file
         Dim strMods As String
         strMods = stream.ReadAll
         strMods = Replace(strMods, "**path**", strPath)

         IXMLSpyPlugIn_GetUIModifications = strMods
      Else
         IXMLSpyPlugIn_GetUIModifications = ""
      End If
   End If
End Function
```

2.5.5 GetDescription

See also

Declaration: `GetDescription()` as String

Description

`GetDescription()` is used to define the description string for the plug-in entries visible in the Customize dialog box.

Example

```
Public Function IXMLSpyPlugIn_GetDescription() As String
   IXMLSpyPlugIn_GetDescription = "Sample Plug-in for XMLSpy: This Plug-in demonstrates the
implementation of a simple VisualBasic DLL as a Plug-in for XMLSpy."
End Function
```

3 Application API

The COM-based API of XMLSpy (also called the Application API from now on) enables other applications to use the functionality of XMLSpy. As a result, it is possible to automate a wide range of tasks, from validating an XML file to modifying complex XML content (with the XMLData interface).

XMLSpy and its Application API follow the common specifications for automation servers set out by Microsoft. It is possible to access the methods and properties of the Application API from common development environments, such as those using C, C++, VisualBasic, and Delphi, and with scripting languages like JScript and VBScript.

Execution environments for the Application API

The Application API can be accessed from the following execution environments:

- External programs (described below and in the Overview part of this section)
- From within the built-in Scripting Editor of XMLSpy. For a description of the scripting environment, see the section, Scripting Editor.
- XMLSpy allows you to create and integrate your own plug-ins into the application using a special interface for plug-ins. A description of how to create plug-ins is given in the section IDE Plug-ins.
- Via an ActiveX Control, which is available if the integration package is installed. For more information, see the section ActiveX Integration.

External programs

In the Overview part of this section, we describe how the functionality of XMLSpy can be accessed and automated from external programs.

Using the Application API from outside XMLSpy requires an instance of XMLSpy to be started first. How this is done depends on the programming language used. See the section, Programming Languages, for information about individual languages.

Essentially, XMLSpy will be started via its COM registration. Then the `Application` object associated with the XMLSpy instance is returned. Depending on the COM settings, an object associated with an already running XMLSpy can be returned. Any programming language that supports creation and invocation of COM objects can be used. The most common of these are listed below.

- JScript and VBScript script files have a simple syntax and are designed to access COM objects. They can be run directly from a DOS command line or with a double click on Windows Explorer. They are best used for simple automation tasks.
- C# is a a full-fledged programming language that has a wide range of existing functionality. Access to COM objects can be automatically wrapped using C#..
- C++ provides direct control over COM access but requires relatively larger amounts of code than the other languages.
- Java: Altova products come with native Java classes that wrap the Application API and provide a full Java look-and-feel.
- Other programming languages that make useful alternatives are: Visual Basic for Applications, Perl, and Python.

Programming points

The following limitations must be considered in your client code:

- Be aware that if your client code crashes, instances of XMLSpy may still remain in the system.
- Don't hold references to objects in memory longer than you need them, especially those from the `XMLData` interface. If the user interacts between two calls of your client, then there is no guarantee that these references are still valid.
- Don't forget to disable dialogs if the user interface is not visible.
- See Error handling in JScript (and in C# and Java) for details of how to avoid annoying error messages.
- Free references explicitly if you are using C or C++.

This documentation

This documentation section about the Application API is broadly divided into two parts.

- The first part consists of an Overview, which describes the object model for the API and explains how the API is accessed via various programming languages.
- The second part is a reference section (Interfaces and Enumerations) that contains descriptions of the interface objects of the Application API.

3.1 Overview

This overview of the Application API is organized as follows:

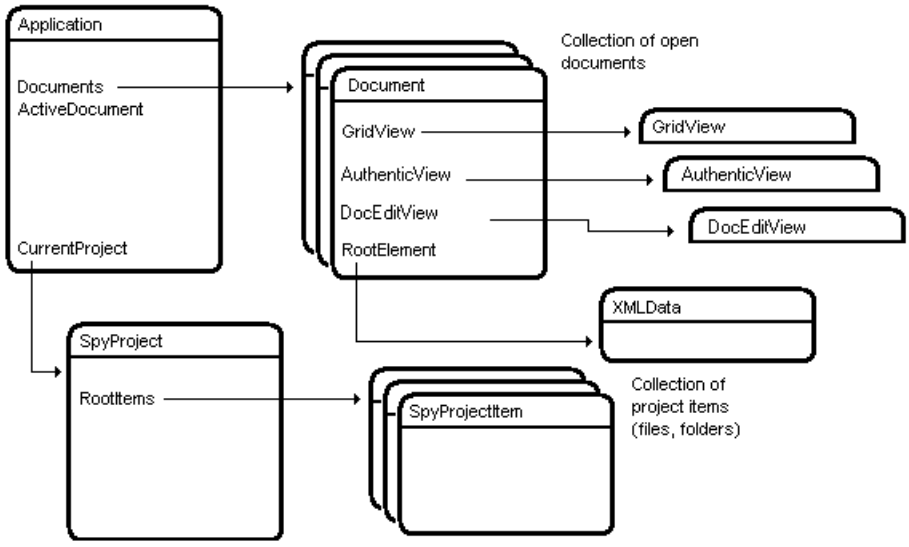
- The Object Model describes the relationships between the objects of the Application API.
- Programming Languages explains how the most commonly used programming languages (JScript, VBScript, C#, and Java) can be used to access the functionality of the Application API. Code listings from the example files supplied with your application package are used to describe basic mechanisms.
- The DOM and XMLData explains the relationship between the Application API's `XMLData` interface and the DOM.
- Obsolete: Authentic View Row Operations supplies information about obsolete objects for Authentic View table row operations.
- Obsolete: Authentic View Editing Operations supplies information about obsolete objects for Authentic View editing operations.

3.1.1 Object Model

The starting point for every application which uses the Application API is the `Application` object. This object contains general methods like import/export support and references to the open documents and any open project.

The `Application` object is created differently in various programming languages. In scripting languages such as JScript or VBScript, this involves calling a function which initializes the application's COM object. For examples, see the Programming Languages section.

The picture below shows the links between the main objects of the Application API:



The application object consists of the following parts:

1. Document collection and reference to the active document.
2. Reference to current project and methods for creating and opening projects.
3. Methods to support the export to and import from databases, text files, and Word documents.
4. URL management.
5. Methods for macro menu items.

Once you have created an `Application` object you can start using the functionality of XMLSpy. In most cases, you either open a project and access the documents from there or you directly open a document via the `Documents` interface.

3.1.2 Programming Languages

Programming languages differ in the way they support COM access. A few examples for the most frequently used languages (*links below*) will help you get started. The code listings in this section show how basic functionality can be accessed. This basic functionality is included in the files in the API Examples folder and can be tested straight away. The path to the API Examples folder is given below:

Windows 10, Windows 8, Windows 7, Windows Vista	`C:\Users\<username>\Documents\Altova\XMLSpy2017\Examples\API\`
Windows XP	`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\API\`

JScript

The JScript listings demonstrate the following basic functionality:

- Start application or attach to a running instance
- Simple document access
- Iteration
- Error handling
- Events
- Import and export of data

VBScript

VBScript is different than JScript only syntactically; otherwise it works in the same way. The listings below describe is an example of how VBScript can be used. For more information, refer to the JScript examples.

- Events: Shows how events are handled using VBScript.

C#

C# can be used to access the Application API functionality. The code listings show how to access the API for certain basic functionality.

- Start XMLSpy: Starts XMLSpy, which is registered as an automation server, or activates the program if XMLSpy is already running.
- Open OrgChart.pxf: Locates one of the example documents installed with XMLSpy and opens it. If this document is already open it becomes the active document.
- OnDocumentOpened Event On/Off: Shows how to listen to XMLSpy events. When turned on, a message box will pop up after a document has been opened.
- Open ExpReport.xml: Opens another example document.
- Toggle View Mode: Changes the view of all open documents between Text View and

Authentic View. The code shows how to iterate through open documents.
- Validate: Validates the active document and shows the result in a message box. The code shows how to handle errors and COM output parameters.
- Shutdown XMLSpy: Stops XMLSpy.

Java

The XMLSpy API can be accessed from Java code. The Java sub-section of this section explains how some basic XMLSpy functionality can be accessed from Java code. It is organized into the following sub-sections:

- Mapping Rules for the Java Wrapper
- Example Java Project
- Application Startup and Shutdown
- Simple Document Access
- Iterations
- Use of Out-Parameters
- Event Handlers

JScript

This section contains listings of JScript code that demonstrate the following basic functionality:

- Start application or attach to a running instance
- Simple document access
- Iteration
- Error handling
- Events
- Import and export of data
-

Example files

The code listings in this section are available in example files that you can test as is or modify to suit your needs. The JScript example files are located in the `JScript` folder of the API Examples folder:

Windows 10, Windows 8, Windows 7, Windows Vista	`C:\Users\<username>\Documents\Altova\XMLSpy2017\Examples\API\`
Windows XP	`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\API\`

The example files can be run in one of two ways:

- *From the command line:* Open a command prompt window, change the directory to the path above, and type the name of one of the example scripts (for example, `Start.js`).
- *From Windows Explorer:* In Windows Explorer, browse for the JScript file and double-click it.

The script is executed by Windows Script Host that is packaged with Windows operating system. For more information about Windows Script Host, refer to MSDN documentation (https://msdn.microsoft.com).

Start Application

The JScript below starts the application and shuts it down. If the COM object of the 32-bit XMLSpy cannot be found, the code attempts to get the COM object of the 64-bit application; otherwise, an error is thrown. If an instance of the application is already running, the running instance will be returned.

Note: For 32-bit XMLSpy, the registered name, or programmatic identifier (ProgId) of the COM object is `XMLSpy.Application`. For 64-bit XMLSpy, the name is `XMLSpy_x64.Application`.

```
// Initialize application's COM object. This will start a new instance of the
application and
// return its main COM object. Depending on COM settings, the main COM object of an
already
// running application might be returned.

try {   objSpy = WScript.GetObject("", "XMLSpy.Application");   }
catch(err) {}

if( typeof( objSpy ) == "undefined" )
{
   try   {   objSpy = WScript.GetObject("", "XMLSpy_x64.Application")   }
   catch(err)
   {
      WScript.Echo( "Can't access or create XMLSpy.Application" );
      WScript.Quit();
   }
}

// if newly started, the application will start without its UI visible. Set it to
visible.
objSpy.Visible = true;

WScript.Echo(objSpy.Edition + " has successfully started. ");

objSpy.Visible = false;      // will shutdown application if it has no more COM
connections
//objSpy.Visible = true;   // will keep application running with UI visible
```

The JScript code listed above is available in the sample file `Start.js` (see Example Files).

Simple Document Access

After you have started the application as shown in Start Application, you will most likely want to programmatically open a document in order to work with it. The JScript code listing below illustrates how to open two documents from the XMLSpy Examples folder and set one of them as the active document.

```
// Locate examples via USERPROFILE shell variable. The path needs to be adapted to
major release versions.
objWshShell = WScript.CreateObject("WScript.Shell");
majorVersionYear = objSpy.MajorVersion + 1998
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\My
Documents\\Altova\\XMLSpy" + majorVersionYear + "\\Examples\\";

// Tell XMLSpy to open two documents. No dialogs
objDoc1 = objSpy.Documents.OpenFile(strExampleFolder + "OrgChart.pxf", false);
```

```
objSpy.Documents.OpenFile(strExampleFolder + "ExpReport.xml", false);

// The document currently active can be easily located.
objDoc2 = objSpy.ActiveDocument;

// Let us make sure that the document is shown in grid view.
objDoc2.SwitchViewMode(0);   // SPYViewModes.spyViewGrid = 0

// Now switch back to the document opened first
objDoc1.SetActiveDocument();
```

The JScript code listed above is available in the sample file `DocumentAccess.js` (see Example Files).

Iteration

The JScript listing below shows how to iterate through the open documents. It is assumed that you have already started the application and opened some documents as shown in the previous sections.

```
// go through all open documents using a JScript Enumerator
bRequiresSaving = false;
for (var iterDocs = new Enumerator(objSpy.Documents); !iterDocs.atEnd();
iterDocs.moveNext())
{
   if (iterDocs.item().IsModified)
      bRequiresSaving = true;

   var strErrorText = new Array(1);
   var nErrorNumber = new Array(1);
   var errorData = new Array(1);

   if (!iterDocs.item().IsValid(strErrorText, nErrorNumber, errorData))
   {
       var text = strErrorText;
      // access that XMLData object only if filled in
       if (errorData[0] != null)
         text += "(" + errorData[0].Name + "/" + errorData[0].TextValue + ")";

      WScript.Echo("Document \"" + iterDocs.item().Name +"\" validation error[" +
nErrorNumber + "]: " + text);
   }
   else
   {
      // The COM call succeeded and the document is valid.
      WScript.Echo("Document \"" + iterDocs.item().Name + "\" is valid.");
   }
}

// go through all open documents using index-based access to the document collection
for (i = objSpy.Documents.Count; i > 0; i--)
   objSpy.Documents.Item(i).Close(false);
```

The JScript code listed above is available in the sample file `DocumentAccess.js` (see Example Files).

Error Handling

The Application API returns errors in two different ways:

- The `HRESULT` returned by every API method

- The IErrorInfo interface of the Application API

Every API method returns an HRESULT. This return value gives the caller information about errors during execution of the method. If the call was successful, the return value is S_OK. The HRESULT option is commonly used in C/C++ programs.

However, programming languages such as VisualBasic and scripting languages (and other high-level development environments) don't give the programmer access to the HRESULT return of a COM call. Such languages use the IErrorInfo interface, which is also supported by the Application API. If an error occurs, the Application API creates a new object that implements the IErrorInfo interface. The information provided by the IErrorInfo interface is imported by the development environment into its own error-handling mechanism.

For example, the JScript code listing below causes an error to be thrown by incorrectly declaring an array. Additional information about the error object is provided by its properties number and description.

```
try {
   var arr = new Array(-1);
}
catch (err) {
   WScript.Echo("Error : (" + (err.number & 0xffff) + ")" +
err.description);
}
```

Events

COM specifies that a client must register itself at a server for callbacks using the connection point mechanism. The automation interface for XMLSpy defines the necessary event interfaces. The way to connect to those events depends on the programming language you use in your client. The following code listing shows how this is done using JScript.

The method WScript.ConnectObject is used to receive events.

```
// The event-handler function
function DocEvent_OnBeforeCloseDocument(objDocument)
{
   WScript.Echo("Received event - before closing document");
}

// Create or connect to XMLSpy (or Authentic Desktop)
try
{
   // Create the environment and XMLSpy (or Authentic Desktop)
   objWshShell = WScript.CreateObject("WScript.Shell");
   objFSO = WScript.CreateObject("Scripting.FileSystemObject");
   objSpy = WScript.GetObject("", "XMLSpy.Application");

// If only Authentic Desktop is installed (and XMLSpy is not installed) use:
// objSpy = WScript.GetObject("", "AuthenticDesktop.Application")

}
catch(err)
   { WScript.Echo ("Can't create WScript.Shell object or XMLSpy"); }

// Create document object and connect to its events
objSpy.Visible = true;
majorVersionYear = objSpy.MajorVersion + 1998
docPath = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\Documents\
```

```
\Altova\\XMLSpy" + majorVersionYear + "\\Examples\\ExpReport.xml";
objDoc = objSpy.Documents.OpenFile (docPath, false);
WScript.ConnectObject(objDoc, "DocEvent_");

// Keep running while waiting for the event
// In the meanwhile close this document in XMLSpy (or Authentic Desktop) manually
WScript.Echo ("Sleeping for 10 seconds ...");
WScript.Sleep (10000);

objDoc = null;
WScript.Echo ("Stopped listening for event");
objSpy.Quit();
```

Import and Export of Data

Before you implement your import and export tasks with the Application API, it is good practice to test the connections, parameters, SQL queries and so on in XMLSpy. In this way you are able to verify the results and make quick adjustments to import or export parameters.

Most of the methods for importing and exporting data are placed in the `Application` object, the remaining functions are accessible via the `Document` interface.

There is some preparatory work necessary, before the actual import or export can be started. Every import/export job consists of two parts. You need to define a connection to your data and the specific behaviour for the import/export process.

In case of an import, the connection is either a database, a text-file or a Word document. The behaviour is basically which data (columns) should be imported in XMLSpy.

In case of an export, the connection is either a database or a text file. Specify which data (elements of the XML file) and additional parameters (for example, automatic key generation or number of sub-levels) to use from the XML-structure.

The properties in the `DatabaseConnection`, `TextImportExportSettings` and `ExportSettings` interfaces have default values. See the corresponding descriptions in the `Interfaces` chapter for further information.

The sub-sections of this section describe each of these operations in detail.

- Import from Database
- Export to Database
- Import from Text
- Export to Text

Given below are the steps to establish a connection to an existing database for import:

1. Use a `DatabaseConnection` object and set the properties:
   The method `Application.GetDatabaseSettings` returns a new object for a database connection:

   ```
   objImpSettings = objSpy.GetDatabaseSettings();
   ```

   You have to set either an **ADO connection string**,
   - `objImpSettings.ADOConnection = strADOConnection`

   or the **path** to an existing database file:
   - `objImpSettings.File = strExampleFolder + "Tutorial\\Company.mdb";`

To complete the settings you create a **SQL select statement** to define the data to be queried:

- `objImpSettings.SQLSelect = "SELECT * FROM Address";`

2. Call `Application.GetDatabaseImportElementList` to get a collection of the resulting columns of the SQL query:

   `objElementList = objSpy.GetDatabaseImportElementList(objImpSettings);`

   This collection gives you the opportunity to control which columns should be imported and specify the datatype of the new elements. Each item of the collection represents one column to import. If you remove an item, the corresponding column will not be imported. You can additionally modify the `ElementListItem.ElementKind` property to set the datatype of the XML elements for each column.

   Please consider that `GetDatabaseImportElementList()` executes the SQL query and could initiate a time-consuming call. To avoid this, it is possible to pass a null-pointer as the second parameter to `ImportFromDatabase()`; this imports all columns as plain XML elements.

3. Start the import with `Application.ImportFromDatabase`:

`objImpDocFromDB = objSpy.ImportFromDatabase(objImpSettings,objElementList);`

```
// Locate examples via USERPROFILE shell variable.
objWshShell = WScript.CreateObject("WScript.Shell");
majorVersionYear = objSpy.MajorVersion + 1998
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\My
Documents\\Altova\\XMLSpy" + majorVersionYear + "\\Examples\\";

try
{
   // specify the source of data import
   objImpSettings = objSpy.GetDatabaseSettings();
   objImpSettings.File = strExampleFolder + "Tutorial\\Company.mdb";
   objImpSettings.SQLSelect = "SELECT * FROM Address";

   // column filter
   objElementList = objSpy.GetDatabaseImportElementList(objImpSettings);

   // import into a new XML file
   objImpDocFromDB = objSpy.ImportFromDatabase(objImpSettings,objElementList);
}
catch(err)
{
   WScript.Echo("Error importing from database.\n\n" +
                "Error: " + (err.number & 0xffff) + "\n" +
                "Description: " + err.description);
}
```

The JScript code listed above is available in the sample file `ImportExport.js` (see Example Files).

To export data to a database, carry out the steps below:

1. Use a `DatabaseConnection` object and set the necessary properties.
   All properties except `SQLSelect` are important for the export. `ADOConnection` or `File` defines the target for the output. You need to set only one of them.

2. Fill an ExportSettings object with the required values.
   These properties are the same options as those available in the export dialog of XMLSpy. Select the menu option **Convert | Export to Text files/Database** to see the options and try a combination of export settings. After that it is easy to transfer these settings to the properties of the interface.

   Call Application.GetExportSettings to get an ExportSettings object:

   ```
   objExpSettings = objSpy.GetExportSettings()
   ```

3. Build an element list with Document.GetExportElementList.
   The element list enables you to eliminate XML elements from the export process. It also gives you information about the record and field count in the RecordCount and FieldCount properties. Set the ExportSettings.ElementList property to this collection. It is possible to set the element list to null/Nothing (default) to export all elements.

4. Call Document.ExportToDatabase to execute the export.
   The description of the ExportToDatabase method contains also a code example for a database export.

```
   // set the behaviour of the export with ExportSettings
   objExpSettings = objSpy.GetExportSettings()

   //set the destination with DatabaseConnection
   objDB = objSpy.GetDatabaseSettings();
   objDB.CreateMissingTables = true;
   objDB.CreateNew = true;
   objDB.File = "C:\\Temp\\Export.mdb";

try
{
   objImpDocFromDB.ExportToDatabase(objImpDocFromDB.RootElement, objExpSettings,
objDB);
}
catch(err)
{
   WScript.Echo("Error exporting to database.\n\n" +
                     "Error: " + (err.number & 0xffff) + "\n" +
                     "Description: " + err.description);
}
```

The JScript code listed above is available in the sample file ImportExport.js (see Example Files).

Importing data from a text file is similar to the import from a database. You must use other interfaces (described in steps 1 to 3 below) with different methods and properties:

1. Use a TextImportExportSettings object and set the properties:
   The method Application.GetTextImportExportSettings returns a new object to specify a text file for import.

   ```
   objImpSettings = objSpy.GetTextImportExportSettings();
   ```

   You have to set at least the ImportFile property to the path of the file for the import. Another important property is HeaderRow. Set it to False if the text file does not contain a leading line as a header row.

   ```
   objImpSettings.ImportFile = strExampleFolder + "Tutorial\\Shapes.txt";
   ```

2. Call Application.GetTextImportElementList to get a collection of all columns inside the text file:

```
objElementList = objSpy.GetTextImportElementList(objImpSettings);
```

3. Start the import with Application.ImportFromText.

```
objImpDocFromText = objSpy.ImportFromText(objImpSettings,objElementList);
```

```
try
{
   // specify the source of data import
   objImpSettings = objSpy.GetTextImportExportSettings();
   objImpSettings.ImportFile = strExampleFolder + "Tutorial\\Shapes.txt";
   objImpSettings.HeaderRow = false;

   // column filter
   objElementList = objSpy.GetTextImportElementList(objImpSettings);

   // import into a new XML file
   objImpDocFromText = objSpy.ImportFromText(objImpSettings,objElementList);
}
catch(err)
{
   WScript.Echo("Error importing from text file.\n\n" +
                    "Error: " + (err.number & 0xffff) + "\n" +
                    "Description: " + err.description);
}
```

The JScript code listed above is available in the sample file ImportExport.js (see Example Files).

To export data to text, carry out the steps below:

1. Use a TextImportExportSettings object and set the necessary properties.
2. Fill an ExportSettings object with the required values. See Item 2 in Export to database.
3. Build an element list with Document.GetExportElementList. See Item 3 in Export to database.
4. Call Document.ExportToText to execute the export.

```
objExpSettings = objSpy.GetExportSettings();
objExpSettings.ElementList =
objImpDocFromText.GetExportElementList(objImpDocFromText.RootElement,
objExpSettings);

objTextExp = objSpy.GetTextImportExportSettings();
objTextExp.HeaderRow = true;
objTextExp.DestinationFolder = "C:\\Temp";

try
{

       objImpDocFromText.ExportToText(objImpDocFromText.RootElement, objExpSettings,
objTextExp);
}
catch(err)
{
   WScript.Echo("Error exporting to text.\n\n" +
                    "Error: " + (err.number & 0xffff) + "\n" +
                    "Description: " + err.description);
}
```

The JScript code listed above is available in the sample file ImportExport.js (see Example

Files).

VBScript

VBScript is syntactically different than JScript but works in the same way. This section contains a listing showing how events are used with VBScript and an example.

For information about other functionality, refer to the JScript examples listed below:

- Start application or attach to a running instance
- Simple document access
- Iteration
- Error handling
- Import and export of data

Events

COM specifies that a client must register itself at a server for callbacks using the connection point mechanism. The automation interface for XMLSpy defines the necessary event interfaces. The way to connect to those events depends on the programming language you use in your client. The following code listing shows how this is done using VBScript.

The method `WScript.ConnectObject` is used to receive events.

To run this code, paste it into a file with .vbs extension, and either double-click in Windows Explorer, or run it from a command prompt.

```
' the event handler function
Function DocEvent_OnBeforeCloseDocument(objDocument)
   Call WScript.Echo("received event - before closing document")
End Function

' create or connect to XmlSpy
Set objWshShell = WScript.CreateObject("WScript.Shell")
Set objFSO = WScript.CreateObject("Scripting.FileSystemObject")
Set objSpy = WScript.GetObject("", "XMLSpy.Application")
' If only Authentic is installed (and XMLSpy is not installed) use:
' Set objSpy = WScript.GetObject("", "AuthenticDesktop.Application")
' If only XMLSpy 64-bit is intalled, use:
' Set objSpy = WScript.GetObject("", "XMLSpy_x64.Application")

' create document object and connect to its events
objSpy.Visible = True

' Find out user's personal folder and locate one of the installed examples.
personalFolder = objWshShell.ExpandEnvironmentStrings("%UserProfile%")
majorVersionYear = objSpy.MajorVersion + 1998
xmlspyExamplesFolder = personalFolder & "\Documents\Altova\XMLSpy" & majorVersionYear
 & "\Examples\"
docPath = xmlspyExamplesFolder & "ExpReport.xml"

' open a document
Set objDoc = objSpy.Documents.OpenFile (docPath, False)
Call WScript.ConnectObject(objDoc, "DocEvent_")

' keep running while waiting on the event
' in the meantime close the document in XMLSPY manually
Call WScript.Echo ("sleeping for 10 seconds ...")
Call WScript.Sleep (10000)
```

```
Set objDoc = Nothing
Call WScript.Echo ("stopped listening for event")
Call objSpy.Quit
```

Note: For 32-bit XMLSpy, the registered name, or programmatic identifier (ProgId) of the COM object is `XMLSpy.Application`. For 64-bit XMLSpy, the name is `XMLSpy_x64.Application`.

Example: Using Events

Authentic View supports event connection on a per-object basis. Implementation of this feature is based on COM connection points and is available in environments that support this mechanism.

The following example is a VBScript code example that shows how to use events from within a VBScript project.

```
' ---------------------------------------------------------------------------
' VBScript example that demonstrates how to use events.
' ---------------------------------------------------------------------------

' Event handler for OnSelectionChanged event of AuthenticView
Function AuthenticViewEvent_OnSelectionChanged(objAuthenticRange)
    If objAuthenticRange.FirstTextPosition <> objAuthenticRange.LastTextPosition Then
        Call WScript.Echo("Selection: " & objAuthenticRange.Text & vbNewLine &
vbNewLine & "Close this dialog.")
    Else
        Call WScript.Echo("Cursor position: " & objAuthenticRange.FirstTextPosition &
vbNewLine & vbNewLine & "Close this dialog.")
    End If
End Function

' Start/access XMLSpy and connect to its automation interface.
Set WshShell = WScript.CreateObject("WScript.Shell")
Set objSpy = GetObject("", "XMLSpy.Application")
' Make the UI of XMLSpy visible.
objSpy.Visible = True

' Find out user's personal folder and locate one of the installed XMLSpy examples.
personalFolder = WshShell.ExpandEnvironmentStrings("%UserProfile%")
majorVersionYear = objSpy.MajorVersion + 1998
xmlspyExamplesFolder = personalFolder & "\Documents\Altova\XMLSpy" & majorVersionYear
 & "\Examples\"
docPath = xmlspyExamplesFolder & "ExpReport.xml"

' Create object to access windows file system and test if the our document exists.
Set fso = CreateObject("Scripting.FileSystemObject")
If fso.FileExists(docPath) Then
    ' open the document
    Call objSpy.Documents.OpenFile(docPath, False)
    set objDoc = objSpy.ActiveDocument

    ' switch active document to authentic view
    objDoc.SwitchViewMode 4 ' spyViewAuthentic

    ' Register for connection point events on the authentic view of the active
document.
    ' Any function with a valid event name prefixed with "AuthenticViewEvent_" will
    ' be called when the corresponding event gets triggered on the specified object.
    set objView = objDoc.AuthenticView
    Call WScript.ConnectObject(objView, "AuthenticViewEvent_")
    Call WScript.Echo("Events are connected." & vbNewLine & vbNewLine & "Now set or
```

```
move the cursor in XMLSpy." & vbNewLine & vbNewLine & "Close this dialog to shut down
XMLSpy.")

    ' To disconnect from the events delete the reference to the object.
    set objView = Nothing
Else
    Call WScript.Echo("The file " & docPath & " does not exist.")
End If

' shut down XMLSpy when this script ends
objSpy.Visible = False
```

C#

The C# programming language can be used to access the Application API functionality. You could use Visual Studio 2008 or Visual Studio 2010 to create the C# code, saving it in a Visual Studio project. Create the project as follows:

1. In Microsoft Visual Studio, add a new project using **File | New | Project**.
2. Add a reference to the XMLSpy Type Library by clicking **Project | Add Reference**. The Add Reference dialog appears. Browse for the XMLSpy Type Library component, which is located in the XMLSpy application folder, and add it.
3. Enter the code you want.
4. Compile the code and run it.

Example C# project

Your XMLSpy package contains an example C# project, which is located in the C# folder of the API Examples folder:

Windows 10, Windows 8, Windows 7, Windows Vista	`C:\Users\<username>\Documents\Altova\XMLSpy2017\Examples\API\`
Windows XP	`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\API\`

You can compile and run the project from within Visual Studio 2008 or Visual Studio 2010.

The code listing below shows how basic application functionality can be used. This code is similar to the example C# project in the API Examples folder of your application package, but might differ slightly.

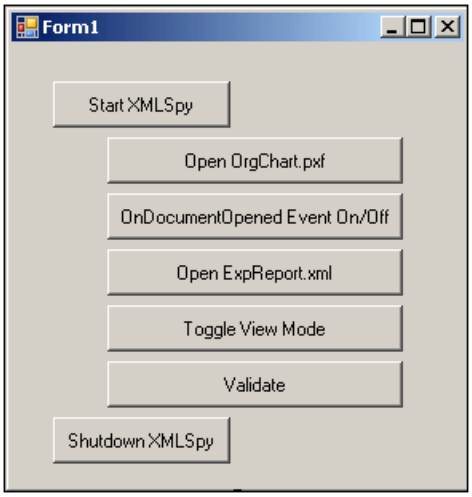
Platform configuration

If you have a 64-bit operating system and are using a 32-bit installation of XMLSpy, you must add the x86 platform in the solution's Configuration Manager and build the sample using this configuration.

A new x86 platform (for the active solution in Visual Studio) can be created in the New Solution Platform dialog (**Build | Configuration Manager | Active solution platform | <New...>**).

What the code listing below does

The example code listing below creates a simple user interface (*screenshot below*) with buttons that invoke basic XMLSpy operations:



- Start XMLSpy: Starts XMLSpy, which is registered as an automation server, or activates the application if it is already running.
- Open OrgChart.pxf: Locates one of the example documents installed with XMLSpy and opens it. If this document is already open it becomes the active document.
- OnDocumentOpened Event On/Off: Shows how to listen to XMLSpy events. When turned on, a message box will pop up after a document has been opened.
- Open ExpReport.xml: Opens another example document.
- Toggle View Mode: Changes the view of all open documents between Text View and Authentic View. The code shows how to iterate through open documents.
- Validate: Validates the active document and shows the result in a message box. The code shows how to handle errors and COM output parameters.
- Shut down XMLSpy: Stops XMLSpy.

You can modify the code (of the code listing below or of the example C# project in the API Examples folder) in any way you like and run it.

Compiling and running the example

In the API Examples folder, double-click the file `AutomateXMLSpy_VS2008.sln` (to open it in Visual Studio 2008) or the file `AutomateXMLSpy_VS2010.sln` (to open it in Visual Studio 2010). Alternatively the file can be opened from within Visual Studio (with **File | Open | Project/ Solution**). To compile and run the example, select **Debug | Start Debugging** or **Debug | Start Without Debugging**.

Code listing of the example

Given below is the C# code listing of the basic functionality of the form (`Form1.cs`) created in the `AutomateXMLSpy` example. Note that the code listed below might differ slightly from the code in the API Examples form.The listing below is commented for ease of understanding. Parts of the code are also presented separately in the sub-sections of this section, according to the Application API functionality they access.

The code essentially consists of a series of handlers for the buttons in the user interface shown

in the screenshot above.

```
namespace WindowsFormsApplication2
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      // An instance of XMLSpy accessed via its automation interface
      XMLSpyLib.Application XMLSpy;

      // Location of examples installed with XMLSpy
      String strExamplesFolder;

      private void Form1_Load(object sender, EventArgs e)
      {
         // Locate examples installed with XMLSpy
         // REMARK: You might need to adapt this if you have a different major version of the product
         strExamplesFolder = Environment.GetEnvironmentVariable("USERPROFILE") + "\\My Documents\\Altova\
\XMLSpy2012\\Examples\\";
      }

      // Handler for the "Start XMLSpy" button
      private void StartXMLSpy_Click(object sender, EventArgs e)
      {
         if (XMLSpy == null)
         {
            Cursor.Current = Cursors.WaitCursor;

            // If no XMLSpy instance is open, create one and make it visible
            XMLSpy = new XMLSpyLib.Application();
            XMLSpy.Visible = true;

            Cursor.Current = Cursors.Default;
         }
         else
         {
            // If an instance of XMLSpy is already running, make sure it's visible
            if (XMLSpy.Visible)
               XMLSpy.Visible = true;
         }
      }

      // Handler for the "Open OrgChart.pxf" button
      private void openOrgChart_Click(object sender, EventArgs e)
      {
         // Make sure there's a running XMLSpy instance, and that it's visible
         StartXMLSpy_Click(null, null);

         // Open one of the example files installed with the product
         XMLSpy.Documents.OpenFile(strExamplesFolder + "OrgChart.pxf", false);
      }

      // Handler for the "Open ExpReport.xml" button
      private void openExpReport_Click(object sender, EventArgs e)
      {
         // Make sure there's a running XMLSpy instance, and that it's visible
         StartXMLSpy_Click(null, null);

         // Open one of the sample files installed with the product.
         XMLSpy.Documents.OpenFile(strExamplesFolder + "ExpReport.xml", false);
      }

      // Handler for the "Toggle View Mode" button
      private void toggleView_Click(object sender, EventArgs e)
      {
         // Make sure there's a running XMLSpy instance, and that it's visible
         StartXMLSpy_Click(null, null);

         // Iterate through all open documents and toggle view between Text View and Authentic View
         foreach (XMLSpyLib.Document doc in XMLSpy.Documents)
            if (doc.CurrentViewMode == XMLSpyLib.SPYViewModes.spyViewText)
               doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewAuthentic);
```

```
            else
               doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewText);
         }

         // Handler for the "Shutdown XMLSpy" button
         // Shut down the application instance by explicitly releasing the COM object
         private void shutdownXMLSpy_Click(object sender, EventArgs e)
         {
            if (XMLSpy != null)
            {
               // Allow shutdown of XMLSpy by releasing the UI
               XMLSpy.Visible = false;

               // Explicitly release the COM object
               try
               {
                  while (System.Runtime.InteropServices.Marshal.ReleaseComObject(XMLSpy) > 0);
               }
               finally
               {
                  // Disallow subsequent access to this object
                  XMLSpy = null;
               }
            }
         }

         // Handler for button "Validate"
         private void validate_Click(object sender, EventArgs e)
         {
            // COM errors are returned to C# as exceptions. We use a try/catch block to handle them.
            try
            {
               // Method 'IsValid' is one of the few functions that uses output parameters
               // Use 'object' type for these parameters
               object strErrorText = "";
               object nErrorNumber = 0;
               object errorData = null;

               if (XMLSpy.ActiveDocument.IsValid(ref strErrorText, ref nErrorNumber, ref errorData))
               {
                  // The COM call succeeded but the document is not valid
                  // A detailed description of the problem is returned in strErrorText, nErrorNumber and errorData
                  listBoxMessages.Items.Add("Document " + XMLSpy.ActiveDocument.Name + " is not valid.");
                  listBoxMessages.Items.Add("\tErrorText :" + strErrorText);
                  listBoxMessages.Items.Add("\tErrorNumber:" + nErrorNumber);
                  listBoxMessages.Items.Add("\tElement   :" + (errorData != null ? ((XMLSpyLib.XMLData)errorData).TextValue
:"null"));
               }
               else
               {
                  // The COM call succeeded and the document is valid
                  listBoxMessages.Items.Add("Document " + XMLSpy.ActiveDocument.Name + " is valid.");
               }
            }
            catch (Exception ex)
            {
               // The COM call was not successful
               // Probably no application instance has been started or no document is open.
               listBoxMessages.Items.Add("Error validating active document:" + ex.Message);
            }
         }



         // Event handler for OnDocumentOpened event
         private void handleOnDocumentOpened(XMLSpyLib.Document in_pDocument)
         {
            MessageBox.Show("Document " + in_pDocument.Name + " was opened!");
         }

         // Remember if the event handler is currently registered.
         private bool bEventHandlerIsRegistered = false;

         // Handler for button 'OnDocumentOpened Event On/Off
         private void toggleOnDocumentOpenedEvent_Click(object sender, EventArgs e)
         {
            if (XMLSpy != null)
            {
               if (bEventHandlerIsRegistered)
```

```
            XMLSpy.OnDocumentOpened -= new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
         else
            XMLSpy.OnDocumentOpened += new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);

         bEventHandlerIsRegistered = !bEventHandlerIsRegistered;
      }
   }
  }
}
```

Add Reference to XMLSpy API

Add the application's type library as a reference in a .NET project as follows: With the .NET project open, click **Project | Add Reference**. Then browse for the type library, which is called `XMLSpy.tlb`, and is located in the XMLSpy application folder.

Then declare a variable to access the XMLSpy API:

```
// An instance of XMLSpy is accessed via its automation interface.
XMLSpyLib.Application XMLSpy;
```

Application Startup and Shutdown

In the code snippets below, the methods `StartXMLSpy_Click` and `ShutdownXMLSpy_Click` are those assigned to buttons in the AutomateXMLSpy example that, respectively, start up and shut down the application. This example is located in the C# folder of the API Examples folder (*see the file Form1.cs*):

Windows 10, Windows 8, Windows 7, Windows Vista	`C:\Users\<username>\Documents\Altova\XMLSpy2017\Examples\API\`
Windows XP	`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\API\`

You can compile and run the project from within Visual Studio 2008 or Visual Studio 2010.

Starting XMLSpy

The following code snippet from the AutomateXMLSpy example shows how to start up the application.

```
// Handler for the "Start XMLSpy" button
private void StartXMLSpy_Click(object sender, EventArgs e)
{
   if (XMLSpy == null)
   {
      Cursor.Current = Cursors.WaitCursor;

      // If no XMLSpy instance is running, we create one and make it visible
      XMLSpy = new XMLSpyLib.Application();
      XMLSpy.Visible = true;

      Cursor.Current = Cursors.Default;
   }
   else
   {
      // If an instance of XMLSpy is already running, make sure it's visible
```

```
      if (XMLSpy.Visible)
        XMLSpy.Visible = true;
    }
  }
```

Shutting down XMLSpy

The following code snippet from the AutomateXMLSpy example shows how to shut down the application.

```
    // Handler for the "Shutdown XMLSpy" button
  // Shut down the application instance by explicitly releasing the COM object
  private void shutdownXMLSpy_Click(object sender, EventArgs e)
  {
    if (XMLSpy != null)
    {
      // Allow shutdown of XMLSpy by releasing the UI
      XMLSpy.Visible = false;

      // Explicitly release COM object
      try
      {
        while (System.Runtime.InteropServices.Marshal.ReleaseComObject(XMLSpy) > 0);
      }
      finally
      {
        // Disallow subsequent access to this object
        XMLSpy = null;
      }
    }
  }
```

Opening Documents

The code snippets below (from the AutomateXMLSpy example) show how two files are opened via two separate methods assigned to two buttons in the user interface. Both methods use the same Application API access mechanism: `XMLSpy.Documents.OpenFile(string, boolean)`.

The AutomateXMLSpy example (*see the file Form1.cs*) is located in the C# folder of the API Examples folder:

Windows 10, Windows 8, Windows 7, Windows Vista	`C:\Users\<username>\Documents\Altova\XMLSpy2017\Examples\API\`
Windows XP	`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\API\`

You can compile and run the project from within Visual Studio 2008 or Visual Studio 2010.

Code snippet

```
    // Handler for the "Open OrgChart.pxf" button
  private void openOrgChart_Click(object sender, EventArgs e)
  {
    // Make sure there's a running XMLSpy instance, and that it's visible
    StartXMLSpy_Click(null, null);

    // Open a file from the Examples folder installed with the product
    XMLSpy.Documents.OpenFile(strExamplesFolder + "OrgChart.pxf", false);
  }
```

```
// Handler for the "Open ExpReport.xml" button
private void openExpReport_Click(object sender, EventArgs e)
{
   // Make sure there's a running XMLSpy instance, and that it's visible
   StartXMLSpy_Click(null, null);

   // Open a file from the Examples folder installed with the product
   XMLSpy.Documents.OpenFile(strExamplesFolder + "ExpReport.xml", false);
}
```

The file opened last will be the active file.

Iterating through Open Documents

The code snippet below (from the AutomateXMLSpy example; *see the file Form1.cs*) shows how to iterate through open documents. A condition is then tested within the iteration loop, and the document view is switched between Text View and Authentic View.

```
   // Handler for the "Toggle view mode" button
private void toggleView_Click(object sender, EventArgs e)
{
   // Make sure there's a running XMLSpy instance, and that it's visible
   StartXMLSpy_Click(null, null);

   // Iterate through open documents and toggle current view between text and authentic view.
   foreach (XMLSpyLib.Document doc in XMLSpy.Documents)
      if (doc.CurrentViewMode == XMLSpyLib.SPYViewModes.spyViewText)
         doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewAuthentic);
      else
         doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewText);
}
```

The AutomateXMLSpy example example is located in the C# folder of the API Examples folder:

Windows 10, Windows 8, Windows 7, Windows Vista	`C:\Users\<username>\Documents\Altova\XMLSpy2017\Examples\API\`
Windows XP	`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\API\`

You can compile and run the project from within Visual Studio 2008 or Visual Studio 2010.

Errors and COM Output Parameters

The code snippet below (from the AutomateXMLSpy example) shows how to handle errors and COM output parameters. The method `XMLSpy.ActiveDocument.IsValid(ref strErrorText, ref nErrorNumber, ref errorData)` uses output parameters that are used, in the code snippet below, to generate an error-message text.

The AutomateXMLSpy example (*see the file Form1.cs*) is located in the C# folder of the API Examples folder:

Windows 10, Windows 8, Windows 7,	`C:\Users\<username>\Documents\Altova\XMLSpy2017\Examples\API\`

Windows Vista	
Windows XP	C:\Documents and Settings\<username>\My Documents\ Altova\XMLSpy2017\Examples\API\

You can compile and run the project from within Visual Studio 2008 or Visual Studio 2010.

Code snippet

```
    // Handler for button "Validate"
    private void validate_Click(object sender, EventArgs e)
    {
      // COM errors are returned to C# as exceptions. We use a try/catch block to handle them.
      try
      {
        // Method 'IsValid' is one of the few functions that uses output parameters
        // Use 'object' type for these parameters
        object strErrorText = "";
        object nErrorNumber = 0;
        object errorData = null;

        if (XMLSpyActiveDocument.IsValid(ref strErrorText, ref nErrorNumber, ref errorData))
        {
          // The COM call succeeded but the document is not valid
          // A detailed description of the problem is returned in strErrorText, nErrorNumber and errorData
          listBoxMessages.Items.Add("Document " + XMLSpyActiveDocument.Name + " is not valid.");
          listBoxMessages.Items.Add("\tErrorText :" + strErrorText);
          listBoxMessages.Items.Add("\tErrorNumber:" + nErrorNumber);
          listBoxMessages.Items.Add("\tElement   :" + (errorData != null ? ((XMLSpyLib.XMLData)errorData).TextValue
:"null"));
        }
        else
        {
          // The COM call succeeded and the document is valid
          listBoxMessages.Items.Add("Document " + XMLSpyActiveDocument.Name + " is valid.");
        }
      }
      catch (Exception ex)
      {
        // The COM call was not successful
        // Probably no application instance has been started or no document is open.
        listBoxMessages.Items.Add("Error validating active document:" + exMessage);
      }
    }
```

Events

The code snippet below (from the AutomateXMLSpy example) lists the code for two event handlers. The AutomateXMLSpy example (*see the file Form1.cs*) is located in the C# folder of the API Examples folder:

Windows 10, Windows 8, Windows 7, Windows Vista	C:\Users\<username>\Documents\ Altova\XMLSpy2017\Examples\API\
Windows XP	C:\Documents and Settings\<username>\My Documents\ Altova\XMLSpy2017\Examples\API\

You can compile and run the project from within Visual Studio 2008 or Visual Studio 2010.

```
    // Event handler for OnDocumentOpened event
    private void handleOnDocumentOpened(XMLSpyLib.Document ipDocument)
    {
```

```
            MessageBox.Show("Document " + ipDocumentName + " was opened!");
        }

        // Remember if the event handler is currently registered.
        private bool bEventHandlerIsRegistered = false;

        // Handler for button 'OnDocuemntOpened Event On/Off
        private void toggleOnDocumentOpenedEvent_Click(object sender, EventArgs e)
        {
            if (XMLSpy != null)
            {
                if (bEventHandlerIsRegistered)
                    XMLSpy.OnDocumentOpened -= new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
                else
                    XMLSpy.OnDocumentOpened += new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);

                bEventHandlerIsRegistered = !bEventHandlerIsRegistered;
            }
        }
```

Java

The Application API can be accessed from Java code. To allow accessing the XMLSpy automation server directly from Java code, the libraries listed below must reside in the `classpath`. They are installed in the folder: `JavaAPI` in the XMLSpy application folder.

- `AltovaAutomation.dll`: a JNI wrapper for Altova automation servers (`AltovaAutomation_x64.dll` in the case of 64-bit versions)
- `AltovaAutomation.jar`: Java classes to access Altova automation servers
- `XMLSpyAPI.jar`: Java classes that wrap the XMLSpy automation interface
- `XMLSpyAPI_JavaDoc.zip`: a `Javadoc` file containing help documentation for the Java API

Note: In order to use the Java API, the DLL and Jar files must be on the Java Classpath.

Example Java project

An example Java project is supplied with your product installation. You can test the Java project and modify and use it as you like. For more details of the example Java project, see the section, Example Java Project.

Rules for mapping the Application API names to Java

The rules for mapping between the Application API and the Java wrapper are as follows:

- ***Classes and class names***
  For every interface of the XMLSpy automation interface a Java class exists with the name of the interface.

- ***Method names***
  Method names on the Java interface are the same as used on the COM interfaces but start with a small letter to conform to Java naming conventions. To access COM properties, Java methods that prefix the property name with `get` and `set` can be used. If a property does not support write-access, no setter method is available. Example: For the `Name` property of the `Document` interface, the Java methods `getName` and `setName` are available.

- ***Enumerations***

For every enumeration defined in the automation interface, a Java enumeration is defined with the same name and values.

- ***Events and event handlers***
  For every interface in the automation interface that supports events, a Java interface with the same name plus '`Event`' is available. To simplify the overloading of single events, a Java class with default implementations for all events is provided. The name of this Java class is the name of the event interface plus '`DefaultHandler`'. For example:
  `Application`: Java class to access the application
  `ApplicationEvents`: Events interface for the Application
  `ApplicationEventsDefaultHandler`: Default handler for `ApplicationEvents`

Exceptions to mapping rules

There are some exceptions to the rules listed above. These are listed below:

Interface	Java name
`Document`, method `SetEncoding`	`setFileEncoding`
`AuthenticView`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Clone`	`cloneRange`

This section

This section explains how some basic XMLSpy functionality can be accessed from Java code. It is organized into the following sub-sections:

- Example Java Project
- Application Startup and Shutdown
- Simple Document Access
- Iterations
- Use of Out-Parameters
- Event Handlers

Example Java Project

The XMLSpy installation package contains an example Java project, located in the Java folder of the API Examples folder:

Windows 10, Windows 8, Windows 7, Windows Vista	`C:\Users\<username>\Documents\Altova\XMLSpy2017\Examples\API\`
Windows XP	`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2017\Examples\API\`

This folder contains Java examples for the XMLSpy API. You can test it directly from the

command line using the batch file `BuildAndRun.bat`, or you can compile and run the example project from within Eclipse. See below for instructions on how to use these procedures.

File list

The Java examples folder contains all the files required to run the example project. These files are listed below. If you are using a 64-bit version of the application, some filenames contain `_x64` in the name. These filenames are indicated with `(_x64)`.

`AltovaAutomation(_x64).dll`	Java-COM bridge: DLL part
`AltovaAutomation.jar`	Java-COM bridge: Java library part
`XMLSpyAPI.jar`	Java classes of the XMLSpy API
`RunXMLSpy.java`	Java example source code
`BuildAndRun.bat`	Batch file to compile and run example code from the command line prompt. Expects folder where Java Virtual Machine resides as parameter.
`.classpath`	Eclipse project helper file
`.project`	Eclipse project file
`XMLSpyAPI_JavaDoc.zip`	`Javadoc` file containing help documentation for the Java API

What the example does

The example starts up XMLSpy and performs a few operations, including opening and closing documents. When done, XMLSpy stays open. You must close it manually.

- Start XML Spy: Starts XMLSpy, which is registered as an automation server, or activates XMLSpy if it is already running.
- Open OrgChart.pxf: Locates one of the example documents installed with XMLSpy and opens it.
- Iteration and Changing the View Mode: Changes the view of all open documents to Text View. The code also shows how to iterate through open documents.
- Iteration, validation, output parameters: Validates the active document and shows the result in a message box. The code shows how to use output parameters.
- Event Handling: Shows how to handle XMLSpy events.
- Shut down XMLSpy: Shuts down XMLSpy.

You can modify the example in any way you like and run it.

Running the example from the command line

To run the example from the command line, open a command prompt window, go to the Java folder of the API Examples folder (*see above for location*), and then type:

```
buildAndRun.bat "<Path-to-the-Java-bin-folder>"
```

The Java binary folder must be that of a JDK 1.5 or later installation on your computer.

Press the **Return** key. The Java source in `RunXMLSpy.java` will be compiled and then executed.

Loading the example in Eclipse

Open Eclipse and use the **Import | Existing Projects into Workspace** command to add the Eclipse project file (`.project`) located in the Java folder of the API Examples folder (*see above for location*). The project `RunXMLSpy` will then appear in your Package Explorer or Navigator.

Select the project and then the command **Run as | Java Application** to execute the example.

Note: You can select a class name or method of the Java API and press F1 to get help for that class or method.

Java source code listing

The Java source code in the example file `RunXMLSpy.java` is listed below with comments.

```
001 // Access general JAVA-COM bridge classes
002 import com.altova.automation.libs.*;
003
004 // Access XMLSpy Java-COM bridge
005 import com.altova.automation.XMLSpy.*;
006 import com.altova.automation.XMLSpyEnums.SPYViewModes;
007
008 /**
009  * An example that starts XMLSpy COM server and performs view operations on it
010  * Feel free to extend
011  */
012 public class RunXMLSpy
013 {
014  public static void main(String[] args)
015  {
016   // An instance of the application.
017   Application xmlSpy = null;
018
019   // Instead of COM error handling, use Java exception mechanism
020   try
021    {
022    // Start XMLSpy as COM server
023    xmlSpy = new Application();
024
025    // COM servers start up invisible, so make it visible
026    xmlSpy.setVisible(true);
027
028    // Locate samples installed with the product
029    String strExamplesFolder =
030    System.getenv("USERPROFILE") + "\\My Documents\\Altova\\XMLSpy2012\\Examples\\";
031
032    // Open two example files
033    xmlSpy.getDocuments().openFile(strExamplesFolder + "OrgChart.spf", false);
034    xmlSpy.getDocuments().openFile(strExamplesFolder + "ExpReport.xml", false);
035
036    // Iterate through open documents and set view mode to 'Text'.
037    for (Document doc:xmlSpy.getDocuments())
038    if (doc.getCurrentViewMode() != SPYViewModes.spyViewText)
039       doc.switchViewMode(SPYViewModes.spyViewText);
040
041    // An alternative iteration mode is index-based
042    // COM indices are typically zero-based
043    Documents documents = xmlSpy.getDocuments();
044    for (int i=1; i<=documents.getCount();
045       i++)
046       {
047        Document doc = documents.getItem(i);
048
049    // Validation is one of the few methods to have output parameters.
050    // The class JVariant is the correct type for parameters in these cases.
051    // To get values back mark them with the by-reference flag.
052    JVariant validationErrorText = new
053
054    JVariant.JStringVariant("");
```

```
055
056     validationErrorText.setByRefFlag();
057     JVariant validationErrorCount = new
058
059     JVariant.JInitVariant(0);
060
061     validationErrorCount.setByRefFlag();
062     JVariant validationErrorXMLData = new
063
064     JVariant.JInitDispatchVariant(0);
065
066     validationErrorXMLData.setByRefFlag();
067     if (!doc.isValid(validationErrorText, validationErrorCount, validationErrorXMLData))
068       System.out.println("Document" + doc.getName() + " is not wellformed - " + validationErrorText.getStringValue());
069      else
070       System.out.println("Document" + doc.getName() + " is wellformed.");
071     }
072
073     // The following lines attach to the document events using a default implementation
074     // for the events and override one of its methods.
075     // If you want to override all document events it is better to derive your listener class
076     // from DocumentEvents and implement all methods of this interface.
077     Document doc = xmlSpy.getActiveDocument();
078     doc.addListener(new
079
080 DocumentEventsDefaultHandler()
081     {
082      @Override
083      public boolean
084
085 onBeforeCloseDocument(Document ippDoc) throws AutomationException
086       {
087        System.out.println("Document
088
089 " + ippDoc.getName() + " requested closing.");
090
091        // Allow closing of document
092        return true;
093       }
094     });
095     doc.close(true);
096     doc = null;
097
098     System.out.println("Watch XMLSpy!");
099    }
100    catch (AutomationException e)
101    {
102     // e.printStackTrace();
103    }
104    finally
105    {
106     // Make sure that XMLSpy can shut down properly.
107     if (xmlSpy != null)
108       xmlSpy.dispose();
109
110     // Since the COM server was made visible and still is visible,
111     // it will keep running, and needs to be closed manually.
112     System.out.println("Now close XMLSpy!");
113    }
114  }
115 }
```

Application Startup and Shutdown

The code listings below show how the application can be started up and shut down.

Application startup

Before starting up the application, the appropriate classes must be imported (*see below*).

```
01 // Access general JAVA-COM bridge classes
02 import com.altova.automation.libs.*;
03
04 // Access XMLSpy Java-COM bridge
05 import com.altova.automation.XMLSpy.*;
06 import com.altova.automation.XMLSpyEnums.SPYViewModes;
```

```
07
08 /**
09  * An example that starts XMLSpy COM server and performs view operations on it
10  * Feel free to extend
11  */
12 public class RunXMLSpy
13 {
14   public static void main(String[] args)
15   {
16     // An instance of the application.
17     Application xmlSpy = null;
18
19     // Instead of COM error handling, use Java exception mechanism
20     try
21     {
22        // Start XMLSpy as COM server
23         xmlSpy = new Application();
24        // COM servers start up invisible, so make it visible
25         xmlSpy.setVisible(true);
26
27 ...
28     }
29   }
30 }
```

Application shutdown

The application can be shut down as shown below.

```
01 {
02     // Allow shutdown of XMLSpy by releasing the UI.
03     xmlSpy.setVisible(true);
04
05     // Make sure that XMLSpy can shut down properly.
06     if (xmlSpy != null)
07       xmlSpy.dispose();
08
09     // Since the COM server was made visible and still is visible,
10     // it will keep running, and needs to be closed manually.
11     System.out.println("Now close XMLSpy!");
12 }
```

Simple Document Access

The code listing below shows how to open a document.

```
1 // Locate samples installed with the product
2 String strExamplesFolder =
3 System.getenv("USERPROFILE") + "\\My Documents\\Altova\\XMLSpy2012\\Examples\\";
4
5
6 // Open file
7 xmlSpy.getDocuments().openFile(strExamplesFolder + "OrgChart.pxf", false);
```

Iterations

The listing below shows how to iterate through open documents.

```
01 // Iterate through open documents and set view mode to 'Text'.
02 for (Document doc:xmlSpy.getDocuments())
03   if (doc.getCurrentViewMode() != SPYViewModes.spyViewText)
04         doc.switchViewMode(SPYViewModes.spyViewText);
05
06 // An alternative iteration mode is index-based
07 // COM indices are typically zero-based
08 Documents documents = xmlSpy.getDocuments();
09   for (int i= 1;i<= documents.getCount();
```

```
10      i++)
11    {
12      Document doc = documents.getItem(i);
13          ...
14    }
```

Use of Out-Parameters

The code listing below iterates through open documents and validates each of them. For each validation, a message is generated using the output parameters of the Validation method.

```
01 // An alternative iteration mode is index-based
02 // COM indices are typically zero-based
03 Documents documents = xmlSpy.getDocuments();
04 for (int i=1;i<=documents.getCount();
05     i++)
06   {
07     Document doc = documents.getItem(i);
08
09 // Validation is one of the few methods to have output parameters.
10 // The class JVariant is the correct type for parameters in these cases.
11 // To get values back mark them with the by-reference flag.
12 JVariant validationErrorText = new
13
14 JVariant.JStringVariant("");
15
16 validationErrorText.setByRefFlag();
17     JVariant validationErrorCount = new
18
19 JVariant.JIntVariant(0);
20
21 validationErrorCount.setByRefFlag();
22     JVariant validationErrorXMLData = new
23
24 JVariant.JIDispatchVariant(0);
25
26 validationErrorXMLData.setByRefFlag();
27     if (doc.isValid(validationErrorText,
28
29       validationErrorCount, validationErrorXMLData))
30       System.out.println("Document
31
32       " + doc.getName() + " is not wellformed - " +
33
34       validationErrorText.getStringValue());
35     else
36       System.out.println("Document
37
38       " + doc.getName() + " is wellformed.");
39   }
```

Event Handlers

The listing below shows how to listen for and use events.

```
01 // The following lines attach to the document events using a default implementation
02 // for the events and override one of its methods.
03 // If you want to override all document events it is better to derive your listener class
04 // from DocumentEvents and implement all methods of this interface.
05
06 Document doc = xmlSpy.getActiveDocument();
07 doc.addListener(new DocumentEventsDefaultHandler()
08   {
09     @Override
10     public boolean
11     onBeforeCloseDocument(Document ipDoc) throws AutomationException
12     {
13       System.out.println("Document " + ipDoc.getName() + " requested closing.");
14
15       // Allow closing of document
```

```
16      return true;
17     }
18    });
19 doc.close(true);
20 doc = null;
```

3.1.3 The DOM and XMLData

The XMLData interface gives you full access to the XML structure behind the current document with less methods than DOM and is much simpler. The XMLData interface is a minimalist approach to reading and modifying existing, or newly created XML data. You might however, want to use a DOM tree because you can access one from an external source or you just prefer the MSXML DOM implementation.

The ProcessDOMNode() and ProcessXMLDataNode() functions provided below convert any segments of an XML structure between XMLData and DOM.

To use the ProcessDOMNode() function:

- pass the root element of the DOM segment you want to convert in objNode and
- pass the plugin object with the CreateChild() method in objCreator

To use the ProcessXMLDataNode() function:

- pass the root element of the XMLData segment in objXMLData and
- pass the DOMDocument object created with MSXML in xmlDoc

```
////////////////////////////////////////////////////////////////
// DOM To XMLData conversion
Function ProcessDOMNode(objNode,objCreator)
{
 var objRoot;
 objRoot = CreateXMLDataFromDOMNode(objNode,objCreator);

 If(objRoot){
  If((objNode.nodeValue != Null) && (objNode.nodeValue.length > 0))
   objRoot.TextValue = objNode.nodeValue;
   // add attributes
  If(objNode.attributes) {
   var Attribute;
   var oNodeList = objNode.attributes;

   For(var i= 0;i< oNodeList.length;i++) {
    Attribute = oNodeList.item(i);

    var newNode;
    newNode = ProcessDOMNode(Attribute,objCreator);

    objRoot.AppendChild(newNode);
   }
  }
  If(objNode.hasChildNodes) {
   try {
    // add children
    var Item;
    oNodeList = objNode.childNodes;

    For(var i= 0;i< oNodeList.length;i++) {
       Item = oNodeList.item(i);

     var newNode;
     newNode = ProcessDOMNode(Item,objCreator);
```

```
      objRoot.AppendChild(newNode);
     }
    }
    catch(err) {
    }
   }
  }
  Return objRoot;
}

Function CreateXMLDataFromDOMNode(objNode,objCreator)
{
 varbSetName = True;
 varbSetValue = True;

 varnKind = 4;

 switch(objNode.nodeType) {
  Case 2:nKind = 5;break;
  Case 3:nKind = 6;bSetName = False;break;
  Case 4:nKind = 7;bSetName = False;break;
  Case 8:nKind = 8;bSetName = False;break;
  Case 7:nKind = 9;break;
 }
 varobjNew = Null;
 objNew = objCreator.CreateChild(nKind);

 If(bSetName)
  objNew.Name = objNode.nodeName;

 If(bSetValue && (objNode.nodeValue != Null))
  objNew.TextValue = objNode.nodeValue;

 Return objNew;
}
////////////////////////////////////////////////////////////////
// XMLData To DOM conversion

Function ProcessXMLDataNode(objXMLData,xmlDoc)
{
 varobjRoot;
 objRoot = CreateDOMNodeFromXMLData(objXMLData,xmlDoc);

 If(objRoot){
  If(IsTextNodeEnabled(objRoot) && (objXMLData.TextValue.length > 0))
   objRoot.appendChild(xmlDoc.createTextNode(objXMLData.TextValue));

  If(objXMLData.HasChildren){
   try {
    varobjChild;
    objChild = objXMLData.GetFirstChild(-1);

    While(True){
     If(objChild) {
      varnewNode;
      newNode = ProcessXMLDataNode(objChild,xmlDoc);

      If(newNode.nodeType == 2) {
       // child node is an attribute
       objRoot.attributes.setNamedItem(newNode);
      }
      Else
       objRoot.appendChild(newNode);
```

```
      }
      objChild = objXMLData.GetNextChild();
     }
    }
    catch(err) {
    }
   }
  }
  Return objRoot;
}

Function CreateDOMNodeFromXMLData(objXMLData,xmlDoc)
{
 switch(objXMLData.Kind) {
  Case 4:Return xmlDoc.createElement(objXMLData.Name);
  Case 5:Return xmlDoc.createAttribute(objXMLData.Name);
  Case 6:Return xmlDoc.createTextNode(objXMLData.TextValue);
  Case 7:Return xmlDoc.createCDATASection(objXMLData.TextValue);
  Case 8:Return xmlDoc.createComment(objXMLData.TextValue);
  Case 9:Return xmlDoc.createProcessingInstruction(objXMLData.Name,objXMLData.TextValue);
 }

 Return xmlDoc.createElement(objXMLData.Name);
}
Function IsTextNodeEnabled(objNode)
{
 switch(objNode.nodeType) {
  Case 1:
  Case 2:
  Case 5:
  Case 6:
  Case 11:Return True;
 }
 Return False;
}
```

3.1.4 Obsolete: Authentic View Row operations

If the schema on which an XML document is based specifies that an element is repeatable, such a structure can be represented in Authentic View as a table. When represented as a table, rows and their contents can be manipulated individually, thereby allowing you to manipulate each of the repeatable elements individually. Such row operations would be performed by an external script.

If an external script is to perform row operations then two steps must occur:

- The first step checks whether the cursor is currently in a row using a property. Such a check could be, for example, `IsRowInsertEnabled`, which returns a value of either `TRUE` or `FALSE`.
- If the return value is `TRUE` then a row method, such as `RowAppend`, can be called. (`RowAppend` has no parameters and returns no value.)

The following is a list of properties and methods available for table operations. Each property returns a `BOOL`, and the methods have no parameter.

Property	Method	Table operations
IsRowInsertEnabled	**RowInsert**, superseded by AuthenticRange.InsertRow	Insert row operation

IsRowAppendEnabled	**RowAppend**, superseded by AuthenticRange.AppendRow	Append row operation
IsRowDeleteEnabled	**RowDelete**, superseded by AuthenticRange.DeleteRow	Delete row operation
IsRowMoveUpEnabled	**RowMoveUp**, superseded by AuthenticRange.MoveRowUp	Move XML data up one row
IsRowMoveDownEnabled	**RowMoveDown**, superseded by AuthenticRange.MoveRowDown	Move XML data down one row
IsRowDuplicateEnabled	**RowDuplicate**, superseded by AuthenticRange.DuplicateRow	Duplicate currently selected row

3.1.5 Obsolete: Authentic View Editing operations

When XML data is displayed as data in Authentic View, it is possible to manipulate individual elements using standard editing operations such as cut, copy, and paste. However, not all XML data nodes can be edited. So, in order to carry out an editing operation, first a property is used to test whether editing is possible, and then a method is called to perform the editing operation.

The only method that does not have a test is the method `EditSelectAll`, which automatically selects all elements displayed in the document.

The following is a list of properties and methods that perform editing operations. Each property returns a `BOOL`, and the methods have no parameter.

Property	Method	Editing operation
IsEditUndoEnabled	**EditUndo**, superseded by AuthenticView.Undo	Undo an editing operation
IsEditRedoEnabled	**EditRedo**, superseded by AuthenticView.Redo	Redo an editing operation
IsEditCopyEnabled	**EditCopy**, superseded by AuthenticRange.Copy	Copy selected text to the clipboard
IsEditCutEnabled	**EditCut**, superseded by AuthenticRange.Cut	Cut selected text to the clipboard
IsEditPasteEnabled	**EditPaste**, superseded by AuthenticRange.Paste	Paste from clipboard to current cursor position
IsEditClearEnabled	**EditClear**, superseded by AuthenticRange.Delete	Clear selected text from XML document

3.2 Interfaces

Object Hierarchy

Application
- SpyProject
  - SpyProjectItems
    - SpyProjectItem
- Documents
  - Document
    - GridView
    - AuthenticView
      - AuthenticRange
      - AuthenticDataTransfer (previously DocEditDataTransfer)
    - OldAuthenticView (previously DocEditView, **now obsolete**, superseded by AuthenticView and AuthenticRange)
      - AuthenticSelection (previously DocEditSelection, **now obsolete**, superseded by AuthenticRange)
      - AuthenticEvent (previously DocEditEvent, **now obsolete**)
      - AuthenticDataTransfer (previously DocEditDataTransfer)
    - TextView
    - XMLData
- Dialogs
  - CodeGeneratorDlg
  - FileSelectionDlg
  - SchemaDocumentationDlg
  - GenerateSampleXMLDlg
  - DTDSchemaGeneratorDlg
  - FindInFilesDlg
  - WSDLDocumentationDlg
  - WSDL20DocumentationDlg
  - XBRLDocumentationDlg
- DatabaseConnection
- ExportSettings
- TextImportExportSettings
- ElementList
  - ElementListItem

Enumerations

Description
This chapter contains the reference of the XMLSpy 1.5 Type Library.

Most of the given examples are written in VisualBasic. These code snippets assume that there is a variable defined and set, called **objSpy of type Application**. There are also some code samples written in JavaScript.

3.2.1 Application

See also

Methods
GetDatabaseImportElementList
GetDatabaseSettings
GetDatabaseTables
ImportFromDatabase
CreateXMLSchemaFromDBStructure

GetTextImportElementList
GetTextImportExportSettings

ImportFromText

ImportFromWord

ImportFromSchema

GetExportSettings

NewProject
OpenProject

AddMacroMenuItem
ClearMacroMenu

ShowForm

ShowApplication

URLDelete
URLMakeDirectory

AddXSLT_XQParameter
GetXSLT_XQParameterCount
GetXSLT_XQParameterName
GetXSLT_XQParameterXPath
RemoveXSLT_XQParameter

FindInFiles

Quit

Properties
Application
Parent

ActiveDocument
Documents

CurrentProject

Dialogs

WarningNumber
WarningText

Status
MajorVersion
MinorVersion
Edition
IsAPISupported
ServicePackVersion

Description
Application is the root for all other objects. It is the only object you can create by `CreateObject` (VisualBasic) or other similar COM related functions.

Example

```
Dim objSpy As Application
Set objSpy = CreateObject("XMLSpy.Application")
```

Events

OnBeforeOpenDocument

See also

Event: OnBeforeOpenDocument(*objDialog* as FileSelectionDlg)

Description
This event gets fired whenever a document gets opened via the OpenFile or OpenURL menu command. It is sent after a document file has been selected but before the document gets opened. The file selection dialog object is initialized with the name of the selected document file. You can modify this selection. To continue the opening of the document leave the FileSelectionDlgDialogAction property of *io_objDialog* at its default value spyDialogOK. To abort the opening of the document set this property to spyDialogCancel

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_BeforeOpenDocument(objDialog)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeOpenDocument(objDialog)
{
}
```

XMLSpy IDE Plugin:
```
XMLSpyPlugInOnEvent(26, ...)          // nEventId = 26
```

OnBeforeOpenProject

See also

Event: OnBeforeOpenProject(*objDialog* as FileSelectionDlg)

Description
This event gets fired after a project file has been selected but before the project gets opened. The file selection dialog object is initialized with the name of the selected project file. You can modify this selection. To continue the opening of the project leave the FileSelectionDlgDialogAction property of *io_objDialog* at its default value spyDialogOK. To abort the opening of the project set this property to spyDialogCancel

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_BeforeOpenProject(objDialog)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeOpenProject(objDialog)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent(25, ...)          // nEventId = 25
```

OnDocumentOpened

See also

Event: OnDocumentOpened(*objDocument* as Document)

Description
This event gets fired whenever a document opens in XMLSpy. This can happen due to opening a file with the OpenFile or OpenURL dialog, creating a new file or dropping a file onto XMLSpy. The new document gets passed as parameter. The operation cannot be canceled.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_OpenDocument(objDocument)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_OpenDocument(objDocument)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent(7, ...) // nEventId = 7
```

OnProjectOpened

See also

Event: OnProjectOpened(*objProject* as SpyProject)

Description
This event gets fired whenever a project gets opened in XMLSpy. The new project gets passed as parameter.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_OpenProject(objProject)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_OpenProject(objProject)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent(6, ...) // nEventId = 6
```

ActiveDocument

See also

Property: ActiveDocument as Document

Description
Reference to the active document. If no document is open, ActiveDocument is null (nothing).

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

AddMacroMenuItem

See also

Method: AddMacroMenuItem (*strMacro* as String,*strDisplayText* as String)

Description
Adds a menu item to the **Tools** menu. This new menu item invokes the macro defined by strMacro. See also "Calling macros from XMLSpy".

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1108 Number of macro items is limited to 16 items.

AddXSLT_XQParameter

Method: AddXSLT_XQParameter(name as String,XPath as String)

Description
Adds an XSLT or XQuery parameter. The parameter's name and value are the two arguments of the method.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.
1124 The XPath expression is not set.
1125 Not a QName.
1126 The specified XPath is not valid. Reason for invalidity appended.
1127 A parameter with the submitted name already exists.

Application

See also

Property: Application as Application (read-only)

Description

Accesses the XMLSpy application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

ClearMacroMenu

See also

Method: ClearMacroMenu()

Return Value
None

Description
Removes all menu items from the **Tools** menu. See also Running macros.

Errors
1111 The application object is no longer valid.

CreateXMLSchemaFromDBStructure

See also

Method: CreateXMLSchemaFromDBStructure(pImportSettings as DatabaseConnection,pTables as ElementList)

Description
CreateXMLSchemaFromDBStructure creates from a database specified in pImportSettings for the defined tables in pTables new XML Schema document(s) describing the database tables structure.

The parameter pTables specifies which table structures the XML Schema document should contain. This parameter can be NULL, specifying that all table structures will be exported.

See also GetDataBaseTables.

Errors
1112 Invalid database specified.
1120 Database import failed.

CurrentProject

See also

Property: CurrentProject as SpyProject

Description
Reference to the active document. If no project is open, CurrentProject is null (nothing).

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Dialogs

See also

Property: Dialogs as Dialogs (read-only)

Description
Access the built-in dialogs of XMLSpy.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Documents

See also

Property: Documents as Documents

Description
Collection of all open documents.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Edition

See also

Property: Edition as String

Description
Returns the edition of the application, for example `Altova XMLSpy Enterprise Edition` for the Enterprise edition.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

FindInFiles

See also

Method: FindInFiles(*pSettings* as FindInFilesDlg) as FindInFilesResults

Description
Returns a FindInFilesResults object containing information about the files that matched the specified settings.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetDatabaseImportElementList

See also

Method: GetDatabaseImportElementList(*pImportSettings* as DatabaseConnection) as ElementList

Description
The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of the fields that can be selected for import and the properties ElementListItem.ElementKind are initialized either to *spyXMLDataAttr* or *spyXMLDataElement*, depending on the value passed in DatabaseConnection.AsAttributes. This list serves as a filter to what finally gets imported by a future call to ImportFromDatabase. Use ElementList.RemoveElement to exclude fields from import.

Properties mandatory to be filled out for the database connection are one of DatabaseConnection.File, DatabaseConnection.ADOConnection and DatabaseConnection.ODBCConnection, as well as DatabaseConnection.SQLSelect. Use the property DatabaseConnection.AsAttributes to initialize ElementListItem.ElementKind of the resulting element list to either *spyXMLDataAttr* or *spyXMLDataElement*, respectively.

Example
See example at ImportFromDatabase.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from database failed.
1112 Invalid database specified.
1114 Select statement is missing.
1119 database element list import failed.

GetDatabaseSettings

See also

Method: GetDatabaseSettings() as DatabaseConnection

Description
GetDatabaseSettings creates a new object of database settings. The object is used to specify database connection parameters for the methods GetDatabaseTables, GetDatabaseImportElementList, ImportFromDatabase, ImportFromSchema and ExportToDatabase.

Example
See example of ImportFromDatabase.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetDatabaseTables

See also

Method: GetDatabaseTables(*pImportSettings* as DatabaseConnection) as ElementList

Description
GetDatabaseTables reads the table names from the database specified in *pImportSettings*.

Properties mandatory to be filled out for the database connection are one of DatabaseConnection.File, DatabaseConnection.ADOConnection and DatabaseConnection.ODBCConnection. All other properties are ignored.
The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of tables stored in the specified database. The remaining properties of ElementListItem are unused.

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1112 Invalid database specified.
1113 Error while reading database table information.
1118 Database table query failed.

Example

```
Dim objImpSettings As DatabaseConnection
Set objImpSettings = objSpy.GetDatabaseSettings
objImpSettings.ADOConnection = TxtADO.Text

'store table names in listbox
ListTables.Clear

Dim objList As ElementList
Dim objItem As ElementListItem
On Error GoTo ErrorHandler
Set objList = objSpy.GetDatabaseTables(objImpSettings)

For Each objItem In objList
  ListTables.AddItem objItem.Name
Next
```

GetExportSettings

See also

Method: GetExportSettings() as ExportSettings (read-only)

Description

GetExportSettings creates a new object of common export settings. This object is used to pass the parameters to the export functions and defines the behaviour of the export calls. See also the export functions from Document and the examples at Import and Export.

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetTextImportElementList

See also

Method: GetTextImportElementList(pImportSettings as TextImportExportSettings) as ElementList

Description

GetTextImportElementList retrieves importing information about the text-file as specified in pImportSettings. The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of the fields found in the file. The values of remaining properties are undefined.

If the text-file does not contain a column header, set pImportSettings.HeaderRow to false. The resulting element list will contain general column names like 'Field1' and so on.

See also Import and export of data.

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from database failed.
1115 Error during text element list import. Cannot create parser for import file.
1116 Error during text element list import.

Example

```
'-----------------------------------------------------------
'VBA client code fragment - import selected fields from text file
'-----------------------------------------------------------
Dim objImpSettings As TextImportExportSettings
Set objImpSettings = objSpy.GetTextImportExportSettings

objImpSettings.ImportFile = "C:\ImportMe.txt"
objImpSettings.HeaderRow = False

Dim objList As ElementList
Set objList = objSpy.GetTextImportElementList(objImpSettings)

'exclude first column
objList.RemoveItem 1

Dim objImpDoc As Document
On Error Resume Next
Set objImpDoc = objSpy.ImportFromText(objImpSettings, objList)
CheckForError
```

GetTextImportExportSettings

See also

Method: GetTextImportExportSettings() as TextImportExportSettings (read-only)

Description

GetTextImportExportSettings creates a new object of common import and export settings for text files. See also the example for Application.GetTextImportElementList and Import and Export.

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetXSLT_XQParameterCount

Method: GetXSLT_XQParameterCount() as Long

Description

Returns the number of XSLT and XQuery parameters.

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetXSLT_XQParameterName

Method: GetXSLT_XQParameterName(index as Long) as String

Description
Returns the name of the XSLT or XQuery parameter identified by the supplied index.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetXSLT_XQParameterXPath

Method: GetXSLT_XQParameterXPath(index as Long) as String

Description
Returns the XPath expression of the XSLT or XQuery parameter identified by the supplied index.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

ImportFromDatabase

See also

Method: ImportFromDatabase(pImportSettings as DatabaseConnection,pElementList as ElementList) as Document

Return Value
Creates a new document containing the data imported from the database.

Description
ImportFromDatabase imports data from a database as specified in pImportSettings and creates a new document containing the data imported from the database. Properties mandatory to be filled out are one of DatabaseConnection.File, DatabaseConnection.ADOConnection or DatabaseConnection.ODBCConnection and DatabaseConnection.SQLSelect. Additionally, you can use DatabaseConnection.AsAttributes, DatabaseConnection.ExcludeKeys, DatabaseConnection.IncludeEmptyElements and NumberDateTimeFormat to further parameterize import.

The parameter pElementList specifies which fields of the selected data gets written into the newly created document, and which are created as elements and which as attributes. This parameter can be NULL, specifying that all selected fields will be imported as XML elements.

See GetDatabaseSettings and GetDatabaseImportElementList for necessary steps preceding any import of data from a database.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return

parameter was specified.
1107 Import from database failed.
1112 Invalid database specified.
1114 Select statement is missing.
1117 Transformation to XML failed.
1120 Database import failed.

Example

```
Dim objImpSettings As DatabaseConnection
Set objImpSettings = objSpy.GetDatabaseSettings

objImpSettings.ADOConnection = strADOConnection
objImpSettings.SQLSelect = "SELECT * FROM MyTable"

Dim objDoc As Document
On Error Resume Next
Set objDoc = objSpy.ImportFromDatabase(objImpSettings,
objSpy.GetDatabaseImportElementList(objImpSettings))
'CheckForErrorhere
```

ImportFromSchema

See also

Method: ImportFromSchema(*pImportSettings* as DatabaseConnection,*strTable* as String,*pSchemaDoc* as Document) as Document

Return Value
Creates a new document filled with data from the specified database as specified by the schema definition in *pSchemaDoc*.

Description
ImportFromSchema imports data from a database specified in pImportSettings. Properties mandatory to be filled out are one of DatabaseConnection.File, DatabaseConnection.ADOConnection or DatabaseConnection.ODBCConnection. Additionally, you can use DatabaseConnection.AsAttributes, DatabaseConnection.ExcludeKeys and NumberDateTimeFormat to further parameterize import. All other properties get ignored.

ImportFromSchema does not use and explicit SQL statement to select the data. Instead, it expects a structure definition of the document to create in form of an XML schema document in *pSchemaDoc*. From this definition the database select statement is automatically deduced. Specify in *strTable* the table name of the import root that will become the root node in the new document.

See GetDatabaseSettings and GetDatabaseTables for necessary steps preceding an import from a database based on a schema definition. To create the schema definition file use command 'create database schema' from the 'convert' menu of XMLSpy.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from database failed.
1112 Invalid database specified.
1120 Database import failed.
1121 Could not create validator for the specified schema.
1122 Failed parsing schema for database import.

ImportFromText

See also

Method: ImportFromText(*pImportSettings* as TextImportExportSettings,*pElementList* as ElementList) as Document

Description

ImportFromText imports the text file as specified in pImportSettings. The parameter pElementList can be used as import filter. Either pass the list returned by a previous call to GetTextImportElementList or null to import all columns. To avoid import of unnecessary columns use ElementList.RemoveElement to remove the corresponding field names from pElementList before calling ImportFromText.
The method returns the newly created document containing the imported data. This document is the same as the active document of XMLSpy.

See also Import and export of data.

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from text file failed.
1117 Transformation to XML failed.

Example

```
'-----------------------------------------------------------
'VBA client code fragment - import from text file
'-----------------------------------------------------------
Dim objImpSettings As TextImportExportSettings
Set objImpSettings = objSpy.GetTextImportExportSettings

objImpSettings.ImportFile = strFileName
objImpSettings.HeaderRow = False

Dim objImpDoc As Document
On Error Resume Next
Set objImpDoc = objSpy.ImportFromText(objImpSettings,
objSpy.GetTextImportElementList(objImpSettings))

CheckForError
```

ImportFromWord

See also

Method: ImportFromWord(*strFile* as String) as Document

Description

ImportFromWord imports the MS-Word Document strFile into a new XML document.

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
Import from document failed.

IsAPISupported

See also

Property: IsAPISupported as Boolean

Description
Returns whether the API is supported in this version or not.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

MajorVersion

See also

Property: MajorVersion as Integer

Description
Returns the application version's major number, for example 15 for 2013 versions, and 16 for 2014 versions..

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

MinorVersion

See also

Property: MinorVersion as Integer

Description
Returns the application version's minor number.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

NewProject

See also

Method: NewProject(*strPath* as String,*bDiscardCurrent* as Boolean)

Description
NewProject creates a new project.

If there is already a project open that has been modified and bDiscardCurrent is false, then NewProject() fails.

Errors
1111 The application object is no longer valid.
1102 A project is already open but *bDiscardCurrent* is *true*.
1103 Creation of new project failed.

OpenProject

See also

Method: OpenProject(*strPath* as String,*bDiscardCurrent* as Boolean,*bDialog* as Boolean)

Parameters
strPath
Path and file name of the project to open. Can be empty if bDialog is true.

bDiscardCurrent
Discard currently open project and possible lose changes.

bDialog
Show dialogs for user input.

Return Value
None

Description
OpenProject opens an existing project. If there is already a project open that has been modified and bDiscardCurrent is false, then OpenProject() fails.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1101 Cannot open specified project.
1102 A project is already open but *bDiscardCurrent* is *true*.

Parent

See also

Property: Parent as Application (read-only)

Description
Accesses the XMLSpy application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Quit

See also

Method: Quit()

Return Value
None

Description
This method terminates XMLSpy. All modified documents will be closed without saving the changes. This is also true for an open project.

If XMLSpy was automatically started as an automation server by a client program, the application will not shut down automatically when your client program shuts down if a project or any

document is still open. Use the Quit method to ensure automatic shut-down.

Errors
1111 The application object is no longer valid.

ReloadSettings

See also

Method: ReloadSettings

Return Value

Description
The application settings are reloaded from the registry.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.

RemoveXSLT_XQParameter

Method: RemoveXSLT_XQParameter(index as Long)

Description
Removes the XSLT or XQuery parameter identified by the supplied index.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

RunMacro

See also

Method: RunMacro (*strMacro* as String)

Return Value

Description
Calls the specified macro either from the project scripts (if present) or from the global scripts.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.

ScriptingEnvironment

See also

Property: ScriptingEnvironment as IUnknown (read-only)

Description
Reference to any active scripting environment. This property makes it possible to access the

TypeLibrary of the XMLSpyFormEditor.exe application which is used as the current scripting environment.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

ServicePackVersion

See also

Property: ServicePackVersion as Long

Description
Returns the Service Pack version number of the application. Eg: 1 for 2010 R2 SP1

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

ShowApplication

See also

Method: ShowApplication (*bShow* as Boolean)

Return Value
None

Description
The method shows (bShow = True) or hides (bShow = False) XMLSpy.

Errors
1110 The application object is no longer valid.

ShowFindInFiles

See also

Method: ShowFindInFiles (*pSettings* as FindInFilesDlg) as Boolean

Return Value
Returns false if the user pressed the Cancel button, true otherwise.

Description
Displays the FindInFiles dialog preset with the given settings. The user modifications of the settings are stored in the passed dialog object.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.

ShowForm

See also

Method: ShowForm (*strFormName* as String) as Long

Return Value
Returns zero if the user pressed a Cancel button or the form calls TheView.Cancel().

Description
Displays the form strFormName.

Forms, event handlers and macros can be created with the Scripting Environment. Select "Switch to scripting environment" from the **Tools** menu to invoke the Scripting Environment.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.

Status

See also

Property: Status as ENUMApplicationStatus

Description
Returns the current status of the running application.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

URLDelete

See also

Method: URLDelete (*strURL* as String, *strUser* as String, *strPassword* as String)

Return Value
None

Description
The method deletes the file at the URL strURL.

Errors
1111 The application object is no longer valid.
1109 Error deleting file at specified URL.

URLMakeDirectory

See also

Method: URLMakeDirectory (*strURL* as String, *strUser* as String, *strPassword* as String)

Return Value
None

Description
The method creates a new directory at the URL `strURL`.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter specified.

Visible

See also

Property: `Visible` as `VARIANT_BOOL`

Description
Sets or gets the visibility attribute of XMLSpy. This standard automation property makes usage of `ShowApplication` obsolete.

Errors
1110 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

WarningNumber

See also

Property: `WarningNumber` as integer

Description
Some methods fill the property `WarningNumber` with additional information if an error occurs.

Currently just `Documents.OpenFile` fills this property.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

WarningText

See also

Property: `WarningText` as String

Description
Some methods fill the property `WarningText` with additional information if an error occurs.

Currently just `Documents.OpenFile` fills this property.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

3.2.2 AuthenticContextMenu

The context menu interface provides the mean for the user to customize the context menus shown in Authentic. The interface has the methods listed in this section.

CountItems

Method: `CountItems()` nItems as long

Return Value
Returns the number of menu items.

Errors
2501 Invalid object.

DeleteItem

Method: `DeleteItem(IndexPosition as long)`

Return Value
Deletes the menu item that has the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

GetItemText

Method: `GetItemText(IndexPosition as long)` MenuItemName as string

Return Value
Gets the name of the menu item located at the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

InsertItem

Method: `InsertItem(IndexPosition as long, MenuItemName as string, MacroName as string)`

Return Value
Inserts a user-defined menu item at the position in the menu specified in the first parameter and having the name submitted in the second parameter. The menu item will start a macro, so a valid macro name must be submitted.

Errors
2501 Invalid object
2502 Invalid index
2503 No such macro
2504 Internal error

SetItemText

Method: `SetItemText(IndexPosition as long, MenuItemName as string)`

Return Value

Sets the name of the menu item located at the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

3.2.3 AuthenticDataTransfer

Renamed from DocEditDataTransfer to AuthenticDataTransfer

The `DocEditView` object is renamed to `OldAuthenticView`.
`DocEditSelection` is renamed to `AuthenticSelection`.
`DocEditEvent` is renamed to `AuthenticEvent`.
`DocEditDataTransfer` is renamed to `AuthenticDataTransfer`.

Their usage—except for `AuthenticDataTransfer`—is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interfaces. All functionality available up to now in `DocEditView`, `DocEditSelection`, `DocEditEvent` and `DocEditDataTransfer` is now available via `AuthenticView`, `AuthenticRange` and `AuthenticDataTransfer`. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Methods

`getData`

Properties

`dropEffect`
`ownDrag`
`type`

Description

The events `OnDragOver` and `OnBeforeDrop` provide information about the object being dragged with an instance of type `AuthenticDataTransfer`. It contains a description of the dragged object and its content. The latter is available either as string or a pointer to a COM object supporting the `IUnkown` interface.

dropEffect

See also

Property: `dropEffect` as long

Description
The property stores the drop effect from the default event handler. You can set the drop effect if

you change this value and return TRUE for the event handler (or set AuthenticEvent.cancelBubble to TRUE if you are still using the now obsolete AuthenticEvent interface).

Errors
2101 Invalid address for the return parameter was specified.

getData

See also

Method: getData() as Variant

Description
Retrieve the data associated with the dragged object. Depending on AuthenticDataTransfer.type, that data is either a string or a COM interface pointer of type IUnknown.

Errors
2101 Invalid address for the return parameter was specified.

ownDrag

See also

Property: ownDrag as Boolean (read-only)

Description
The property is TRUE if the current dragging source comes from inside Authentic View.

Errors
2101 Invalid address for the return parameter was specified.

type

See also

Property: type as String (read-only)

Description
Holds the type of data you get with the DocEditDataTransfer.getData method.

Currently supported data types are:

OWN	data from Authentic View itself
TEXT	plain text
UNICODETEXT	plain text as UNICODE

Errors
2101 Invalid address for the return parameter was specified.

3.2.4 AuthenticEventContext

The EventContext interface gives access to many properties of the context in which a macro is executed.

EvaluateXPath

Method: EvaluateXPath (strExpression as string) as strValue as string

Return Value
The method evaluates the XPath expression in the context of the node within which the event was triggered and returns a string.

Description
EvaluateXPath() executes an XPath expressions with the given event context. The result is returned as string, in the case of a sequence it is a space-separated string.

Errors
2201 Invalid object.
2202 No context.
2209 Invalid parameter.
2210 Internal error.
2211 XPath error.

GetEventContextType

Method: GetEventContextType () Type as AuthenticEventContextType enumeration

Return Value
Returns the context node type.

Description
GetEventContextType allows the user to determine whether the macro is in an XML node or in an XPath atomic item context. The enumeration AuthenticEventContextType is defined as follows:

```
authenticEventContextXML,
authenticEventContextAtomicItem,
authenticEventContextOther
```

If the context is a normal XML node, the GetXMLNode() function gives access to it (returns NULL if not).

Errors
2201 Invalid object.
2202 No context.
2209 Invalid parameter.

GetNormalizedTextValue

Method: GetNormalizedTextValue() strValue as string

Return Value
Returns the value of the current node as string

Errors
2201 Invalid object.
2202 No context.
2203 Invalid context
2209 Invalid parameter.

GetVariableValue

Method: `GetVariableValue(strName as string) strValue as string`

Return Value
Gets the value of the variable submitted as the parameter.

Description
`GetVariableValue` gets the variable's value in the scope of the context.

```
nZoom = parseInt(AuthenticView.EventContext.GetVariableValue('Zoom'));
if (nZoom > 1)
{
    AuthenticView.EventContext.SetVariableValue('Zoom',nZoom - 1);
}
```

Errors
2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2209 Invalid parameter

GetXMLNode

Method: `GetXMLNode()` Node as XMLData object

Return Value
Returns the context XML node or `NULL`

Errors
2201 Invalid object.
2202 No context.
2203 Invalid context
2209 Invalid parameter.

IsAvailable

Method: `IsAvailable()` as Boolean

Return Value
Returns true if `EventContext` is set, false otherwise.

Errors
2201 Invalid object.

SetVariableValue

Method: `SetVariableValue(strName as string, strValue as string)`

Return Value
Sets the value (second parameter) of the variable submitted in the first parameter.

Description

`SetVariableValue` sets the variable's value in the scope of the context.

```
nZoom = parseInt(AuthenticView.EventContext.GetVariableValue('Zoom'));
if (nZoom > 1)
{
    AuthenticView.EventContext.SetVariableValue('Zoom',nZoom - 1);
}
```

Errors

2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2207 Variable read-only
2208 No modification allowed

3.2.5 AuthenticRange

See also

The first table lists the properties and methods of `AuthenticRange` that can be used to navigate through the document and select specific portions.

Properties		Methods
Application	Clone	MoveBegin
FirstTextPosition	CollapsToBegin	MoveEnd
FirstXMLData	CollapsToEnd	NextCursorPosition
FirstXMLDataOffset	ExpandTo	PreviousCursorPosition
LastTextPosition	Goto	Select
LastXMLData	GotoNext	SelectNext
LastXMLDataOffset	GotoPrevious	SelectPrevious
Parent	IsEmpty	SetFromRange
	IsEqual	

The following table lists the content modification methods, most of which can be found on the right/button mouse menu.

Properties	Edit operations	Dynamic table operations
Text	Copy	AppendRow
	Cut	DeleteRow
	Delete	DuplicateRow
	IsCopyEnabled	InsertRow
	IsCutEnabled	IsFirstRow
	IsDeleteEnabled	IsInDynamicTable
	IsPasteEnabled	IsLastRow
	Paste	MoveRowDown
		MoveRowUp

The following methods provide the functionality of the Authentic entry helper windows for range objects.

Operations of the entry helper windows

Elements	Attributes	Entities
CanPerformActionWith	GetElementAttributeValue	GetEntityNames
CanPerformAction	GetElementAttributeNames	InsertEntity
PerformAction	GetElementHierarchy	

HasElementAttribute
IsTextStateApplied
SetElementAttributeValue

Description
AuthenticRange objects are the 'cursor' selections of the automation interface. You can use them to point to any cursor position in the Authentic view, or select a portion of the document. The operations available for AuthenticRange objects then work on this selection in the same way, as the corresponding operations of the user interface do with the current user interface selection. The main difference is that you can use an arbitrary number of AuthenticRange objects at the same time, whereas there is exactly one cursor selection in the user interface.

To get to an initial range object use AuthenticView.Selection, to obtain a range corresponding with the current cursor selection in the user interface. Alternatively, some trivial ranges are accessible via the read/only properties AuthenticView.DocumentBegin, AuthenticView.DocumentEnd, and AuthenticView.WholeDocument. The most flexible method is AuthenticView.Goto, which allows navigation to a specific portion of the document within one call. For more complex selections, combine the above, with the various navigation methods on range objects listed in the first table on this page.

Another method to select a portion of the document is to use the position properties of the range object. Two positioning systems are available and can be combined arbitrarily:

- **Absolute** text cursor positions, starting with position 0 at the document beginning, can be set and retrieved for the beginning and end of a range. For more information see FirstTextPosition and LastTextPosition. This method requires complex internal calculations and should be used with care.
- The **XMLData** element and a text position inside this element, can be set and retrieved for the beginning and end of a range. For more information see FirstXMLData, FirstXMLDataOffset, LastXMLData, and LastXMLDataOffset. This method is very efficient but requires knowledge on the underlying document structure. It can be used to locate XMLData objects and perform operations on them otherwise not accessible through the user interface.

Modifications to the document content can be achieved by various methods:

- The Text property allows you to retrieve the document text selected by the range object. If set, the selected document text gets replaced with the new text.
- The standard document edit functions Cut, Copy, Paste and Delete.
- Table operations for tables that can grow dynamically.
- Methods that map the functionality of the Authentic entry helper windows.
- Access to the XMLData objects of the underlying document to modify them directly.

AppendRow

See also

Method: AppendRow() as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a new row at the end of the selected table. The selection of the range is modified to point to the beginning of the new row. The function returns *true* if the append operation was successful, otherwise *false*.

Errors
2001 The authentic range object or its related view object is no

longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
'-----------------------------------------------------------
'Scripting environment - VBScript
'Append row at end of current dynamically growable table
'-----------------------------------------------------------
Dim objRange
'we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

'check if we can insert something
If objRange.IsInDynamicTable Then
	objRange.AppendRow
	'objRange points to beginning of new row
	objRange.Select
End If
```

Application

See also

Property: Application as Application (read-only)

Description
Accesses the XMLSpy application object.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

CanPerformAction

See also

Method: CanPerformAction (*eAction* as SPYAuthenticActions, *strElementName* as String) as Boolean

Description
CanPerformAction and its related methods enable access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content, without having to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If the location can be found, the method returns *True*, otherwise it returns *False*.

HINT: To find out all valid element names for a given action, use CanPerformActionWith.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples
See PerformAction.

CanPerformActionWith

See also

Method: CanPerformActionWith (*eAction* as SPYAuthenticActions, *out_arrElementNames* as Variant)

Description
PerformActionWith and its related methods, enable access to the entry-helper functions of Authentic. These function allows easy and consistent modification of the document content without having to know exactly where the modification will take place.

This method returns an array of those element names that the specified action can be performed with.

HINT: To apply the action use CanPerformActionWith.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples
See PerformAction.

Clone

See also

Method: Clone() as AuthenticRange

Description
Returns a copy of the range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

CollapsToBegin

See also

Method: CollapsToBegin() as AuthenticRange

Description
Sets the end of the range object to its begin. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

CollapsToEnd

See also

Method: CollapsToEnd() as AuthenticRange

Description
Sets the beginning of the range object to its end. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Copy

See also

Method: Copy() as Boolean

Description
Returns *False* if the range contains no portions of the document that may be copied.
Returns *True* if text, and in case of fully selected XML elements the elements as well, has been copied to the copy/ paste buffer.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Cut

See also

Method: Cut() as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted.
Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document and saved in the copy/ paste buffer.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Delete

See also

Method: Delete() as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted.
Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document.

Errors

2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

DeleteRow

See also

Method: DeleteRow () as Boolean

Description

If the beginning of the range is inside a dynamic table, this method deletes the selected row. The selection of the range gets modified to point to the next element after the deleted row. The function returns *true,* if the delete operation was successful, otherwise *false*.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
'-----------------------------------------------------------
'Scripting environment - VBScript
'Delete selected row from dynamically growing table
'-----------------------------------------------------------
Dim objRange
'we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

'check if we are in a table
If objRange.IsInDynamicTable Then
      objRange.DeleteRow
End If
```

DuplicateRow

See also

Method: DuplicateRow () as Boolean

Description

If the beginning of the range is inside a dynamic table, this method inserts a duplicate of the current row after the selected one. The selection of the range gets modified to point to the beginning of the new row. The function returns *true* if the duplicate operation was successful, otherwise *false*.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
'-----------------------------------------------------------
'Scripting environment - VBScript
'duplicate row in current dynamically growable table
'-----------------------------------------------------------
Dim objRange
'we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection
```

```
'check if we can insert something
If objRange.IsInDynamicTable Then
	objRange.DuplicateRow
	'objRange points to beginning of new row
	objRange.Select
End If
```

EvaluateXPath

Method: EvaluateXPath (strExpression as string) strValue as string

Return Value
The method returns a string

Description
EvaluateXPath() executes an XPath expressions with the context node being the beginning of the range selection. The result is returned as string, in the case of a sequence it is a space-separated string. If XML context node is irrelevant, the user may provide any node, like AuthenticView.XMLDataRoot.

Errors
2001 Invalid object
2005 Invalid parameter
2008 Internal error
2202 Missing context node
2211 XPath error

ExpandTo

See also

Method: ExpandTo (*eKind* as SPYAuthenticElementKind), as AuthenticRange

Description
Selects the whole element of type eKind, that starts at, or contains, the first cursor position of the range. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Range expansion would be beyond end of document.
2005 Invalid address for the return parameter was specified.

FirstTextPosition

See also

Property: FirstTextPosition as Long

Description
Set or get the left-most text position index of the range object. This index is always less or equal to LastTextPosition. Indexing starts with 0 at document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decrementing the test position by 1 has the same effect as the cursor-left key.

If you set FirstTextPosition to a value greater than the current LastTextPosition,

LastTextPosition gets set to the new FirstTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2006 A text position outside the document was specified.

Examples

```
'----------------------------------------
'Scripting environment - VBScript
'----------------------------------------
Dim objAuthenticView
'we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

'let's create a range that selects the whole document
'in an inefficient way
Dim objRange
'we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

'let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
        MsgBox "Test using direct text cursor positioning was ok"
Else
        MsgBox "Ooops!"
End If
```

FirstXMLData

See also

Property: FirstXMLData as XMLData

Description

Set or get the first XMLData element in the underlying document that is partially, or completely selected by the range. The exact beginning of the selection is defined by the FirstXMLDataOffset attribute.

Whenever you set FirstXMLData to a new data object, FirstXMLDataOffset gets set to the first cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set FirstXMLData / FirstXMLDataOffset selects a position greater then the current LastXMLData / LastXMLDataOffset, the latter gets moved to the new start position.

HINT: You can use the FirstXMLData and LastXMLData properties, to directly access and manipulate the underlying XML document in those cases where the methods available with the AuthenticRange object are not sufficient.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.

2008 Internal error
2009 The XMLData object cannot be accessed.

Examples

```
'-----------------------------------------------
'Scripting environment - VBScript
'show name of currently selected XMLData element
'-----------------------------------------------
Dim objAuthenticView
'we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objXmlData
Set objXMLData = objAuthenticView.Selection.FirstXMLData
'authentic view adds a 'text' child element to elements
'of the document which have content. So we have to go one
'element up.
Set objXMLData = objXMLData.Parent
MsgBox "Current selection selects element " & objXMLData.Name
```

FirstXMLDataOffset

See also

Property: FirstXMLDataOffset as Long

Description

Set or get the cursor position offset inside FirstXMLData element for the beginning of the range. Offset positions are based on the characters returned by the Text property, and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on screen. Although the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If FirstXMLData / FirstXMLDataOffset selects a position after the current LastXMLData / LastXMLDataOffset, the latter gets moved to the new start position.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

```
'-----------------------------------------------
'Scripting environment - VBScript
'Select the complete text of an XMLData element
'using XMLData based selection and ExpandTo
'-----------------------------------------------
Dim objAuthenticView
'we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

'first we use the XMLData based range properties
'to select all text of the first XMLData element
'in the current selection
```

```
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  'start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  'select only one element
objRange.LastXMLDataOffset = -1  'select till its end

'the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

'were we successful?
If objRange.IsEqual(objRange2) Then
	objRange.Select()
Else
	MsgBox "Oops"
End If
```

GetElementAttributeNames

See also

Method: GetElementAttributeNames (*strElementName* as String, *out_arrAttributeNames* as Variant)

Description
Retrieve the names of all attributes for the enclosing element with the specified name. Use the element/attribute pairs, to set or get the attribute value with the methods GetElementAttributeValue and SetElementAttributeValue.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue.

GetElementAttributeValue

See also

Method: GetElementAttributeValue (*strElementName* as String, *strAttributeName* as String) as String

Description
Retrieve the value of the attribute specified in strAttributeName, for the element identified with strElementName. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use GetElementAttributeNames, or HasElementAttribute.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid attribute name was specified.
Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue.

GetElementHierarchy

See also

Method: GetElementHierarchy (*out_arrElementNames* as Variant)

Description
Retrieve the names of all XML elements that are parents of the current selection. Inner elements get listed before enclosing elements. An empty list is returned whenever the current selection is not inside a single XMLData element.

The names of the element hierarchy, together with the range object uniquely identify XMLData elements in the document. The attributes of these elements can be directly accessed by GetElementAttributeNames, and related methods.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

C# Examples

```
'-------------------------------------------
'C#
'-------------------------------------------

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            XMLSpyLib.Application app = new XMLSpyLib.Application();

            app.ShowApplication(true);

            XMLSpyLib.AuthenticView view = app.ActiveDocument.AuthenticView;
            XMLSpyLib.AuthenticRange range = view.DocumentBegin;

            object o = null;
            range.GetElementHierarchy(ref o);

            object[] elements = (object[])o;

            foreach (string e in elements)
            {
                Console.WriteLine(e);
            }
        }
    }
}
```

Also see: SetElementAttributeValue.

GetEntityNames

See also

Method: GetEntityNames (*out_arrEntityNames* as Variant)

Description
Retrieve the names of all defined entities. The list of retrieved entities is independent of the current selection, or location. Use one of these names with the InsertEntity function.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples
See: GetElementHierarchy and InsertEntity.

GetVariableValue

Method: GetVariableValue(strName as string) strVal as string

Return Value
Gets the value of the variable named as the method's parameter.

Errors
2001 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2209 Invalid parameter

Goto

See also

Method: Goto (*eKind* as SPYAuthenticElementKind, *nCount* as Long, *eFrom* as SPYAuthenticDocumentPosition) as AuthenticRange

Description
Sets the range to point to the beginning of the nCount element of type eKind. The start position is defined by the parameter eFrom.

Use positive values for nCount to navigate to the document end. Use negative values to navigate to the beginning of the document. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2004 Target lies before begin of document.
2005 Invalid element kind specified.
Invalid start position specified.
Invalid address for the return parameter was specified.

GotoNext

See also

Method: GotoNext(*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description
Sets the range to the beginning of the next element of type eKind. The method returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples

```
'-------------------------------------------
'Scripting environment - VBScript
'Scan through the whole document word-by-word
'-------------------------------------------
Dim objAuthenticView
'we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False

On Error Resume Next
While Not bEndOfDocument
	objRange.GotoNext(spyAuthenticWord).Select
	If (Err.number - vbObjectError) = 2003) Then
		bEndOfDocument = True
		Err.Clear
	ElseIf (Err.number <> 0) Then
		Err.Raise 'forward error
	End If
Wend
```

GotoNextCursorPosition

See also

Method: GotoNextCursorPosition() as AuthenticRange

Description
Sets the range to the next cursor position after its current end position. Returns the modified object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2005 Invalid address for the return parameter was specified.

GotoPrevious

See also

Method: GotoPrevious (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description
Sets the range to the beginning of the element of type eKind which is before the beginning of the current range. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2004 Target lies before beginning of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples

```
'-------------------------------------------
'Scripting environment - VBScript
'Scan through the whole document tag-by-tag
'-------------------------------------------
Dim objAuthenticView
'we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
Dim bEndOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
	objRange.GotoPrevious(spyAuthenticTag).Select
	If (Err.number - vbObjectError) = 2004) Then
		bBeginOfDocument = True
		Err.Clear
	ElseIf (Err.number <> 0) Then
		Err.Raise 'forward error
	End If
Wend
```

GotoPreviousCursorPosition

See also

Method: GotoPreviousCursorPosition() as AuthenticRange

Description
Set the range to the cursor position immediately before the current position. Returns the modified object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2004 Target lies before begin of document.
2005 Invalid address for the return parameter was specified.

HasElementAttribute

See also

Method: HasElementAttribute (*strElementName* as String, *strAttributeName* as String) as Boolean

Description
Tests if the enclosing element with name strElementName, supports the attribute specified in strAttributeName.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid address for the return parameter was specified.

InsertEntity

See also

Method: InsertEntity (*strEntityName* as String)

Description
Replace the ranges selection with the specified entity. The specified entity must be one of the entity names returned by GetEntityNames.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Unknown entry name was specified.

Examples

```
'-----------------------------------------------------------
'Scripting environment - VB Script
'Insert the first entity in the list of available entities
'-----------------------------------------------------------
Dim objRange
'we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

'first we get the names of all available entities as they
'are shown in the entry helper of XMLSpy
Dim arrEntities
objRange.GetEntityNames arrEntities

'we insert the first one of the list
If UBound(arrEntities) >= 0 Then
	objRange.InsertEntity arrEntities(0)
Else
	MsgBox "Sorry, no entities are available for this document"
End If
```

InsertRow

See also

Method: InsertRow () as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a new row before the

current one. The selection of the range, gets modified to point to the beginning of the newly inserted row. The function returns *true* if the insert operation was successful, otherwise *false*.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
'-----------------------------------------------------------------
'Scripting environment - VBScript
'Insert row at beginning of current dynamically growing table
'-----------------------------------------------------------------
Dim objRange
'we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

'check if we can insert something
If objRange.IsInDynamicTable Then
      objRange.InsertRow
      'objRange points to beginning of new row
      objRange.Select
End If
```

IsCopyEnabled

See also

Property: IsCopyEnabled as Boolean (read-only)

Description
Checks if the copy operation is supported for this range.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsCutEnabled

See also

Property: IsCutEnabled as Boolean (read-only)

Description
Checks if the cut operation is supported for this range.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsDeleteEnabled

See also

Property: IsDeleteEnabled as Boolean (read-only)

Description
Checks if the delete operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsEmpty

See also

Method: IsEmpty() as Boolean

Description
Tests if the first and last position of the range are equal.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsEqual

See also

Method: IsEqual(*objCmpRange* as AuthenticRange) as Boolean

Description
Tests if the start and end of both ranges are the same.

Errors
2001 One of the two range objects being compared, is invalid.
2005 Invalid address for a return parameter was specified.

IsFirstRow

See also

Property: IsFirstRow as Boolean (read-only)

Description
Test if the range is in the first row of a table. Which table is taken into consideration depends on the extend of the range. If the selection exceeds a single row of a table, the check is if this table is the first element in an embedding table. See the entry helpers of the user manual for more information.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsInDynamicTable

See also

Method: IsInDynamicTable() as Boolean

Description
Test if the whole range is inside a table that supports the different row operations like 'insert',

'append', duplicate, etc.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsLastRow

See also

Property: IsLastRow as Boolean (read-only)

Description
Test if the range is in the last row of a table. Which table is taken into consideration depends on the extend of the range. If the selection exceeds a single row of a table, the check is if this table is the last element in an embedding table. See the entry helpers of the user manual for more information.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsPasteEnabled

See also

Property: IsPasteEnabled as Boolean (read-only)

Description
Checks if the paste operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsSelected

Property: IsSelected as Boolean

Description
Returns true() if selection is present. The selection range still can be empty: that happens when e.g. only the cursor is set.

IsTextStateApplied

See also

Method: IsTextStateApplied (*istrElementName* as String) as Boolean

Description
Checks if all the selected text is embedded into an XML Element with name istrElementName. Common examples for the parameter istrElementName are "strong", "bold" or "italic".

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

LastTextPosition

See also

Property: LastTextPosition as Long

Description

Set or get the rightmost text position index of the range object. This index is always greater or equal to FirstTextPosition. Indexing starts with 0 at the document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decreasing the test position by 1 has the same effect as the cursor-left key.

If you set LastTextPosition to a value less then the current FirstTextPosition, FirstTextPosition gets set to the new LastTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2006 A text position outside the document was specified.

Examples

```
'----------------------------------------
'Scripting environment - VBScript
'----------------------------------------
Dim objAuthenticView
'we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

'let's create a range that selects the whole document
'in an inefficient way
Dim objRange
'we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

'let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
	MsgBox "Test using direct text cursor positioning was ok"
Else
	MsgBox "Oops!"
End If
```

LastXMLData

See also

Property: LastXMLData as XMLData

Description
Set or get the last XMLData element in the underlying document that is partially or completely selected by the range. The exact end of the selection is defined by the LastXMLDataOffset attribute.

Whenever you set LastXMLData to a new data object, LastXMLDataOffset gets set to the last cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set LastXMLData / LastXMLDataOffset, select a position less then the current FirstXMLData / FirstXMLDataOffset, the latter gets moved to the new end position.

HINT: You can use the FirstXMLData and LastXMLData properties to directly access and manipulate the underlying XML document in those cases, where the methods available with the AuthenticRange object are not sufficient.

Errors
2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2008 Internal error
2009 The XMLData object cannot be accessed.

LastXMLDataOffset

See also

Property: LastXMLDataOffset as Long

Description
Set or get the cursor position inside LastXMLData element for the end of the range.

Offset positions are based on the characters returned by the Text property and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on the screen. Although, the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If LastXMLData / LastXMLDataOffset selects a position before FirstXMLData / FirstXMLDataOffset, the latter gets moved to the new end position.

Errors
2001 The authentic range object, or its related view object is not valid.
2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

```
'-------------------------------------------
'Scripting environment - VBScript
'Select the complete text of an XMLData element
'using XMLData based selection and ExpandTo
```

```
'-----------------------------------------------
Dim objAuthenticView
'we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

'first we use the XMLData based range properties
'to select all text of the first XMLData element
'in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  'start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  'select only one element
objRange.LastXMLDataOffset = -1  'select till its end

'the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

'were we successful?
If objRange.IsEqual(objRange2) Then
	objRange.Select()
Else
	MsgBox "Ooops"
End If
```

MoveBegin

See also

Method: MoveBegin (*eKind* as SPYAuthenticElementKind, *nCount* as Long) as AuthenticRange

Description
Move the beginning of the range to the beginning of the nCount element of type eKind. Counting starts at the current beginning of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The end of the range stays unmoved, unless the new beginning would be larger than it. In this case, the end is moved to the new beginning. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before beginning of document.
2005	Invalid element kind specified.
	Invalid address for the return parameter was specified.

MoveEnd

See also

Method: MoveEnd (*eKind* as SPYAuthenticElementKind, *nCount* as Long) as AuthenticRange

Description
Move the end of the range to the begin of the nCount element of type eKind. Counting starts at the current end of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The beginning of the range stays unmoved, unless the new end would be less than it. In this case, the beginning gets moved to the new end. The method

returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2004 Target lies before begin of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

MoveRowDown

See also

Method: `MoveRowDown()` as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the last row in this table, this method swaps this row with the row immediately below. The selection of the range moves with the row, but does not otherwise change. The function returns *true* if the move operation was successful, otherwise *false*.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

MoveRowUp

See also

Method: `MoveRowUp()` as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the first row in this table, this method swaps this row with the row above. The selection of the range moves with the row, but does not change otherwise. The function returns *true* if the move operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `AuthenticView` (read-only)

Description
Access the view that owns this range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Paste

See also

Method: Paste() as Boolean

Description
Returns *False* if the copy/ paste buffer is empty, or its content cannot replace the current selection.

Otherwise, deletes the current selection, inserts the content of the copy/ paste buffer, and returns *True*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

PerformAction

See also

Method: PerformAction (*eAction* as SPYAuthenticActions, *stElementName* as String) as Boolean

Description
PerformAction and its related methods, give access to the entry- helper functions of Authentic. This function allows easy and consistent modification of the document content without a need to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If no such location can be found, the method returns *False*. Otherwise, the document gets modified and the range points to the beginning of the modification.

HINT: To find out element names that can be passed as the second parameter use CanPerformActionWith.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples

```
'-------------------------------------------
'Scripting environment - VBScript
'Insert the innermost element
'-------------------------------------------
Dim objRange
'we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

'we determine the elements that can be inserted at the current position
Dim arrElements()
objRange.CanPerformActionWith spyAuthenticInsertBefore, arrElements

'we insert the first (innermost) element
If UBound(arrElements) >= 0 Then
	objRange.PerformAction spyAuthenticInsertBefore, arrElements(0)
	'objRange now points to the beginning of the inserted element
	'we set a default value and position at its end
	objRange.Text = "Hello"
```

```
        objRange.ExpandTo(spyAuthenticTag).CollapsToEnd().Select
Else
        MsgBox "Can't insert any elements at current position"
End If
```

Select

See also

Method: Select()

Description
Makes this range the current user interface selection. You can achieve the same result using: '*objRange.Parent.Selection = objRange*'

Errors
2001 The authentic range object or its related view object is no longer valid.

Examples
```
'----------------------------------------
'Scripting environment - VBScript
'----------------------------------------
Dim objAuthenticView
'we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

'set current selection to end of document
objAuthenticView.DocumentEnd.Select()
```

SelectNext

See also

Method: SelectNext(*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description
Selects the element of type eKind after the current end of the range. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples
```
'-------------------------------------------
'Scripting environment - VBScript
'Scan through the whole document word-by-word
'-------------------------------------------
Dim objAuthenticView
'we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False

On Error Resume Next
While Not bEndOfDocument
```

```
        objRange.SelectNext(spyAuthenticWord).Select
        If (Err.number - vbObjectError) = 2003) Then
                bEndOfDocument = True
                Err.Clear
        ElseIf (Err.number <> 0) Then
                Err.Raise  'forward error
        End If
Wend
```

SelectPrevious

See also

Method: GotoPrevious (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description

Selects the element of type eKind before the current beginning of the range. The method returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2004 Target lies before begin of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples

```
'-------------------------------------------
'Scripting environment - VBScript
'Scan through the whole document tag-by-tag
'-------------------------------------------
Dim objAuthenticView
'we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
Dim bEndOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
        objRange.SelectPrevious(spyAuthenticTag).Select
        If (Err.number - vbObjectError) = 2004) Then
                bBeginOfDocument = True
                Err.Clear
        ElseIf (Err.number <> 0) Then
                Err.Raise  'forward error
        End If
Wend
```

SetElementAttributeValue

See also

Method: SetElementAttributeValue (*strElementName* as String, *strAttributeName* as String, *strAttributeValue* as String)

Description

Set the value of the attribute specified in strAttributeName for the element identified with

strElementName. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use GetElementAttributeNames, or HasElementAttribute.

Errors

2001 The authentic range object or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid attribute name was specified.
Invalid attribute value was specified.

Examples

```
'-------------------------------------------
'Scripting environment - VBScript
'Get and set element attributes
'-------------------------------------------
Dim objRange
'we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

'first we find out all the elements below the beginning of the range
Dim arrElements
objRange.GetElementHierarchy arrElements

If IsArray(arrElements) Then
	If UBound(arrElements) >= 0 Then
		'we use the top level element and find out its valid attributes
		Dim arrAttrs()
		objRange.GetElementAttributeNames arrElements(0), arrAttrs

		If UBound(arrAttrs) >= 0 Then
			'we retrieve the current value of the first valid attribute
			Dim strAttrVal
			strAttrVal = objRange.GetElementAttributeValue (arrElements(0), arrAttrs(0))
			msgbox "current value of " & arrElements(0) & "//" & arrAttrs(0) & " is: " &
strAttrVal

			'we change this value and read it again
			strAttrVal = "Hello"
			objRange.SetElementAttributeValue arrElements(0), arrAttrs(0), strAttrVal
			strAttrVal = objRange.GetElementAttributeValue (arrElements(0), arrAttrs(0))
			msgbox "new value of " & arrElements(0) & "//" & arrAttrs(0) & " is: " &
strAttrVal
		End If
	End If
End If
```

SetFromRange

See also

Method: SetFromRange (*objSrcRange* as AuthenticRange)

Description

Sets the range object to the same beginning and end positions as objSrcRange.

Errors

2001 One of the two range objects, is invalid.
2005 Null object was specified as source object.

SetVariableValue

Method: SetVariableValue(strName as string, strValue as string)

Return Value
Sets the value (second parameter) of the variable named in the first parameter.

Errors
2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2207 Variable read-only
2208 No modification allowed

Text

See also

Property: Text as String

Description
Set or get the textual content selected by the range object.

The number of characters retrieved are not necessarily identical, as there are text cursor positions between the beginning and end of the selected range. Most document elements support an end cursor position different to the beginning cursor position of the following element. Drop-down lists maintain only one cursor position, but can select strings of any length. In the case of radio buttons and check boxes, the text property value holds the string of the corresponding XML element.

If the range selects more then one element, the text is the concatenation of the single texts. XML entities are expanded so that '&' is expected as '&'.

Setting the text to the empty string, does not delete any XML elements. Use Cut, Delete or PerformAction instead.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for a return parameter was specified.

3.2.6 AuthenticView

See also

Properties	Methods	Events
Application	Goto	OnBeforeCopy
AsXMLString	IsRedoEnabled	OnBeforeCut
DocumentBegin	IsUndoEnabled	OnBeforeDelete
DocumentEnd	Print	OnBeforeDrop
Event	Redo	OnBeforePaste
MarkupVisibility	Undo	OnDragOver
Parent	UpdateXMLInstanceEntities	OnKeyBoardEvent
Selection		OnMouseEvent
XMLDataRoot		OnSelectionChanged

WholeDocument

Description
AuthenticView and its child objects AuthenticRange and AuthenticDataTransfer provide you with an interface for Authentic View, which allow easy and consistent modification of document contents. These interfaces replace the following interfaces which are marked now as **obsolete**:

OldAuthenticView (old name was DocEditView)
AuthenticSelection (old name was DocEditSelection, superseded by AuthenticRange)
AuthenticEvent (old name was DocEditEvent)

AuthenticView gives you easy access to specific features such as printing, the multi-level undo buffer, and the current cursor selection, or position.

AuthenticView uses objects of type AuthenticRange to make navigation inside the document straight-forward, and to allow for the flexible selection of logical text elements. Use the properties DocumentBegin, DocumentEnd, or WholeDocument for simple selections, while using the Goto method for more complex selections. To navigate relative to a given document range, see the methods and properties of the AuthenticRange object.

Examples

```
'----------------------------------------
'XMLSpy scripting environment - VBScript
'secure access to authentic view object
'----------------------------------------
Dim objDocument
Set objDocument = Application.ActiveDocument
If (Not objDocument Is Nothing) Then
	'we have an active document, now check for view mode
	If (objDocument.CurrentViewMode <> spyViewAuthentic) Then
		If (Not objDocument.SwitchViewMode (spyViewAuthentic)) Then
			MsgBox "Active document does not support authentic view mode"
		Else
			'now it is safe to access the authentic view object
			Dim objAuthenticView
			Set objAuthenticView = objDocument.AuthenticView
			'now use the authentic view object

		End If
	End If
Else
	MsgBox "No document is open"
End If
```

Events

OnBeforeCopy

See also

Event: OnBeforeCopy() as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticBeforeCopy()
	'On_AuthenticBeforeCopy = False  'to disable operation
End Function
```

Scripting environment - JScript:

```
function On_AuthenticBeforeCopy()
```

```
{
	// return false; /* to disable operation */
}
```

IDE Plugin:

```
IXMLSpyPlugIn.OnEvent(21, ...)		// nEventId = 21
```

Description

This event gets triggered before a copy operation gets performed on the document. Return *True* (or nothing) to allow copy operation. Return *False* to disable copying.

OnBeforeCut

See also

Event: `OnBeforeCut()` as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticBeforeCut()
	'On_AuthenticBeforeCut = False  'to disable operation
End Function
```

Scripting environment - JScript:

```
function On_AuthenticBeforeCut()
{
	// return false; /* to disable operation */
}
```

IDE Plugin:

```
IXMLSpyPlugIn.OnEvent(20, ...)		// nEventId = 20
```

Description

This event gets triggered before a cut operation gets performed on the document. Return *True* (or nothing) to allow cut operation. Return *False* to disable operation.

OnBeforeDelete

See also

Event: `OnBeforeDelete()` as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticBeforeDelete()
	'On_AuthenticBeforeDelete = False  'to disable operation
End Function
```

Scripting environment - JScript:

```
function On_AuthenticBeforeDelete()
{
	// return false; /* to disable operation */
}
```

IDE Plugin:

```
IXMLSpyPlugIn.OnEvent(22, ...)		// nEventId = 22
```

Description

This event gets triggered before a delete operation gets performed on the document. Return *True* (or nothing) to allow delete operation. Return *False* to disable operation.

OnBeforeDrop

See also

Event: OnBeforeDrop (*inXPos* as Long, *inYPos* as Long, *ippRange* as AuthenticRange, *ippData* as cancelBoolean

Scripting environment - VBScript:
```
Function On_AuthenticBeforeDrop(nXPos, nYPos, objRange, objData)
        'On_AuthenticBeforeDrop = False  'to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticBeforeDrop(nXPos, nYPos, objRange, objData)
{
        // return false; /* to disable operation */
}
```

IDE Plugin:
```
IXMLSpyPlugIn.OnEvent(11, ...)          // nEventId = 11
```

Description
This event gets triggered whenever a previously dragged object gets dropped inside the application window. All event related information gets passed as parameters.

The first two parameters specify the mouse position at the time when the event occurred. The parameter *objRange* passes a range object that selects the XML element below the mouse position. The value of this parameter might be *NULL*. Be sure to check before you access the range object. The parameter *objData* allows to access information about the object being dragged.

Return *False* to cancle the drop operation. Return *True* (or nothing) to continue normal operation.

Examples
```
'--------------------------------------------------------------------------------
'VB code snippet - connecting to object level events
'--------------------------------------------------------------------------------
'access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("","XMLSpy.Application")

'this is the event callback routine connected to the OnBeforeDrop
'event of object objView
Private Function objView_OnBeforeDrop(ByVal inXPos As Long, ByVal inYPos As Long,
                          ByVal ippRange As IAuthenticRange,
                          ByVal ippData As IAuthenticDataTransfer) As Boolean

   If (Not ippRange Is Nothing) Then
      MsgBox ("Dropping on content is prohibited");
      Return False;
   Else
      Return True;
   End If
End Function

'use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

'continue here with something useful...
'and serve the windows message loop
```

OnBeforePaste

See also

Event: OnBeforePaste (*objData* as Variant, *strType* as String) as Boolean

Scripting environment - VBScript:
```
Function On_AuthenticBeforePaste(objData, strType)
	'On_AuthenticBeforePaste = False  'to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticBeforePaste(objData, strType)
{
	// return false; /* to disable operation */
}
```

IDE Plugin:
```
IXMLSpyPlugIn.OnEvent(19, ...)  // nEventId = 19
```

Description
This event gets triggered before a paste operation gets performed on the document. The parameter *strType* is one of "TEXT", "UNICODETEXT" or "IUNKNOWN". In the first two cases *objData* contains a string representation of the object that will be pasted. In the later case, *objData* contains a pointer to an IUnknown COM interface.

Return *True* (or nothing) to allow paste operation. Return *False* to disable operation.

OnBeforeSave

Event: OnBeforeSave (SaveAs flag) as Boolean

Description: OnBeforeSave gives the opportunity to e.g. warn the user about overwriting the existing XML document, or to make the document read-only when specific circumstances are not met. The event will be fired before the file dialog is shown.

OnDragOver

See also

Event: OnDragOver(*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as SPYMouseEvent, *objRange* as AuthenticRange, *objData* as AuthenticDataTransfer) as Boolean

Scripting environment - VBScript:
```
Function On_AuthenticDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
	'On_AuthenticDragOver = False  'to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
{
	// return false; /* to disable operation */
}
```

IDE Plugin:
```
IXMLSpyPlugIn.OnEvent(10, ...)		// nEventId = 10
```

Description
This event gets triggered whenever an object from within or outside of Authentic View gets dragged with the mouse over the application window. All event related information gets passed as parameters.

The first three parameters specify the mouse position, the mouse button status and the status of the virtual keys at the time when the event occurred. The parameter *objRange* passes a range object that selects the XML element below the mouse position. The value of this parameter might be *NULL*. Be sure to check before you access the range object. The parameter *objData* allows to access information about the object being dragged.

Return *False* to cancel the drag operation. Return *True* (or nothing) to continue normal operation.

Examples

```
'--------------------------------------------------------------------------------
'VB code snippet - connecting to object level events
'--------------------------------------------------------------------------------
'access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("","XMLSpy.Application")

'this is the event callback routine connected to the OnDragOver
'event of object objView
Private Function objView_OnDragOver(ByVal inXPos As Long, ByVal inYPos As Long,
                        ByVal ieMouseEvent As SPYMouseEvent,
                        ByVal ipRange As IAuthenticRange,
                        ByVal ipData As IAuthenticDataTransfer) As Boolean

   If (((ieMouseEvent And spyShiftKeyDownMask) <> 0) And
         (Not ipRange Is Nothing)) Then
      MsgBox ("Floating over element " & ipRange.FirstXMLData.ParentName);
   End If

   Return True;
End Function

'use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

'continue here with something useful...
'and serve the windows message loop
```

OnKeyboardEvent

See also

Event: OnKeyboardEvent(*eKeyEvent* as SPYKeyEvent, *nKeyCode* as Long, *nVirtualKeyStatus* as Long) as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticKeyboardEvent(eKeyEvent, nKeyCode, nVirtualKeyStatus)
      'On_AuthenticKeyboardEvent = True  'to cancel bubbling of event
End Function
```

Scripting environment - JScript:

```
function On_AuthenticKeyboardEvent(eKeyEvent, nKeyCode, nVirtualKeyStatus)
{
      // return true; /* to cancel bubbling of event */
}
```

IDE Plugin:

```
XMLSpyPlugInOnEvent(30, ...)          // nEventId = 30
```

Description

This event gets triggered for *WM_KEYDOWN*, *WM_KEYUP* and *WM_CHAR* Windows messages.

The actual message type is available in the *eKeyEvent* parameter. The status of virtual keys is combined in the parameter *nVirtualKeyStatus*. Use the bit-masks defined in the enumeration datatype SPYVirtualKeyMask, to test for the different keys or their combinations.

OnLoad

Event: OnLoad()

Description: OnLoad can be used e.g. to restrict some AuthenticView functionality, as shown in the example below:

```
function On_AuthenticLoad( )
{
    // We are disabling all entry helpers in order to prevent user from manipulating XML tree
    AuthenticView.DisableElementEntryHelper();
    AuthenticView.DisableAttributeEntryHelper();

    // We are also disabling the markup buttons for the same purpose
    AuthenticView.SetToolbarButtonState('AuthenticMarkupSmall', authenticToolbarButtonDisabled);
    AuthenticView.SetToolbarButtonState('AuthenticMarkupLarge', authenticToolbarButtonDisabled);
    AuthenticView.SetToolbarButtonState('AuthenticMarkupMixed', authenticToolbarButtonDisabled);
}
```

In the example the status of the Markup Small, Markup Large, Markup Mixed toolbar buttons are manipulated with the help of button identifiers. See complete list.

OnMouseEvent

See also

Event: OnMouseEvent(*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as SPYMouseEvent, *objRange* as AuthenticRange) as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
        'On_AuthenticMouseEvent = True  'to cancelbubbling ofevent
End Function
```

Scripting environment - JScript:

```
function On_AuthenticMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
{
        // return true; /* to cancelbubbling ofevent */
}
```

IDE Plugin:

```
XMLSpyPlugInOnEvent(31, ...)          // nEventId = 31
```

Description

This event gets triggered for every mouse movement and mouse button Windows message.

The actual message type and the mouse buttons status, is available in the *eMouseEvent*

parameter. Use the bit-masks defined in the enumeration datatype SPYMouseEvent to test for the different messages, button status, and their combinations.

The parameter *objRange* identifies the part of the document found at the current mouse cursor position. The range objects always selects a complete tag of the document. (This might change in future versions, when a more precise positioning mechanism becomes available). If no selectable part of the document is found at the current position, the range object is *null*.

OnSelectionChanged

See also

Event: OnSelectionChanged (*objNewSelection* as AuthenticRange)

Scripting environment - VBScript:
```
Function On_AuthenticSelectionChanged (objNewSelection)
End Function
```

Scripting environment - JScript:
```
function On_AuthenticSelectionChanged (objNewSelection)
{
}
```

IDE Plugin:
```
IXMLSpyPlugIn.OnEvent(23, ...)          // nEventId = 23
```

Description
This event gets triggered whenever the selection in the user interface changes.

Examples
```
'--------------------------------------------------------------------------
'VB code snippet - connecting to object level events
'--------------------------------------------------------------------------
'access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("","XMLSpy.Application")

'this is the event callback routine connected to the OnSelectionChanged
'event of object objView
Private Sub objView_OnSelectionChanged (ByVal ipNewRange As XMLSpyLib.IAuthenticRange)
   MsgBox ("new selection:" & ipNewRange.Text)
End Sub

'use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

'continue here with something useful...
'and serve the windows message loop
```

OnToolbarButtonClicked

Event: OnToolbarButtonClicked (Button identifier)

Description: OnToolbarButtonClicked is fired when a toolbar button was clicked by user. The parameter button identifier helps to determine which button was clicked. The list of predefined button identifiers is below:

- AuthenticPrint
- AuthenticPrintPreview

- AuthenticUndo
- AuthenticRedo
- AuthenticCut
- AuthenticCopy
- AuthenticPaste
- AuthenticClear
- AuthenticMarkupHide
- AuthenticMarkupLarge
- AuthenticMarkupMixed
- AuthenticMarkupSmall
- AuthenticValidate
- AuthenticChangeWorkingDBXMLCell
- AuthenticSave
- AuthenticSaveAs
- AuthenticReload
- AuthenticTableInsertRow
- AuthenticTableAppendRow
- AuthenticTableDeleteRow
- AuthenticTableInsertCol
- AuthenticTableAppendCol
- AuthenticTableDeleteCol
- AuthenticTableJoinCellRight
- AuthenticTableJoinCellLeft
- AuthenticTableJoinCellAbove
- AuthenticTableJoinCellBelow
- AuthenticTableSplitCellHorizontally
- AuthenticTableSplitCellVertically
- AuthenticTableAlignCellContentTop
- AuthenticTableCenterCellVertically
- AuthenticTableAlignCellContentBottom
- AuthenticTableAlignCellContentLeft
- AuthenticTableCenterCellContent
- AuthenticTableAlignCellContentRight
- AuthenticTableJustifyCellContent
- AuthenticTableInsertTable
- AuthenticTableDeleteTable
- AuthenticTableProperties
- AuthenticAppendRow
- AuthenticInsertRow
- AuthenticDuplicateRow
- AuthenticMoveRowUp
- AuthenticMoveRowDown
- AuthenticDeleteRow
- AuthenticDefineEntities
- AuthenticXMLSignature

For custom buttons the user might add his own identifiers. Please, note that the user must take care, as the identifiers are not checked for uniqueness. The same identifiers can be used to identify buttons in the `Set/GetToolbarState()` COM API calls. By adding code for different buttons, the user is in the position to completely redefine the AuthenticView toolbar behavior, adding own methods for table manipulation, etc.

OnToolbarButtonExecuted

Event: `OnToolbarButtonExecuted` (Button identifier)

Description: `OnToolbarButtonClicked` is fired when a toolbar button was clicked by user. The parameter button identifier helps to determine which button was clicked. See the list of predefined button identifiers.

`OnToolbarButtonExecuted` is fired after the toolbar action was executed. It is useful e.g. to add update code, as shown in the example below:

```
//event fired when a toolbar button action was executed
function On_AuthenticToolbarButtonExecuted(varBtnIdentifier)
{
    // After whatever command user has executed - make sure to update toolbar button states
    UpdateOwnToolbarButtonStates();
}
```

In this case `UpdateOwnToolbarButtonStates` is a user function defined in the Global Declarations.

OnUserAddedXMLNode

Event: `OnUserAddedXMLNode` (XML node)

Description: `OnUserAddedXMLNode` will be fired when the user adds an XML node as a primary action. This happens in the situations, where the user clicks on

- auto-add hyperlinks (see example OnUserAddedXMLNode.sps)
- the Insert…, Insert After…, Insert Before… context menu items
- Append row, Insert row toolbar buttons
- Insert After…, Insert Before… actions in element entry helper (outside StyleVision)

The event doesn't get fired on Duplicate row, or when the node was added externally (e.g. via COM API), or on Apply (e.g. Text State Icons), or when in XML table operations or in DB operations.

The event parameter is the XML node object, which was added giving the user an opportunity to manipulate the XML node added. An elaborate example for an event handler can be found in the `OnUserAddedXMLNode.sps` file.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Accesses the XMLSpy application object.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

AsXMLString

See also

Property: AsXMLString as String

Description
Returns or sets the document content as an XML string. Setting the content to a new value does not change the schema file or sps file in use. If the new XMLString does not match the actual schema file error 2011 gets returned.

Errors
2000 The authentic view object is no longer valid.
2011 AsXMLString was set to a value which is no valid XML for the current schema file.

ContextMenu

Property: ContextMenu() as ContextMenu

Description
The property ContextMenu gives access to customize the context menu. The best place to do it is in the event handler OnContextMenuActivated.

Errors
2000 Invalid object.
2005 Invalid parameter.

CreateXMLNode

Method: CreateXMLNode (*nKind* as SPYXMLDataKind) as XMLData

Return Value
The method returns the new XMLData object.

Description
To create a new XMLData object use the CreateXMLNode() method.

Errors
2000 Invalid object.
2012 Cannot create XML node.

DisableAttributeEntryHelper

Method: DisableAttributeEntryHelper()

Description
DisableAttributeEntryHelper() disables the attribute entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

DisableElementEntryHelper

Method: DisableElementEntryHelper()

Description
DisableElementEntryHelper() disables the element entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

DisableEntityEntryHelper

Method: DisableEntityEntryHelper()

Description
DisableEntityEntryHelper() disables the entity entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

DocumentBegin

See also

Property: DocumentBegin as AuthenticRange (read-only)

Description
Retrieve a range object that points to the beginning of the document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

DocumentEnd

See also

Property: DocumentEnd as AuthenticRange (read-only)

Description
Retrieve a range object that points to the end of the document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

DoNotPerformStandardAction

Method: DoNotPerformStandardAction ()

Description
DoNotPerformStandardAction() serves as cancel bubble for macros, and stops further execution after macro has finished.

Errors
2000 Invalid object.

EvaluateXPath

Method: EvaluateXPath (XMLData as XMLData, strExpression as string) strValue as string

Return Value
The method returns a string

Description
EvaluateXPath() executes an XPath expressions with the given XML context node. The result is returned as string, in the case of a sequence it is a space-separated string.

Errors
2000 Invalid object.
2005 Invalid parameter.
2008 Internal error.
2013 XPath error.

Event

See also

Property: Event as AuthenticEvent (read-only)

Description
This property gives access to parameters of the last event in the same way as OldAuthenticView.event does. Since all events for the scripting environment and external clients are now available with parameters this Event property should only be used from within IDE-Plugins.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

EventContext

Property: EventContext() as EventContext

Description
EventContext property gives access to the running macros context. See the EventContext interface description for more details.

Errors
2000 Invalid object.

GetToolbarButtonState

Method: GetToolbarButtonState (ButtonIdentifier as string) as AuthenticToolbarButtonState

Return Value
The method returns AuthenticToolbarButtonState

Description
`Get/SetToolbarButtonState` queries the status of a toolbar button, and lets the user disable or enable the button, identified via its button identifier (see list above). One usage is to disable toolbar buttons permanently. Another usage is to put `SetToolbarButtonState` in the `OnSelectionChanged` event handler, as toolbar buttons are updated regularly when the selection changes in the document.

Toolbar button states are given by the listed enumerations.

The default state means that the enable/disable of the button is governed by AuthenticView. When the user sets the button state to enable or disable, the button remains in that state as long as the user does not change it.

Errors
2000 Invalid object.
2005 Invalid parameter.
2008 Internal error.
2014 Invalid button identifier.

Goto

See also

Method: Goto (*eKind* as SPYAuthenticElementKind, *nCount* as Long, *eFrom* as *SPYAuthenticDocumentPosition*) as *AuthenticRange*

Description
Retrieve a range object that points to the beginning of the *nCount* element of type *eKind*. The start position is defined by the parameter *eFrom*. Use positive values for *nCount* to navigate to the document end. Use negative values to navigate towards the beginning of the document.

Errors
2000 The authentic view object is no longer valid.
2003 Target lies after end of document.
2004 Target lies before beginning of document.
2005 Invalid element kind specified.
The document position to start from is not one of *spyAuthenticDocumentBegin* or *spyAuthenticDocumentEnd*.
Invalid address for the return parameter was specified.

Examples

```
'----------------------------------------
'Scripting environment - VBScript
'----------------------------------------
Dim objAuthenticView
'we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

On Error Resume Next
Dim objRange
'goto beginning of first table in document
Set objRange = objAuthenticView.Goto (spyAuthenticTable,1,spyAuthenticDocumentBegin)
If (Err.number = 0) Then
      objRange.Select()
Else
      MsgBox "No table found in document"
End If
```

IsRedoEnabled

See also

Property: IsRedoEnabled as Boolean (read-only)

Description
True if redo steps are available and Redo is possible.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

IsUndoEnabled

See also

Property: IsUndoEnabled as Boolean (read-only)

Description
True if undo steps are available and Undo is possible.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

MarkupVisibility

See also

Property: MarkupVisibility as SPYAuthenticMarkupVisibility

Description
Set or get current visibility of markup.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid enumeration value was specified.
Invalid address for the return parameter was specified.

Parent

See also

Property: Parent as Document (read-only)

Description
Access the document shown in this view.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Print

See also

Method: Print(*bWithPreview* as Boolean, *bPromptUser* as Boolean)

Description
Print the document shown in this view. If *bWithPreview* is set to *True*, the print preview dialog pops up. If *bPromptUser* is set to *True*, the print dialog pops up. If both parameters are set to *False*, the document gets printed without further user interaction.

Errors
2000 The authentic view object is no longer valid.

Redo

See also

Method: Redo() as Boolean

Description
Redo the modification undone by the last undo command.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Selection

See also

Property: Selection as AuthenticRange

Description
Set or get current text selection in user interface.

Errors
2000 The authentic view object is no longer valid.
2002 No cursor selection is active.
2005 Invalid address for the return parameter was specified.

Examples

```
'----------------------------------------
'Scripting environment - VBScript
'----------------------------------------
Dim objAuthenticView
'we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

'if we are the end of the document, re-start at the beginning
If (objAuthenticView.Selection.IsEqual(objAuthenticView.DocumentEnd)) Then
      objAuthenticView.Selection = objAuthenticView.DocumentBegin
Else
      'objAuthenticView.Selection = objAuthenticView.Selection.GotoNextCursorPosition()
      'or shorter:
       objAuthenticView.Selection.GotoNextCursorPosition().Select
End If
```

SetToolbarButtonState

Method: SetToolbarButtonState (ButtonIdentifier as string, AuthenticToolbarButtonState state)

Description
Get/SetToolbarButtonState queries the status of a toolbar button, and lets the user disable or enable the button, identified via its button identifier (see list above). One usage is to disable toolbar buttons permanently. Another usage is to put SetToolbarButtonState in the OnSelectionChanged event handler, as toolbar buttons are updated regularly when the selection changes in the document.

Toolbar button states are given by the listed enumerations.

The default state means that the enable/disable of the button is governed by AuthenticView. When the user sets the button state to enable or disable, the button remains in that state as long as the user does not change it.

Errors
2000 Invalid object.
2008 Internal error.
2014 Invalid button identifier.

Undo

See also

Method: Undo () as Boolean

Description
Undo the last modification of the document from within this view.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

UpdateXMLInstanceEntities

See also

Method: UpdateXMLInstanceEntities()

Description
Updates the internal representation of the declared entities, and refills the entry helper. In addition, the validator is reloaded, allowing the XML file to validate correctly. Please note that this may also cause schema files to be reloaded.

Errors
The method never returns an error.

Example

```
// ----------------------------------------
// Scripting environment - JavaScript
// ----------------------------------------
if(Application.ActiveDocument && (Application.ActiveDocument.CurrentViewMode == 4))
{
	var objDocType;
```

```
        objDocType = Application.ActiveDocument.DocEditView.XMLRoot.GetFirstChild(10);

        if(objDocType)
        {
                var objEntity = Application.ActiveDocument.CreateChild(14);
                objEntity.Name = "child";
                objEntity.TextValue = "SYSTEM \"childxml\"";
                objDocType.AppendChild(objEntity);

                Application.ActiveDocument.AuthenticView.UpdateXMLInstanceEntities();
        }
}
```

WholeDocument

See also

Property: WholeDocument as AuthenticRange (read-only)

Description
Retrieve a range object that selects the whole document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

XMLDataRoot

See also

Property: XMLDataRoot as XMLData (read-only)

Description
Returns or sets the top-level XMLData element of the current document. This element typically describes the document structure and would be of kind spyXMLDataXMLDocStruct, spyXMLDataXMLEntityDocStruct or spyXMLDataDTDDocStruct..

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

3.2.7 CodeGeneratorDlg

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Properties and Methods

Standard automation properties
Application
Parent

Programming language selection properties
ProgrammingLanguage
TemplateFileName

Settings for C++ code
CPPSettings_DOMType

CPPSettings_LibraryType
CPPSettings_UseMFC
CPPSettings_GenerateVC6ProjectFile
CPPSettings_GenerateVSProjectFile

Settings for C# code
CSharpSettings_ProjectType

Dialog handling for above code generation properties
PropertySheetDialogAction

Output path selection properties
OutputPath
OutputPathDialogAction

Presentation of result
OutputResultDialogAction

Description
Use this object to configure the generation of program code for schema files. The method GenerateProgramCode expects a CodeGeneratorDlg as parameter to configure code generation as well as the associated user interactions.

Application

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

CompatibilityMode (obsolete)

Property: CompatibilityMode as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Set to `true` to generate code compatible to XMLSpy 2005R3. Set to `false` to use newly added code-generation features.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_DOMType

Property: CPPSettings_DOMType as SPYDOMType
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_GenerateVC6ProjectFile

Property: CPPSettings_GenerateVC6ProjectFile as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_GenerateGCCMakefile

Property: CPPSettings_GenerateGCCMakefile as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Creates makefiles to compile the generated code under Linux with GCC.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_GenerateVSProjectFile

Property: CSharpSettings_GenerateVSProjectFile as SPYProjectType
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code. Only `spyVisualStudio2005Project (=4)` and `spyVisualStudio2008Project (=5)` and `spyVisualStudio2010Project (=6)` are valid project types.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_LibraryType

Property: CPPSettings_LibraryType as SPYLibType
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_UseMFC

Property: CPPSettings_UseMFC as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CSharpSettings_ProjectType

Property: CSharpSettings_ProjectType as SPYProjectType
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines the only setting to configure generation of C# code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

OutputPath

Property: OutputPath as String
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the base directory for all generated code.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

OutputPathDialogAction

Property: OutputPathDialogAction as SPYDialogAction
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog for selecting the code generation output path gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current value of the OutputPath property as default. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

OutputResultDialogAction

Property: OutputResultDialogAction as SPYDialogAction
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog that asks to show the result of the code generation process gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

Parent

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

ProgrammingLanguage

Property: ProgrammingLanguage as ProgrammingLanguage
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the output language for the code to be generated.

CAUTION: Setting this property to one of C++, C# or Java, changes the property TemplateFileName to the appropriate template file delivered with XMLSpy as well. If you want to generate C++, C# or Java code based on your own templates, set first the programming language and then select your template file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

PropertySheetDialogAction

Property: PropertySheetDialogAction as SPYDialogAction
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog that configures the code generation process gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current values as defaults. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

TemplateFileName

Property: TemplateFileName as String
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the code generation template file. XMLSpy comes with template files for C++, C# or Java in the SPL folder of your installation directory.

Setting this property to one of the code generation template files of your XMLSpy installation automatically sets the ProgrammingLanguage property to its appropriate value.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

3.2.8 DatabaseConnection

See also

Properties for import and export
File or
ADOConnection or
ODBCConnection

Properties for import only
DatabaseKind

SQLSelect
AsAttributes
ExcludeKeys
IncludeEmptyElements
NumberDateTimeFormat
NullReplacement
CommentIncluded

Properties for export only
CreateMissingTables
CreateNew
TextFieldLen
DatabaseSchema

Properties for XML Schema from DB Structure generation
PrimaryKeys
ForeignKeys
UniqueKeys
SchemaExtensionType
SchemaFormat
ImportColumnsType

Description
DatabaseConnection specifies the parameters for the database connection.

Please note that the properties of the DatabaseConnection interface are referring to the settings of the import and export dialogs of XMLSpy.

ADOConnection

See also

Property: ADOConnection as String

Description
The property ADOConnection contains a connection string. Either use this property or ODBCConnection or File to refer to a database.

Errors
No error codes are returned.

Example

```
Dim objSpyConn As DatabaseConnection
Set objSpyConn = objSpy.GetDatabaseSettings

Dim objADO As DataLinks
Set objADO = CreateObject("DataLinks")

If Not (objADO Is Nothing) Then
 Dim objConn As Connection
 Set objConn = objADO.PromptNew
 objSpyConn.ADOConnection = objConn.ConnectionString
End If
```

AsAttributes

See also

Property: AsAttributes as Boolean

Description
Set AsAttributes to true if you want to initialize all import fields to be imported as attributes. Default is false and will initialize all fields to be imported as elements. This property is used only in calls to Application.GetDatabaseImportElementList.

Errors
No error codes are returned.

CommentIncluded

See also

Property: CommentIncluded as Boolean

Description
This property tells whether additional comments are added to the generated XML. Default is true. This property is used only when importing from databases.

Errors
No error codes are returned.

CreateMissingTables

See also

Property: CreateMissingTables as Boolean

Description
If CreateMissingTables is true, tables which are not already defined in the export database will be created during export. Default is true. This property is used only when exporting to databases.

Errors
No error codes are returned.

CreateNew

See also

Property: CreateNew as Boolean

Description
Set CreateNew true if you want to create a new database on export. Any existing database will be overwritten. See also DatabaseConnectionFile. Default is false. This property is used only when exporting to databases.

Errors
No error codes are returned.

DatabaseKind

See also

Property: DatabaseKind as SPYDatabaseKind

Description
Select the kind of database that gets access. The default value is spyDB_Unspecified(7) and is sufficient in most cases. This property is used only when importing from databases.

Errors
No error codes are returned.

DatabaseSchema

See also

Property: DatabaseSchema as String

Description
This property specifies the Schema used for export in Schema aware databases. Default is "". This property is used only when exporting to databases.

Errors
No error codes are returned.

ExcludeKeys

See also

Property: `ExcludeKeys` as `Boolean`

Description
Set `ExcludeKeys` to true if you want to exclude all key columns from the import data. Default is false. This property is used only when importing from databases.

Errors
No error codes are returned.

File

See also

Property: `File` as `String`

Description
The property `File` sets the path for the database during export or import. This property can only be used in conjunction with a Microsoft Access database. Either use this property or `ODBCConnection` or `ADOConnection` to refer to the database.

See also Import and Export.

Errors
No error codes are returned.

ForeignKeys

See also

Property: `ForeignKeys` as `Boolean`

Description
Specifies whether the Foreign Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

ImportColumnsType

See also

Property: ImportColumnsType as SPYImportColumnsType

Description
Defines if column information from the DB is saved as element or attribute in the XML Schema. Default is as element. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

IncludeEmptyElements

See also

Property: IncludeEmptyElements as Boolean

Description
Set IncludeEmptyElements to false if you want to exclude all empty elements. Default is true. This property is used only when importing from databases.

Errors
No error codes are returned.

NullReplacement

See also

Property: NullReplacement as String

Description
This property contains the text value that is used during import for empty elements (null values). Default is "". This property is used only when importing from databases.

Errors
No error codes are returned.

NumberDateTimeFormat

See also

Property: NumberDateTimeFormat as SPYNumberDateTimeFormat

Description
The property NumberDateTimeFormat sets the format of numbers and date- and time-values. Default is spySystemLocale. This property is used only when importing from databases.

Errors
No error codes are returned.

ODBCConnection

See also

Property: ODBCConnection as String

Description
The property ODBCConnection contains a ODBC connection string. Either use this property or ADOConnection or File to refer to a database.

Errors
No error codes are returned.

PrimaryKeys

See also

Property: PrimaryKeys as Boolean

Description
Specifies whether the Primary Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

SchemaExtensionType

See also

Property: SchemaExtensionType as SPYSchemaExtensionType

Description
Defines the Schema extension type used during the Schema generation. This property is used only when creating a XML Schema from a DB structure.

See also Create XML Schema from DB Structure.

Errors
No error codes are returned.

SchemaFormat

See also

Property: SchemaFormat as SPYSchemaFormat

Description
Defines the Schema format used during the Schema generation. This property is used only when creating a XML Schema from a DB structure.

See also Create XML Schema from DB Structure.

Errors
No error codes are returned.

SQLSelect

See also

Property: SQLSelect as String

Description
The SQL query for the import is stored in the property SQLSelect. This property is used only when importing from databases. See also Import and Export.

Errors
No error codes are returned.

TextFieldLen

See also

Property: TextFieldLen as long

Description
The property TextFieldLen sets the length for created text fields during the export. Default is 255. This property is used only when exporting to databases.

Errors
No error codes are returned.

UniqueKeys

See also

Property: UniqueKeys as Boolean

Description
Specifies whether the Unique Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

3.2.9 Dialogs

See also

Properties and Methods

Standard automation properties
Application
Parent

Various dialog objects
CodeGeneratorDlg
FileSelectionDlg
SchemaDocumentationDlg
GenerateSampleXMLDlg
DTDSchemaGeneratorDlg
FindInFilesDlg
WSDLDocumentationDlg
WSDL20DocumentationDlg

XBRLDocumentationDlg

Description
The Dialogs object provides access to different built-in dialogs of XMLSpy. These dialog objects allow to initialize the fields of user dialogs before they get presented to the user or allow to simulate complete user input by your program.

Application

See also

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
2300 The object is no longer valid.
2301 Invalid address for the return parameter was specified.

CodeGeneratorDlg

See also
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: CodeGeneratorDlg as CodeGeneratorDlg (read-only)

Description
Get a new instance of a code generation dialog object. You will need this object to pass the necessary parameters to the code generation methods. Initial values are taken from last usage of the code generation dialog.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

FileSelectionDlg

See also

Property: FileSelectionDlg as FileSelectionDlg (read-only)

Description
Get a new instance of a file selection dialog object.

File selection dialog objects are passed to you with the some events that signal opening or saving of documents and projects.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

Parent

See also

Property: Parent as Application (read-only)

Description
Access the XMLSpy application object.

Errors
2300 The object is no longer valid.
2301 Invalid address for the return parameter was specified.

SchemaDocumentationDlg

See also

Property: SchemaDocumentationDlg as SchemaDocumentationDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of schema documentation. See Document.GenerateSchemaDocumentation for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

GenerateSampleXMLDlg

See also

Property: GenerateSampleXMLDlg as GenerateSampleXMLDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of a sample XML based on a W3C schema or DTD. See GenerateSampleXML for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

DTDSchemaGeneratorDlg

See also

Property: DTDSchemaGeneratorDlg as DTDSchemaGeneratorDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of a schema or DTD. See Document.GenerateDTDOrSchemaEx for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

FindInFilesDlg

See also

Property: FindInFilesDlg as FindInFilesDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes the search (or replacement) of strings in files. See Application.FindInFiles for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

WSDLDocumentationDlg

See also

Property: WSDLDocumentationDlg as WSDLDocumentationDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of WSDL documentation. See Document.GenerateWSDLDocumentation for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

WSDL20DocumentationDlg

See also

Property: WSDL20DocumentationDlg as WSDL20DocumentationDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of WSDL 2.0 documentation. See Document.GenerateWSD20LDocumentation for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

XBRLDocumentationDlg

See also

Property: XBRL20DocumentationDlg as XBRL20DocumentationDlg (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of WSDL 2.0 documentation. See Document.GenerateXBRLDocumentation for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

3.2.10 Document

See also

Properties and Methods

Standard automation properties
Application
Parent

Various document properties and methods
SetActiveDocument
Encoding
SetEncoding (obsolete)
Suggestions

XML validation
IsValid
SetExternalIsValid

Document conversion and transformation
AssignDTD
AssignSchema
AssignXSL
AssignXSLFO
ConvertDTDOrSchema
ConvertDTDOrSchemaEx
GenerateDTDOrSchema
GenerateDTDOrSchemaEx
CreateSchemaDiagram
ExecuteXQuery
TransformXSL
TransformXSLEx
TransformXSLFO
GenerateProgramCode (Enterprise Edition only)
GenerateSchemaDocumentation
GenerateSampleXML
GenerateWSDL20Documentation
GenerateWSDLDocumentation
GenerateXBRLDocumentation
ConvertToWSDL20

Document export
GetExportElementList
ExportToText
ExportToDatabase
CreateDBStructureFromXMLSchema
GetDBStructureList

File saving and naming
FullName
Name
Path
GetPathName (obsolete)
SetPathName (obsolete)
Title
IsModified
Saved
SaveAs
Save
SaveInString

SaveToURL
Close

View access
CurrentViewMode
SwitchViewMode
AuthenticView
GridView
DocEditView (obsolete)

Access to XMLData
RootElement
DataRoot
CreateChild
UpdateViews
StartChanges
EndChanges
UpdateXMLData

Description
Document objects represent XML documents opened in XMLSpy.

Use one of the following properties to access documents that are already open XMLSpy:
Application.ActiveDocument
Application.Documents

Use one of the following methods to open a new document in XMLSpy:
Documents.OpenFile
Documents.OpenURL
Documents.OpenURLDialog
Documents.NewFile
Documents.NewFileFromText
SpyProjectItem.Open
Application.ImportFromDatabase
Application.ImportFromSchema
Application.ImportFromText
Application.ImportFromWord
Document.ConvertDTDOrSchema
Document.GenerateDTDOrSchema

Events

OnBeforeSaveDocument

See also

Event: OnBeforeSaveDocument(*objDocument* as Document, *objDialog* as FileSelectionDlg)

XMLSpy scripting environment - VBScript:

```
Function On_BeforeSaveDocument(objDocument, objDialog)
End Function

'old handler - now obsolete
'return string to save to new file name
'return empty string to cancel save operation
'return nothing to save to original name
Function On_SaveDocument(objDocument, strFilePath)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeSaveDocument(objDocument,objDialog)
{
}

// old handler - now obsolete
// return string to save to new file name
// return empty string to cancel save operation
// return nothing to save to original name
function On_SaveDocument(objDocument,strFilePath)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent(27, ...)          // nEventId = 27
```

Description
This event gets fired on any attempt to save a document. The file selection dialog object is initialized with the name chosen for the document file. You can modify this selection. To continue saving the document leave the FileSelectionDlg.DialogAction property of *io_objDialog* at its default value spyDialogOK. To abort saving of the document set this property to spyDialogCancel.

OnBeforeCloseDocument

See also

Event: OnBeforeCloseDocument(*objDocument* as Document)as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_BeforeCloseDocument(objDocument)
      'On_BeforeCloseDocument = False  'to prohibit closing of document
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeCloseDocument(objDocument)
{
      // return false; /* to prohibit closing of document */
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent(28, ...)          // nEventId = 28
```

Description
This event gets fired on any attempt to close a document. To prevent the document from being closed return false.

OnBeforeValidate

See also

Event: OnBeforeValidate (*objDocument* as Document,*bOnLoading* as Boolean,*bOnCommand* as Boolean) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_BeforeValidate (objDocument,bOnLoading,bOnCommand)
      On_BeforeValidate = bCancelDefaultValidation      'set by the script if necessary
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeValidate(objDocument,bOnLoading,bOnCommand)
{
        return bCancelDefaultValidation  //set by the script if necessary
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent(32, ...)          // nEventId = 32
```

Description

This event gets fired before the document is validated. It is possible to suppress the default validation by returning false from the event handler. In this case the script should also set the validation result using the SetExternalIsValid method.

bOnLoading is true if the event is raised on the initial validation on loading the document.

bOnCommand is true whenever the user selected the Validate command from the Toolbar or menu.

Available with TypeLibrary version 1.5

OnCloseDocument

See also

Event: OnCloseDocument(*objDocument* as Document)

XMLSpy scripting environment - VBScript:

```
Function On_Close Document(objDocument)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_Close Document(objDocument)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent(8, ...) // nEventId = 8
```

Description

This event gets fired as a result of closing a document. Do not modify the document from within this event.

OnViewActivation

See also

Event: OnViewActivation(*objDocument* as Document, *eViewMode* as SPYViewModes, *bActivated* as Boolean)

XMLSpy scripting environment - VBScript:

```
Function On_ViewActivation(objDocument,eViewMode,bActivated)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_ViewActivation(objDocument,eViewMode,bActivated)
{
```

```
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent(29, ...)          // nEventId = 29
```

Description
This event gets fired whenever a view of a document becomes visible (i.e. becomes the active view) or invisible (i.e. another view becomes the active view or the document gets closed). However, the first view activation event after a document gets opened cannot be received, since there is no document object to get the event from. Use the Application.OnDocumentOpened event instead.

Application

See also

Property: Application as Application (read-only)

Description
Accesses the XMLSpy application object.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

AssignDTD

See also

Method: AssignDTD (*strDTDFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the DTD file "strDTDFile" into the document. Note that no error occurs if the file does not exist, or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign a DTD to the document.

AssignSchema

See also

Method: AssignSchema (*strSchemaFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the schema file "strSchemaFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign a schema file to the document.

AssignXSL

See also

Method: AssignXSL (*strXSLFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the XSL file "strXSLFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign an XSL file to the document.

AssignXSLFO

See also

Method: AssignXSLFO (*strXSLFOFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the XSLFO file "strXSLFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign an XSL file to the document.

AsXMLString

See also

Property: AsXMLString as String

Description
This property can be used to get or set the document content.

Errors
1400 The document object is no longer valid.
1404 Cannot create XMLData object.
1407 View mode cannot be switched.

AuthenticView

See also

Method: AuthenticView as AuthenticView (read-only)

Description
Returns an object that gives access to properties and methods specific to Authentic view. The object returned is only valid if the current document is opened in Authentic view mode. The lifetime of an object ends with the next view switch. Any attempt to access objects or any of its children afterwards will result in an error indicating that the object is invalid.

AuthenticView and DocEditView both provide automation access to the Authentic view mode of

XMLSpy. Functional overlap is intentional. A future version of Authentic View will include all functionality of DocEditView and its sub-objects, thereby making usage of DocEditView obsolete.

Errors

1400 The object is no longer valid.
1417 Document needs to be open in authentic view mode.

Examples

```
'---------------------------------------
'XMLSpy scripting environment - VBScript
'secure access to authentic view object
'---------------------------------------
Dim objDocument
Set objDocument = Application.ActiveDocument
If (Not objDocument Is Nothing) Then
	'we have an active document, now check for view mode
	If (objDocument.CurrentViewMode <> spyViewAuthentic) Then
		If (Not objDocument.SwitchViewMode (spyViewAuthentic)) Then
			MsgBox "Active document does not support authentic view mode"
		Else
			'now it is safe to access the authentic view object
			Dim objAuthenticView
			Set objAuthenticView = objDocument.AuthenticView
			'now use the authentic view object

		End If
	End If
Else
	MsgBox "No document is open"
End If
```

Close

See also

Method: Close (*bDiscardChanges* as Boolean)

Description

To close the document call this method. If bDiscardChanges is true and the document is modified, the document will be closed but not saved.

Errors

1400 The object is no longer valid.
1401 Document needs to be saved first.

ConvertDTDOrSchema

See also

Method: ConvertDTDOrSchema (*nFormat* as SPYDTDSchemaFormat, *nFrequentElements* as SPYFrequentElements)

Parameters

nFormat
Sets the schema output format to DTD or W3C.

nFrequentElements
Create complex elements as elements or complex types.

Description
ConvertDTDOrSchema takes an existing schema format and converts it into a different format. For a finer tuning of DTD/ XSD conversion, use ConvertDTDOrSchemaEx.

Errors
1400 The object is no longer valid.
1412 Error during conversion. In the case of DTD to DTD or XSD to XSD conversion, the following errors are returned: *DTD to DTD conversion is not supported. Please use function FlattenDTDOrSchema instead* and *Schema to schema conversion is not supported. Please use function FlattenDTDOrSchema instead*.

ConvertDTDOrSchemaEx

See also

Method: ConvertDTDOrSchemaEx (*nFormat* as SPYDTDSchemaFormat, *nFrequentElements* as SPYFrequentElements, sOutputPath as String, nOutputPathDialogAction as SPYDialogAction)

Parameters
nFormat
Sets the schema output format to DTD, or W3C.

nFrequentElements
Create complex elements as elements or complex types.

sOutputPath
The file path for the newly generated file.

nOutputPathDialogAction
Defines the dialog interaction for this call.

Description
ConvertDTDOrSchemaEx takes an existing schema format and converts it into a different format.

Errors
1400 The object is no longer valid.
1412 Error during conversion. In the case of DTD to DTD or XSD to XSD conversion, the following errors are returned: *DTD to DTD conversion is not supported. Please use function FlattenDTDOrSchema instead* and *Schema to schema conversion is not supported. Please use function FlattenDTDOrSchema instead*.

ConvertToWSDL20

Method: ConvertToWSDL20 (*sFilePath* as String, *bShowDialogs* as Boolean)

Parameters

sFilePath
This specifies the file name of the converted WSDL. In case the source WSDL includes files which also must be converted, then only the directory part of the given path is used and the file names are generated automatically.

bShowDialogs
Defines whether file/ folder selection dialogs are shown.

Description

Converts the WSDL 1.1 document to a WSDL 2.0 file. It will also convert any referenced WSDL files that are referenced from within this document. Note that this functionality is limited to WSDL View only. See DocumentCurrentViewMode. and SPYViewModes.

Errors

1400 The document object is no longer valid.
1407 Invalid parameters have been passed or an empty file name has been specified as output target.
1417 The document is not opened in WSDL view, maybe it is not an '.wsdl' file.
1421 Feature is not available in this edition.
1433 WSDL 1.1 to WSDL 2.0 conversion failed.

CreateChild

See also

Method: CreateChild (*nKind* as SPYXMLDataKind) as XMLData

Return Value

The method returns the new XMLData object.

Description

To create a new XMLData object use the CreateChild() method.

Errors

1400 The object is no longer valid.
1404 Cannot create XMLData object.
1407 Invalid address for the return parameter was specified.

CreateDBStructureFromXMLSchema

See also

Method: CreateDBStructureFromXMLSchema (*pDatabase* as DatabaseConnection, *pTables* as ElementList, *bDropTableWithExistingName* as Boolean) as String

Description

CreateDBStructureFromXMLSchema exports the given tables to the specified database. The function returns the SQL statements that were necessary to perform the changes.

See also GetDBStructureList.

Errors

1429 Database selection missing.
1430 Document export failed.

CreateSchemaDiagram

See also

Method: CreateSchemaDiagram (*nKind* as SPYSchemaDefKind, *strName* as String, *strFile* as String)

Return Value
None.

Description
The method creates a diagram of the schema type strName of kind nKind and saves the output file into strFile. Note that this functionality is limited to Schema View only. See DocumentCurrentViewMode. and SPYViewModes.

Errors
1400 The object is no longer valid.
1414 Failed to save diagram.
1415 Invalid schema definition type specified.

CurrentViewMode

See also

Method: CurrentViewMode as SPYViewModes

Description
The property holds the current view mode of the document. See also DocumentSwitchViewMode.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

DataRoot

See also

Property: DataRoot as XMLData (read-only)

Description
This property provides access to the document's first XMLData object of type *spyXMLDataElement*. This is typically the root element for all document content data. See XMLSpyDocument.RootElement to get the root element of the whole document including XML prolog data. If the CurrentViewMode is not *spyViewGrid* or *spyViewAuthentic* an UpdateXMLData may be necessary to get access to the latest XMLData.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

DocEditView

See also

Method: DocEditView as DocEditView

Description
Holds a reference to the current Authentic View object.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Document needs to be open in authentic view mode.

Encoding

See also

Property: Encoding as String

Description
This property provides access to the document's encoding value. However, this property can only be accessed when the document is opened in *spyViewGrid*, *spyViewText* or *spyViewAuthentic*. See CurrentViewMode on how to detect that a document's actual view mode.

This property makes the method SetEncoding obsolete.

Possible values are, for example:

8859-1,
8859-2,
ASCII, ISO-646,
850,
1252,
1255,
SHIFT-JIS, MS-KANJI,
BIG5, FIVE,
UTF-7,
UTF-8,
UTF-16

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.
1416 Operation not supported in current view mode.

EndChanges

See also

Method: EndChanges()

Description
Use the method EndChanges to display all changes since the call to Document.StartChanges.

Errors

1400 The object is no longer valid.

ExecuteXQuery

See also

Method: ExecuteXQuery (*strXMLFileName* as String)

Description
Execute the XQuery statements contained in the document of the document object. Either an XQuery execution or an XQuery Update is performed depending on the file extension of the document. Use the XML file specified in the argument as the XML target document that the XQuery document processes.

- If the document has an XQuery file extension as defined in the Options dialog of XMLSpy, then an XQuery execution is performed. By default: `.xq`, `.xql`, and `.xquery` are set as XQuery file extensions in XMLSpy.
- If the document has an XQuery Update file extension as defined in the Options dialog of XMLSpy, then an XQuery Update action is performed. By default: `.xqu` is set as an XQuery Update file extension in XMLSpy.

If your XQuery script does not use an XML source, set the parameter `strXMLFileName` to an empty string.

Errors

1400 The document object is no longer valid.
1423 XQuery transformation error.
1424 Not all files required for operation could be loaded. Most likely, the file specified in `strXMLFileName` does not exist or is not valid.

ExportToDatabase

See also

Method: ExportToDatabase (*pFromChild* as XMLData, *pExportSettings* as ExportSettings, *pDatabase* as DatabaseConnection)

Description
`ExportToDatabase` exports the XML document starting with the element `pFromChild`. The parameter `pExportSettings` defines the behaviour of the export (see Application.GetExportSettings). The parameter `pDatabase` specifies the destination of the export (see Application.GetDatabaseSettings). UpdateXMLData() might be indirectly needed as you have to pass the XMLData as parameter to this function.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1416 Error during export.
1429 Database selection missing.
1430 Document export failed.

Example

```
Dim objDoc As Document
```

```
Set objDoc = objSpyActiveDocument

'set the behaviour of the export with ExportSettings
Dim objExpSettings As ExportSettings
Set objExpSettings = objSpy.GetExportSettings

'set the destination with DatabaseConnection
Dim objDB As DatabaseConnection
Set objDB = objSpy.GetDatabaseSettings

objDB.CreateMissingTables = True
objDB.CreateNew = True
objDB.File = "C:\Export.mdb"

objDoc.ExportToDatabase objDoc.RootElement, objExpSettings, objDB
If ErrNumber <> 0 Then
 a = MsgBox("Error:" & (ErrNumber - vbObjectError) & Chr(13) &
   "Description:" & ErrDescription)
End If
```

ExportToText

See also

Method: ExportToText (*pFromChild* as XMLData, *pExportSettings* as ExportSettings, *pTextSettings* as TextImportExportSettings)

Description

ExportToText exports tabular information from the document starting at pFromChild into one or many text files. Columns of the resulting tables are generated in alphabetical order of the column header names. Use GetExportElementList to learn about the data that will be exported. The parameter pExportSettings defines the specifics for the export. Set the property ExportSettings.ElementList to the - possibly modified - list returned by GetExportElementList to avoid exporting all contained tables. The parameter pTextSettings defines the options specific to text export and import. You need to set the property TextImportExportSettings.DestinationFolder before you call ExportToText. UpdateXMLData() might be indirectly needed as you have to pass the XMLData as parameter to this function.

See also Import and export of data.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1416 Error during export.
1430 Document export failed.

Example

```
'-----------------------------------------------------------
'VBA client code fragment - export document to text files
'-----------------------------------------------------------
Dim objDoc As Document
Set objDoc = objSpyActiveDocument

Dim objExpSettings As ExportSettings
Set objExpSettings = objSpy.GetExportSettings
objExpSettings.ElementList = objDoc.GetExportElementList(
       objDoc.RootElement,
       objExpSettings)
```

```
Dim objTextExp As TextImportExportSettings
Set objTextExp = objSpyGetTextExportSettings
objTextExpHeaderRow = True
objTextExpDestinationFolder = "C:\Exports"

On Error Resume Next
objDocExportToText objDocRootElement, objExpSettings, objTextExp

If ErrNumber <> 0 Then
 a = MsgBox("Error:" & (ErrNumber - vbObjectError) & Chr(13) & "Description:"
 & ErrDescription)
End If
```

FlattenDTDOrSchema

Method: FlattenDTDOrSchema (sOutputPath as String, nOutputPathDialogAction as SPYDialogAction)

Parameters

sOutputPath
The file path for the newly generated file.

nOutputPathDialogAction
Defines the dialog interaction for this call.

Description

FlattenDTDOrSchema takes an existing DTD or schema, generates a flattened file, and saves the generated file at the specified location. In the case of DTDs, flattening removes parameter entities and produces a single DTD from a collection of modules; sections marked IGNORE are suppressed and unused parameter entities are deleted. When an XML Schema is flattened, (i) the components of all included schemas are added as global components of the active schema, and (ii) included schemas are deleted.

Errors

1400 The object is no longer valid.
1412 Error during conversion.

FullName

See also

Property: FullName as String

Description

This property can be used to get or set the full file name - including the path - to where the document gets saved. The validity of the name is not verified before the next save operation.

This property makes the methods GetPathName and SetPathName obsolete.

Errors

1400 The document object is no longer valid.
1402 Empty string has been specified as full file name.

GenerateDTDOrSchema

See also

Method: GenerateDTDOrSchema (*nFormat* as SPYDTDSchemaFormat, *nValuesList* as integer, *nDetection* as SPYTypeDetection, *nFrequentElements* as SPYFrequentElements)

Parameters

nFormat
Sets the schema output format to DTD, or W3C.

nValuesList
Generate not more than this amount of enumeration-facets per type. Set to -1 for unlimited.

nDetection
Specifies granularity of simple type detection.

nFrequentElements
Shall the types for all elements be defined as global? Use that value *spyGlobalComplexType* to define them on global scope. Otherwise, use the value *spyGlobalElements*.

Description
Use this method to automatically generate a DTD or schema for the current XML document. For a finer tuning of DTD / schema generation, use GenerateDTDOrSchemaEx.
Note that this functionality is not available in ZIP View only. See DocumentCurrentViewMode. and SPYViewModes.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

GenerateDTDOrSchemaEx

See also

Method: GenerateDTDOrSchemaEx (*objDlg* as DTDSchemaGeneratorDlg) as Document

Description
Use this method to automatically generate a DTD or schema for the current XML document. A DTDSchemaGeneratorDlg object is used to pass information to the schema/ DTD generator. The generation process can be configured to allow user interaction or run without further user input.
Note that this functionality is not available in ZIP View only. See DocumentCurrentViewMode. and SPYViewModes.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

GenerateProgramCode

Method: GenerateProgramCode (*objDlg* as CodeGeneratorDlg)
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Generate Java, C++ or C# class files from the XML Schema definitions in your document. A

CodeGeneratorDlg object is used to pass information to the code generator. The generation process can be configured to allow user interaction or run without further user input.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1421 Feature not available in this edition

GenerateSampleXML

Method: GenerateSampleXML (*objDlg* as GenerateSampleXMLDlg) as Document

Description
Generates a sample XML if the document is a schema or DTD. Use Dialogs.GenerateSampleXMLDlg to get an initialized set of options.

Available with TypeLibrary version 1.5

Errors
1400 The document object is no longer valid.

GenerateSchemaDocumentation

Method: GenerateSchemaDocumentation (*objDlg* as SchemaDocumentationDlg)

Description
Generate documentation for a schema definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.SchemaDocumentationDlg to get an initialized set of options. As a minimum, you will need to set the property SchemaDocumentationDlg.OutputFile before starting the generation process. Note that this functionality is limited to Schema View only. See Document.CurrentViewMode and SPYViewModes.

Errors
1400 The document object is no longer valid.
1407 Invalid parameters have been passed or an empty file name has been specified as output target.
1417 The document is not opened in schema view, maybe it is not an '.xsd' file.
1421 Feature is not available in this edition.
1422 Error during generation

GenerateWSDL20Documentation

Method: GenerateWSDL20Documentation (*objDlg* as WSDL20DocumentationDlg)

Description
Generate documentation for a WSDL definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.WSDL20DocumentationDlg to get an initialized set of options. As a minimum, you will need to set the property WSDL20DocumentationDlg.OutputFile before starting the generation process. Note that this functionality is limited to WSDL View only. See Document.CurrentViewMode and SPYViewModes.

Errors

1400 The document object is no longer valid.
1407 Invalid parameters have been passed or an empty file name has been specified as output target.
1417 The document is not opened in schema view, maybe it is not an '.xsd' file.
1421 Feature is not available in this edition.
1422 Error during generation

GenerateWSDLDocumentation

Method: GenerateWSDLDocumentation (*objDlg* as WSDLDocumentationDlg)

Description
Generate documentation for a WSDL definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use DialogsWSDLDocumentationDlg to get an initialized set of options. As a minimum, you will need to set the property WSDLDocumentationDlgOutputFile before starting the generation process. Note that this functionality is limited to WSDL View only. See DocumentCurrentViewMode and SPYViewModes.

Errors
1400 The document object is no longer valid.
1407 Invalid parameters have been passed or an empty file name has been specified as output target.
1417 The document is not opened in schema view, maybe it is not an '.xsd' file.
1421 Feature is not available in this edition.
1422 Error during generation

GenerateXBRLDocumentation

Method: GenerateXBRLDocumentation (*objDlg* as XBRLDocumentationDlg)

Description
Generate documentation for a WSDL definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use DialogsXBRLDocumentationDlg to get an initialized set of options. As a minimum, you will need to set the property XBRLDocumentationDlgOutputFile before starting the generation process. Note that this functionality is limited to XBRL View only. See DocumentCurrentViewMode and SPYViewModes.

Errors
1400 The document object is no longer valid.
1407 Invalid parameters have been passed or an empty file name has been specified as output target.
1417 The document is not opened in schema view, maybe it is not an '.xsd' file.
1421 Feature is not available in this edition.
1422 Error during generation

GetDBStructureList

See also

Method: GetDBStructureList (*pDatabase* as DatabaseConnection) as ElementList

Description
GetDBStructureList creates a collection of elements from the Schema document for which tables in the specified database are created. The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of the tables.

See also CreateDBStructureFromXMLSchema.

Errors
1400 The object is no longer valid.
1427 Failed creating parser for the specified XML.
1428 Export of element list failed.
1429 Database selection missing.

GetExportElementList

See also

Method: GetExportElementList(*pFromChild* as XMLData, *pExportSettings* as ExportSettings) as ElementList

Description
GetExportElementList creates a collection of elements to export from the document, depending on the settings in pExportSettings and starting from the element pFromChild. The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of the tables that can be exported from the document. The property ElementListItem.FieldCount contains the number of columns in the table. The property ElementListItem.RecordCount contains the number of records in the table. The property ElementListItem.ElementKind is unused. UpdateXMLData() might be indirectly needed as you have to pass the XMLData as parameter to this function.

See also Import and export of data.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1427 Failed creating parser for the specified XML.
1428 Export of element list failed.

GetPathName (obsolete)

Superseded by Document.FullName

```
// ----- javascript sample -----
// instead of:
// strPathName = Application.ActiveDocument.GetPathName();
// use now:
strPathName = Application.ActiveDocument.FullName;
```

See also

Method: GetPathName() as String

Description

The method GetPathName gets the path of the active document.

See also DocumentSetPathName (obsolete).

GridView

See also

Property: GridView as GridView

Description
This property provides access to the grid view functionality of the document.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Document needs to be open in enhanced grid view mode.

IsModified

See also

Property: IsModified as Boolean

Description
True if the document is modified.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

IsValid

See also

Method: IsValid (*strError* as Variant) as Boolean

Return Value
True if the document is valid, false if not. To call IsValid(), the application GUI must be visible. (If you wish to validate without the GUI being visible, please use Altova RaptorXML Server.)

Description
IsValid validates the document against its associated schema or DTD. strError gives you the same error message as when you validate the file within the GUI.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1408 Unable to validate file.

IsValidEx

Method: IsValidEx (*nXSDVersion* as SPYValidateXSDVersion, *nErrorLimit* as int, *nErrorFormat* as SPYValidateErrorFormat, out *strError* as Variant) as Boolean

Return Value
True if the document is valid, false if not.

Description
`IsValidEx` validates the document against its associated schema or DTD.

In parameters:
`nXSDVersion` which is an enumeration value of `SPYValidateXSDVersion` that selects the XSD version to validate against.
`nErrorLimit` which is an integer. Values must be `1` to `999`.
`nErrorFormat` which is an enumeration value of `SPYValidateErrorFormat` that selects the XSD version to validate against.

Out parameter:
`strError` is the error message, and is the same as that received when validating the file within the GUI.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1408 Unable to validate file.

IsWellFormed

See also

Method: `IsWellFormed` (*pData* as `XMLData`, *bWithChildren* as `Boolean`, *strError* as Variant, *nErrorPos* as Variant, *pBadXMLData* as Variant) as Boolean

Return Value
True if the document is well formed.

Description
`IsWellFormed` checks the document for well-formedness starting at the element `pData`.

If the document is not well formed, `strError` contains an error message, `nErrorPos` the position in the file and `pBadXMLData` holds a reference to the element which breaks the well-formedness. These out-parameters are defined as VARIANTs to support scripting languages like VBScript.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

Example

See IsValid.

Name

See also

Property: Name as String (read-only)

Description
Use this property to retrieve the name - not including the path - of the document file. To change the file name for a document use the property FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Parent

See also

Property: Parent as Documents (read-only)

Description
Access the parent of the document object.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Property: Parent as Application (read-only)

Path

See also

Property: Path as String (read-only)

Description
Use this property to retrieve the path - not including the file name - of the document file. To change the file name and path for a document use the property FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

RootElement

See also

Property: RootElement as XMLData (read-only)

Description
The property RootElement provides access to the root element of the XML structure of the document including the XML prolog data. To access the first element of a document's content navigate to the first child of kind *spyXMLDataElement* or use the Document.DataRoot property. If the CurrentViewMode is not *spyViewGrid* or *spyViewAuthentic* an UpdateXMLData may be necessary to get access to the latest XMLData.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Save

See also

Method: Save()

Description
The method writes any modifications of the document to the associated file. See also DocumentFullName.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1403 Error when saving file, probably the file name is invalid.

SaveAs

See also

Method: SaveAs (*strFileName* as String)

Description
Save the document to the file specified. If saving was successful, the FullName property gets set to the specified file name.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1403 Error when saving file, probably the file name is invalid.

Saved

See also

Property: Saved as Boolean (read-only)

Description
This property can be used to check if the document has been saved after the last modifications. It returns the negation of IsModified.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

SaveInString

See also

Method: SaveInString (*pData* as XMLData, *bMarked* as Boolean) as String

Parameters
pData
XMLData element to start. Set pData to DocumentRootElement if you want to copy the complete

file.

bMarked
If bMarked is true, only the elements selected in the grid view are copied.

Return Value
Returns a string with the XML data.

Description
SaveInString starts at the element pData and converts the XMLData objects to a string representation. UpdateXMLData() might be indirectly needed as you have to pass the XMLData as parameter to this function.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

SaveToURL

See also

Method: SaveToURL (*strURL* as String, *strUser* as String, *strPassword* as String)

Return Value

Description
SaveToURL() writes the document to the URL strURL. This method does not set the permanent file path of the document.

Errors
1400 The object is no longer valid.
1402 Invalid URL specified.
1403 Error while saving to URL.

SetActiveDocument

See also

Method: SetActiveDocument()

Description
The method sets the document as the active and brings it to the front.

Errors
1400 The object is no longer valid.

SetEncoding (obsolete)

Superseded by Document.Encoding

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetEncoding("UTF-16");
// use now:
Application.ActiveDocument.Encoding = "UTF-16";
```

See also

Method: SetEncoding (*strEncoding* as String)

Description
SetEncoding sets the encoding of the document like the menu item "File/Encoding..." in XMLSpy. Possible values for strEncoding are, for example:

8859-1,
8859-2,
ASCII, ISO-646,
850,
1252,
1255,
SHIFT-JIS, MS-KANJI,
BIG5, FIVE,
UTF-7,
UTF-8,
UTF-16

SetExternalIsValid

See also

Method: SetExternalIsValid (*bValid* as Boolean)

Parameters

bValid
Sets the result of an external validation process.

Description
The internal information set by this method is only queried on cancelling the default validation in any OnBeforeValidate handler.

Available with TypeLibrary version 1.5

Errors
1400 The object is no longer valid.

SetPathName (obsolete)

Superseded by Document.FullName

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetPathName("C:\\myXMLFiles\\test.xml");
// use now:
Application.ActiveDocument.FullName = "C:\\myXMLFiles\\test.xml";
```

See also

Method: SetPathName (*strPath* as String)

Description
The method SetPathName sets the path of the active document. SetPathName only copies the string and does not check if the path is valid. All succeeding save operations are done into this file.

StartChanges

See also

Method: StartChanges()

Description
AfterStartChanges is executed XMLSpy will not update its editor windows until DocumentEndChanges is called. This increases performance of complex tasks to the XML structure.

Errors
1400 The object is no longer valid.

Suggestions

Property: Suggestions as Array

Description
This property contains the last valid user suggestions for this document. The XMLSpy generated suggestions can be modified before they are shown to the user in the OnBeforeShowSuggestions event.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

SwitchViewMode

See also

Method: SwitchViewMode (*nMode* as SPYViewModes) as Boolean

Return value
Returns true if view mode is switched.

Description
The method sets the current view mode of the document in XMLSpy. See also DocumentCurrentViewMode.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Invalid view mode specified.

TextView

See also

Property: TextView as TextView

Description
This property provides access to the text view functionality of the document.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

Title

See also

Property: Title as String (read-only)

Description
Title contains the file name of the document. To get the path and filename of the file use FullName.

Errors
1400 The document object is no longer valid.

1407 Invalid address for the return parameter was specified.

TransformXSL

See also

Method: TransformXSL()

Description
TransformXSL processes the XML document via the associated XSL file. See DocumentAssignXSL on how to place a reference to a XSL file into the document.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

TransformXSLEx

See also

Method: TransformXSLEx(*nAction* as SPYDialogAction)

Description
TransformXSLEx processes the XML document via the associated XSL file. The parameter specifies whether a dialog asking for the result document name should pop up or not. See DocumentAssignXSL on how to place a reference to a XSL file into the document.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

TransformXSLFO

See also

Method: TransformXSLFO()

Description
TransformXSLFO processes the XML document via the associated XSLFO file. See AssignXSLFO on how to place a reference to a XSLFO file into the document. You need to assign a FOP processor to XMLSpy before you can use this method.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

TreatXBRLInconsistenciesAsErrors

Property: TreatXBRLInconsistenciesAsErrors as Boolean

Description
If this is set to `true` the `Document.IsValid()` method will return `false` for XBRL instances containing inconsistencies as defined by the XBRL Specification. The default value of this property is `false`.

Errors
1400 The document object is no longer valid.

1407 Invalid address for the return parameter was specified.

UpdateViews

See also

Method: UpdateViews()

Description
To redraw the Enhanced Grid View and the Tree View call UpdateViews. This can be important after you changed the XMLData structure of a document. This method does not redraw the text view of XMLSpy.

Errors
1400 The document object is no longer valid.

UpdateXMLData

See also

Method: UpdateXMLData() as Boolean

Description
The XMLData tree is updated from the current view. Please note that this can fail in case of the TextView if the current XML text is not well-formed. This is not necessary if CurrentViewMode is *spyViewGrid* or *spyViewAuthentic* because these views keep the XMLData updated.

Available with TypeLibrary version 1.5

Errors
1400 The document object is no longer valid.

3.2.11 Documents

See also

Properties
Count
Item

Methods
NewAuthenticFile
NewFile
NewFileFromText
OpenAuthenticFile
OpenFile
OpenURL
OpenURLDialog

Description
This object represents the set of documents currently open in XMLSpy. Use this object to open further documents or iterate through already opened documents.

Examples

```
'----------------------------------------
'XMLSpy scripting environment - VBScript
'iterate through open documents
'----------------------------------------
Dim objDocuments
Set objDocuments = Application.Documents

For Each objDoc In objDocuments
      'do something usefulwith your document
      objDoc.SetActiveDocument()
Next
```

```
// ----------------------------------------
// XMLSpy scripting environment - JScript
// close allopen documents
// ----------------------------------------
for (var iter= new Enumerator (Application.Documents);
    !iter.atEnd();
    iter.moveNext())
{
      // MsgBox ("Closing file " + iter.item ().Name);
      iter.item ().Close (true);
}
```

Count

See also

Property: Count as long

Description
Count of open documents.

Errors
1600 Invalid Documents object
1601 Invalid input parameter

Item

See also

Method: Item (*n* as long) as Document

Description
Gets the document with the index n in this collection. Index is 1-based.

Errors
1600 Invalid Documents object
1601 Invalid input parameter

NewAuthenticFile

See also

Method: NewAuthenticFile (*strSPSPath* as String, *strXMLPath* as String) as Document

Parameters
strSPSPath
The path to the SPS document.

strXMLPath
The new XML document name.

Return Value
The method returns the new document.

Description
NewAuthenticFile creates a new XML file and opens it in Authentic View using SPS design strSPSPath.

NewFile

See also

Method: NewFile (*strFile* as String, *strType* as String) as Document

Parameters
strFile
Full path of new file.

strType
Type of new file as string (i.e. "xml", "xsd", ...)

Return Value
Returns the new file.

Description
NewFile creates a new file of type strType (i.e. "xml"). The newly created file is also the ActiveDocument.

NewFileFromText

See also

Method: NewFileFromText(*strText* as String, *strType* as String) as Document

Parameters
strText
The content of the new document in plain text.

strType
Type of the document to create (i.e. "xml").

Return Value
The method returns the new document.

Description
NewFileFromText creates a new document with strText as its content.

OpenAuthenticFile

See also

Method: OpenAuthenticFile (*strSPSPath* as String, *strXMLPath* as String) as Document

Parameters
strSPSPath
The path to the SPS document.

strXMLPath
The path to the XML document (can be empty).

Return Value
The method returns the new document.

Description
OpenAuthenticFile opens an XML file or database in Authentic View using SPS design strSPSPath.

OpenFile

See also

Method: OpenFile (*strPath* as String, *bDialog* as Boolean) as Document

Parameters
strPath
Path and file name of file to open.

bDialog
Show dialogs for user input.

Return Value
Returns the opened file on success.

Description
OpenFile opens the file strPath. If bDialog is TRUE, a file-dialog will be displayed.

Example

```
Dim objDoc As Document
Set objDoc = objSpyDocuments.OpenFile(strFile, False)
```

OpenURL

See also

Method: OpenURL (*strURL* as String, *nURLType* as SPYURLTypes, *nLoading* as SPYLoading, *strUser* as String, *strPassword* as String) as Document

Parameters
strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

nLoading
Set nLoading to 0 (zero) if you want to load it from cache or proxy. Otherwise set nLoading to 1.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentification. Can be empty.

Return Value
The method returns the opened document.

Description
OpenURL opens the URL strURL.

OpenURLDialog

See also

Method: OpenURLDialog (*strURL* as String, *nURLType* as SPYURLTypes, *nLoading* as SPYLoading, *strUser* as String, *strPassword* as String) as Document

Parameters
strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

nLoading
Set nLoading to 0 (zero) if you want to load it from cache or proxy. Otherwise set nLoading to 1.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentification. Can be empty.

Return Value
The method returns the opened document.

Description
OpenURLDialog displays the "open URL" dialog to the user and presets the input fields with the given parameters.

3.2.12 DTDSchemaGeneratorDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

DTDSchemaFormat
ValueList
TypeDetection
FrequentElements
MergeAllEqualNamed
ResolveEntities
AttributeTypeDefinition
GlobalAttributes
OnlyStringEnums

MaxEnumLength
OutputPath
OutputPathDialogAction

Description
Use this object to configure the generation of a schema or DTD. The method GenerateDTDOrSchemaEx expects a DTDSchemaGeneratorDlg as parameter to configure the generation as well as the associated user interactions.

Application

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

AttributeTypeDefinition

Property: AttributeTypeDefinition as SPYAttributeTypeDefinition

Description
Specifies how attribute definitions get merged.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

DTDSchemaFormat

Property: DTDSchemaFormat as SPYDTDSchemaFormat

Description
Sets the schema output format to DTD, or W3C.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

FrequentElements

Property: FrequentElements as SPYFrequentElements

Description
Shall the types for all elements be defined as global? Use that value *spyGlobalComplexType* to define them on global scope. Otherwise, use the value *spyGlobalElements*.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

GlobalAttributes

Property: GlobalAttributes as Boolean

Description
Shall attributes with same name and type be resolved globally?

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

MaxEnumLength

Property: MaxEnumLength as Integer

Description
Specifies the maximum number of characters allowed for enumeration names. If one value is longer than this, no enumeration will be generated.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

MergeAllEqualNamed

Property: MergeAllEqualNamed as Boolean

Description
Shall types of all elements with the same name be merged into one type?

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

OnlyStringEnums

Property: OnlyStringEnums as Boolean

Description
Specifies if enumerations will be created only for plain strings or all types of values.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

OutputPath

Property: OutputPath as String

Description
Selects the file name for the generated schema/ DTD.

Errors

3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

OutputPathDialogAction

Property: OutputPathDialogAction as SPYDialogAction

Description
Defines how the sub-dialog for selecting the schema/ DTD output path gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current value of the OutputPath property as default. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

Parent

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

ResolveEntities

Property: ResolveEntities as Boolean

Description
Shall all entities be resolved before generation starts? If yes, an info-set will be built.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

TypeDetection

Property: TypeDetection as SPYTypeDetection

Description
Specifies granularity of simple type detection.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

ValueList

Property: ValueList as Integer

Description

Generate not more than this amount of enumeration-facets per type. Set to -1 for unlimited.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

3.2.13 ElementList

See also

Properties
Count
Item

Methods
RemoveElement

Description
Element lists are used for different purposes during export and import of data. Depending on this purpose, different properties of ElementListItem are used.

It can hold
- a list of table names returned by a call to Application.GetDatabaseTables,
- a list of field names retuned by a call to Application.GetDatabaseImportElementList or Application.GetTextImportElementList,
- a field name filter list used in Application.ImportFromDatabase and Application.ImportFromText,
- a list of table names and counts for their rows and columns as returned by calls to GetExportElementList or
- a field name filter list used in DocumentExportToDatabase and DocumentExportToText.

Count

See also

Property: Count as long (read-only)

Description
Count of elements in this collection.

Item

See also

Method: Item (n as long) as ElementListItem

Description
Gets the element with the index n from this collection. The first item has index 1.

RemoveElement

See also

Method: RemoveElement(Index as long)

Description
RemoveElement removes the element Index from the collection. The first Item has index 1.

3.2.14 ElementListItem

See also

Properties
Name

ElementKind

FieldCount
RecordCount

Description
An element in an ElementList. Usage of its properties depends on the purpose of the element list. For details see ElementList.

ElementKind

See also

Property: ElementKind as SPYXMLDataKind

Description
Specifies if a field should be imported as XML element (data value of spyXMLDataElement) or attribute (data value of spyXMLDataAttr).

FieldCount

See also

Property: FieldCount as long (read-only)

Description
Count of fields (i.e. columns) in the table described by this element. This property is only valid after a call to DocumentGetExportElementList.

Name

See also

Property: Name as String (read-only)

Description
Name of the element. This is either the name of a table or a field, depending on the purpose of th element list.

RecordCount

See also

Property: RecordCount as long (read-only)

Description
Count of records (i.e. rows) in the table described by this element. This property is only valid after a call to DocumentGetExportElementList.

3.2.15 ExportSettings

See also

Properties

ElementList

EntitiesToText

ExportAllElements
SubLevelLimit

FromAttributes
FromSingleSubElements
FromTextValues

CreateKeys
IndependentPrimaryKey

Namespace

ExportCompleteXML
StartFromElement

Description
ExportSettings contains options used during export of XML data to a database or text file. See Import and export of data for a general overview.

CreateKeys

See also

Property: CreateKeys as Boolean

Description
This property turns creation of keys (i.e. primary key and foreign key) on or off. Default is True.

ElementList

See also

Property: ElementList as ElementList

Description
Default is empty list. This list of elements defines which fields will be exported. To get the list of available fields use DocumentGetExportElementList. It is possible to prevent exporting columns by removing elements from this list with ElementListRemoveElement before passing it to DocumentExportToDatabase or DocumentExportToText.

EntitiesToText

See also

Property: EntitiesToText as Boolean

Description
Defines if XML entities should be converted to text or left as they are during export. Default is

True.

ExportAllElements

See also

Property: ExportAllElements as Boolean

Description
If set to true, all elements in the document will be exported. If set to false, then ExportSettings.SubLevelLimit is used to restrict the number of sub levels to export. Default is true.

ExportCompleteXML

See also

Property: ExportCompleteXML as Boolean

Description
Defines whether the complete XML is exported or only the element specified by StartFromElement and its children. Default is True.

FromAttributes

See also

Property: FromAttributes as Boolean

Description
Set FromAttributes to false if no export data should be created from attributes. Default is True.

FromSingleSubElements

See also

Property: FromSingleSubElements as Boolean

Description
Set FromSingleSubElements to false if no export data should be created from elements. Default is True.

FromTextValues

See also

Property: FromTextValues as Boolean

Description
Set FromTextValues to false if no export data should be created from text values. Default is True.

IndependentPrimaryKey

See also

Property: IndependentPrimaryKey as Boolean

Description

Turns creation of independent primary key counter for every element on or off. If ExportSettings.CreateKeys is False, this property will be ignored. Default is True.

Namespace

See also

Property: Namespace as SPYExportNamespace

Description
The default setting removes all namespace prefixes from the element names. In some database formats the colon is not a legal character. Default is spyNoNamespace.

StartFromElement

See also

Property: StartFromElement as String

Description
Specifies the start element for the export. This property is only considered when ExportCompleteXML is false.

SubLevelLimit

See also

Property: SubLevelLimit as Integer

Description
Defines the number of sub levels to include for the export. Default is 0. This property is ignored if ExportSettings.ExportAllElements is true.

3.2.16 FileSelectionDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Dialog properties
FullName

Acceptance or cancellation of action that caused event
DialogAction

Description
The dialog object allows you to receive information about an event and pass back information to the event handler in the same way as with a user dialog. Use the FileSelectionDlgFullName to select or modify the file path and set the FileSelectionDlgDialogAction property to cancel or agree with the action that caused the event.

Application

See also

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

DialogAction

Property: DialogAction as SPYDialogAction

Description
If you want your script to perform the file selection operation without any user interaction necessary, simulate user interaction by either setting the property to *spyDialogOK(0)* or *spyDialogCancel(1)*.
To allow your script to fill in the default values but let the user see and react on the dialog, use the value *spyDialogUserInput(2)*. If you receive a FileSelectionDlg object in an event handler, *spyDialogUserInput(2)* is not supported and will be interpreted as *spyDialogOK(0)*.

Errors
2400 The object is no longer valid.
2401 Invalid value for dialog action or invalid address for the return parameter was specified.

FullName

Property: FullName as String

Description
Access the full path of the file the gets selected by the dialog. Most events that pass a FileSelectionDlg object to you allow you modify this value and thus influence the action that caused the event (e.g. load or save to a different location).

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

Parent

See also

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

3.2.17 FindInFilesDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Find
RegularExpression
Replace
DoReplace
ReplaceOnDisk
MatchWholeWord
MatchCase
SearchLocation
StartFolder
IncludeSubfolders
SearchInProjectFilesDoExternal
FileExtension
AdvancedXMLSearch
XMLElementNames
XMLElementContents
XMLAttributeNames
XMLAttributeContents
XMLComments
XMLCData
XMLPI
XMLRest
ShowResult

Description
Use this object to configure the search (or replacement) for strings in files. The method FindInFiles expects a FindInFilesDlg as parameter.

AdvancedXMLSearch

Property: AdvancedXMLSearch as Boolean

Description
Specifies if the XML search properties (XMLElementNames, XMLElementContents, XMLAttributeNames, XMLAttributeContents, XMLComments, XMLCData, XMLPI and XMLRest) are considered. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

Application

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

DoReplace

Property: DoReplace as Boolean

Description
Specifies if the matched string is replaced by the string defined in Replace. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

FileExtension

Property: FileExtension as String

Description
Specifies the file filter of the files that should be considered during the search. Multiple file filters must be delimited with a semicolon (eg: *.xml;*.dtd;a*.xsd). Use the wildcards * and ? to define the file filter.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

Find

Property: Find as String

Description
Specifies the string to search for.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

IncludeSubfolders

Property: IncludeSubfolders as Boolean

Description
Specifies if subfolders are searched too. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

MatchCase

Property: MatchCase as Boolean

Description
Specifies if the search is case sensitive. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

MatchWholeWord

Property: MatchWholeWord as Boolean

Description
Specifies whether the whole word or just a part of it must match. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

Parent

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

RegularExpression

Property: RegularExpression as Boolean

Description
Specifies if Find contains a regular expression. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

Replace

Property: Replace as String

Description
Specifies the replacement string. The matched string is only replaced if DoReplace is set true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

ReplaceOnDisk

Property: ReplaceOnDisk as Boolean

Description
Specifies if the replacement is done directly on disk. The modified file is not opened. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

SearchInProjectFilesDoExternal

Property: SearchInProjectFilesDoExternal as Boolean

Description
Specifies if the external folders in the open project are searched, when a project search is performed. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

SearchLocation

Property: SearchLocation as SPYFindInFilesSearchLocation

Description
Specifies the location of the search. The default is spyFindInFiles_Documents.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

ShowResult

Property: ShowResult as Boolean

Description
Specifies if the result is displayed in the Find in Files output window. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

StartFolder

Property: StartFolder as String

Description
Specifies the folder where the disk search starts.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLAttributeContents

Property: XMLAttributeContents as Boolean

Description
Specifies if attribute contents are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLAttributeNames

Property: XMLAttributeNames as Boolean

Description
Specifies if attribute names are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLCData

Property: XMLCData as Boolean

Description
Specifies if CData tags are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLComments

Property: XMLComments as Boolean

Description
Specifies if comments are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLElementContents

Property: XMLElementContents as Boolean

Description
Specifies if element contents are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLElementNames

Property: XMLElementNames as Boolean

Description
Specifies if element names are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLPI

Property: XMLPI as Boolean

Description
Specifies if XML processing instructions are searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

XMLRest

Property: XMLRest as Boolean

Description
Specifies if the rest of the XML (which is not covered by the other XML search properties) is searched when AdvancedXMLSearch is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

3.2.18 FindInFilesResult

See also

Properties and Methods

Standard automation properties
Application
Parent

Count
Item

Path
Document

Description
This object represents a file that matched the search criteria. It contains a list of FindInFilesResultMatch objects that describe the matching position.

Application

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

Count

Property: Count as long (read-only)

Description
Count of elements in this collection.

Document

Property: Path as Document (read-only)

Description
This property returns the Document object if the matched file is already open in XMLSpy.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

Item

Method: Item (n as long) as FindInFilesResultMatch

Description
Gets the element with the index n from this collection. The first item has index 1.

Parent

Property: Parent as FindInFilesResults (read-only)

Description
Access the parent of the object.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

Path

Property: Path as String (read-only)

Description
Returns the path of the file that matched the search criteria.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

3.2.19 FindInFilesResultMatch

See also

Properties and Methods

Standard automation properties
Application

Parent

Line
Position
Length
LineText
Replaced

Description
Contains the exact position in the file of the matched string.

Application

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Length

Property: Length as Long (read-only)

Description
Returns the length of the matched string.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Line

Property: Line as Long (read-only)

Description
Returns the line number of the match. The line numbering starts with 0.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

LineText

Property: LineText as String (read-only)

Description
Returns the text of the line.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Parent

Property: Parent as FindInFilesResult (read-only)

Description
Access the parent of the object.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Position

Property: Position as Long (read-only)

Description
Returns the start position of the match in the line. The position numbering starts with 0.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

Replaced

Property: Replaced as Boolean (read-only)

Description
True if the matched string was replaced.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

3.2.20 FindInFilesResults

See also

Properties and Methods

Standard automation properties
Application
Parent

Count
Item

Description
This is the result of the FindInFiles method. It is a list of FindInFilesResult objects.

Application

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
3600 The object is no longer valid.
3601 Invalid address for the return parameter was specified.

Count

Property: `Count` as long (read-only)

Description
Count of elements in this collection.

Item

Method: `Item (n` as long) as FindInFilesResult

Description
Gets the element with the index `n` from this collection. The first item has index 1.

Parent

Property: `Parent` as Application (read-only)

Description
Access the parent of the object.

Errors
3600 The object is no longer valid.
3601 Invalid address for the return parameter was specified.

3.2.21 GenerateSampleXMLDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

NonMandatoryAttributes
NonMandatoryElements
RepeatCount
FillAttributesWithSampleData
FillElementsWithSampleData
ContentOfNillableElementsIsNonMandatory
TryToUseNonAbstractTypes
SchemaOrDTDAssignment
LocalNameOfRootElement
NamespaceURIOfRootElement
OptionsDialogAction

Properties that are no longer supported
TakeFirstChoice - obsolete
FillWithSampleData - obsolete

Optimization - obsolete

Description
Used to set the parameters for the generation of sample XML instances based on a W3C schema or DTD.

Application

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

ChoiceMode

Property: ChoiceMode as SPYSampleXMLGenerationChoiceMode

Description
Specifies which elements will be generated.

Errors

2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

ConsiderSampleValueHints

Property: ConsiderSampleValueHints as Boolean

Description
Selects whether to use SampleValueHints or not.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

ContentOfNillableElementsIsNonMandatory

Property: ContentOfNillableElementsIsNonMandatory as Boolean

Description
If true, the contents of elements that are nillable will not be treated as mandatory.

Errors
2200 The object is no longer valid.

2201 Invalid address for the return parameter was specified.

FillAttributesWithSampleData

Property: FillAttributesWithSampleData as Boolean

Description
If true, attributes will have sample content.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

FillElementsWithSampleData

Property: FillElementsWithSampleData as Boolean

Description
If true, elements will have sample content.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

FillWithSampleData - obsolete

Property: FillWithSampleData as Boolean

Description
Do no longer access this property. Use FillAttributesWithSampleData and FillElementsWithSampleData, instead.

Errors
0001 The property is no longer accessible.

LocalNameOfRootElement

Property: LocalNameOfRootElement as String

Description
Specifies the local name of the root element for the generated sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

NamespaceURIOfRootElement

Property: NamespaceURIOfRootElement as String

Description
Specifies the namespace URI of the root element for the generated sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

NonMandatoryAttributes

Property: NonMandatoryAttributes as Boolean

Description
If true attributes which are not mandatory are created in the sample XML instance file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

NonMandatoryElements

Property: NonMandatoryElements as Boolean

Description
If `true`, elements which are not mandatory are created in the sample XML instance file.

Errors
2200 The object is no longer valid.
2201 Invalid address was specified for the return parameter.

Optimization - obsolete

Property: Optimization as SPYSampleXMLGenerationOptimization

Description
Do not use this property any longer. Use ChoiceMode and NonMandatoryElements.

Errors
0001 The property is no longer accessible.

OptionsDialogAction

Property: OptionsDialogAction as SPYDialogAction

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors
2200 The object is no longer valid.
2201 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

RepeatCount

Property: RepeatCount as long

Description
Number of elements to create for repeated types.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

SampleValueHints

Property: SampleValueHints as SPYSampleXMLGenerationSampleValueHints

Description
Specifies how to select data for the generated sample file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

SchemaOrDTDAssignment

Property: SchemaOrDTDAssignment as SPYSampleXMLGenerationSchemaOrDTDAssignment

Description
Specifies in which way a reference to the related schema or DTD - which is this document - will be generated into the sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

TakeFirstChoice - obsolete

Property: TakeFirstChoice as Boolean

Description
Do no longer use this property.

Errors
0001 The property is no longer accessible.

TryToUseNonAbstractTypes

Property: TryToUseNonAbstractTypes as Boolean

Description
If true, tries to use a non-abstract type for xsi:type, if element has an abstract type.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

3.2.22 GridView

See also

Methods
Deselect
Select

SetFocus

Properties
CurrentFocus

IsVisible

Description
GridView Class

Events

OnBeforeDrag

See also

Event: OnBeforeDrag() as Boolean

XMLSpy scripting environment - VBScript:
```
Function On_BeforeDrag()
	'On_BeforeStartEditing = False  'to prohibit dragging
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeDrag()
{
	// return false; /* to prohibit dragging */
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent(4, ...) // nEventId = 4
```

Description
This event gets fired on an attempt to drag an XMLData element on the grid view. Return *false* to prevent dragging the data element to a different position.

OnBeforeDrop

See also

Event: OnBeforeDrop(*objXMLData* as XMLData) as Boolean

XMLSpy scripting environment - VBScript:
```
Function On_BeforeDrop(objXMLData)
	'On_BeforeStartEditing = False  'to prohibit dropping
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeDrop(objXMLData)
```

```
{
	// return false; /* to prohibit dropping */
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn OnEvent(5, ...) // nEventId = 5
```

Description

This event gets fired on an attempt to drop a previously dragged XMLData element on the grid view. Return *false* to prevent the data element to be moved from its original position to the drop destination position.

OnBeforeStartEditing

See also

Event: `OnBeforeStartEditing(objXMLData as XMLData,bEditingName as Boolean)as Boolean`

XMLSpy scripting environment - VBScript:

```
Function On_BeforeStartEditing(objXMLData,bEditingName)
	'On_BeforeStartEditing = False  'to prohibit editing the field
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeStartEditing(objXMLData,bEditingName)
{
	// return false; /* to prohibit editing the field */
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn OnEvent(1, ...) // nEventId = 1
```

Description

This event gets fired before the editing mode for a grid cell gets entered. If the parameter *bEditingName* is true, the name part of the element will be edited, it its value is false, the value part will be edited.

OnEditingFinished

See also

Event: `OnEditingFinished(objXMLData as XMLData,bEditingName as Boolean)`

XMLSpy scripting environment - VBScript:

```
Function On_EditingFinished(objXMLData,bEditingName)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_EditingFinished(objXMLData,bEditingName)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn OnEvent(2, ...) // nEventId = 2
```

Description
This event gets fired when the editing mode of a grid cell gets left. The parameter *bEditingName* specifies if the name part of the element has been edited.

OnFocusChanged

See also

Event: `OnFocusChanged(objXMLData as XMLData,bSetFocus as Boolean,bEditingName as Boolean)`

XMLSpy scripting environment - VBScript:
```
Function On_FocusChanged(objXMLData,bSetFocus,bEditingName)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_FocusChanged(objXMLData,bSetFocus,bEditingName)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent(3, ...) // nEventId = 3
```

Description
This event gets fired whenever a grid cell receives or looses the cursor focus. If the parameter *bEditingName* is *true*, focus of the name part of the grid element has changed. Otherwise, focus of the value part has changed.

CurrentFocus

See also

Property: `CurrentFocus as XMLData`

Description
Holds the XML element with the current focus. This property is read-only.

Deselect

See also

Method: `Deselect(pData as XMLData)`

Description
Deselects the element `pData` in the grid view.

IsVisible

See also

Property: `IsVisible as Boolean`

Description
True if the grid view is the active view of the document. This property is read-only.

Select

See also

Method: Select(*pData* as XMLData)

Description
Selects the XML element pData in the grid view.

SetFocus

See also

Method: SetFocus(*pFocusData* as XMLData)

Description
Sets the focus to the element pFocusData in the grid view.

3.2.23 SchemaDocumentationDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Interaction and visibility properties
OutputFile
OutputFileDialogAction
OptionsDialogAction
ShowProgressBar
ShowResult

Document generation options and methods
OutputFormat
UseFixedDesign
SPSFile
EmbedDiagrams
DiagramFormat
MultipleOutputFiles
EmbedCSSInHTML
CreateDiagramsFolder
GenerateRelativeLinks

IncludeAll
IncludeIndex
IncludeGlobalAttributes
IncludeGlobalElements
IncludeLocalAttributes
IncludeLocalElements
IncludeGroups
IncludeComplexTypes
IncludeSimpleTypes
IncludeAttributeGroups
IncludeRedefines
IncludeReferencedSchemas

AllDetails

ShowDiagram
ShowNamespace
ShowType
ShowChildren
ShowUsedBy
ShowProperties
ShowSingleFacets
ShowPatterns
ShowEnumerations
ShowAttributes
ShowIdentityConstraints
ShowAnnotations
ShowSourceCode

Description
This object combines all options for schema document generation as they are available through user interface dialog boxes in XMLSpy. The document generation options are initialized with the values used during the last generation of schema documentation. However, before using the object you have to set the SetOutputFile property to a valid file path. Use OptionsDialogAction, OutputFileDialogAction and ShowProgressBar to specify the level of user interaction desired. You can use IncludeAll and AllDetails to set whole option groups at once or the individual properties to operate on a finer granularity.

AllDetails

See also

Method: AllDetails (ibDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

Errors
2900 The object is no longer valid.

Application

See also

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

CreateDiagramsFolder

See also

Property: CreateDiagramsFolder as Boolean

Description
Set this property to true, to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is false.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

DiagramFormat

See also

Property: DiagramFormat as SPYImageKind

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is PNG.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

EmbedCSSInHTML

See also

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

GenerateRelativeLinks

See also

Property: GenerateRelativeLinks as Boolean

Description
Set this property to true, to create relative paths to local files. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is false.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeAll

See also

Method: IncludeAll(ibInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
2900 The object is no longer valid.

IncludeAttributeGroups

See also

Property: IncludeAttributeGroups as Boolean

Description
Set this property to true, to include attribute groups in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeComplexTypes

See also

Property: IncludeComplexTypes as Boolean

Description
Set this property to true, to include complex types in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation.

The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeGlobalAttributes

See also

Property: IncludeGlobalAttributes as Boolean

Description
Set this property to true, to include global attributes in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeGlobalElements

See also

Property: IncludeGlobalElements as Boolean

Description
Set this property to true, to include global elements in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeGroups

See also

Property: IncludeGroups as Boolean

Description
Set this property to true, to include groups in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeIndex

See also

Property: IncludeIndex as Boolean

Description
Set this property to true, to include an index in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeLocalAttributes

See also

Property: IncludeLocalAttributes as Boolean

Description
Set this property to true, to include local attributes in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeLocalElements

See also

Property: IncludeLocalElements as Boolean

Description
Set this property to true, to include local elements in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeRedefines

See also

Property: IncludeRedefines as Boolean

Description
Set this property to true, to include redefines in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeReferencedSchemas

See also

Property: IncludeReferencedSchemas as Boolean

Description
Set this property to true, to include referenced schemas in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeSimpleTypes

See also

Property: IncludeSimpleTypes as Boolean

Description
Set this property to true, to include simple types in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

MultipleOutputFiles

See also

Property: MultipleOutputFiles as Boolean

Description
Set this property to true, to split the documentation files. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is false.

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OptionsDialogAction

See also

Property: OptionsDialogAction as SPYDialogAction

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFile

See also

Property: OutputFile as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to DocumentGenerateSchemaDocumentation.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

OutputFileDialogAction

See also

Property: OutputFileDialogAction as SPYDialogAction

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFormat

See also

Property: OutputFormat as SPYSchemaDocumentationFormat

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is HTML.

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent

See also

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowAnnotations

See also

Property: ShowAnnotations as Boolean

Description
Set this property to true, to show the annotations to a type definition in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowAttributes

See also

Property: ShowAttributes as Boolean

Description
Set this property to true, to show the type definitions attributes in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowChildren

See also

Property: ShowChildren as Boolean

Description
Set this property to true, to show the children of a type definition as links in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors

2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowDiagram

See also

Property: ShowDiagram as Boolean

Description
Set this property to true, to show type definitions as diagrams in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowEnumerations

See also

Property: ShowEnumerations as Boolean

Description
Set this property to true, to show the enumerations contained in a type definition in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowIdentityConstraints

See also

Property: ShowIdentityConstraints as Boolean

Description
Set this property to true, to show a type definitions identity constraints in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowNamespace

See also

Property: ShowNamespace as Boolean

Description
Set this property to true, to show the namespace of type definitions in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowPatterns

See also

Property: ShowPatterns as Boolean

Description
Set this property to true, to show the patterns of a type definition in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowProgressBar

See also

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowProperties

See also

Property: ShowProperties as Boolean

Description
Set this property to true, to show the type definition properties in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowResult

See also

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in XMLSpy. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowSingleFacets

See also

Property: ShowSingleFacets as Boolean

Description
Set this property to true, to show the facets of a type definition in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowSourceCode

See also

Property: ShowSourceCode as Boolean

Description
Set this property to true, to show the XML source code for type definitions in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowType

See also

Property: ShowType as Boolean

Description
Set this property to true, to show the type of type definitions in the schema documentation. The property is initialized with the value used during the last call to

DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowUsedBy

See also

Property: ShowUsedBy as Boolean

Description
Set this property to true, to show the used-by relation for type definitions in the schema documentation. The property is initialized with the value used during the last call to DocumentGenerateSchemaDocumentation. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

SPSFile

See also

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

UseFixedDesign

See also

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

3.2.24 SpyProject

See also

Methods

CloseProject
SaveProject
SaveProjectAs

Properties
RootItems
ProjectFile

Description
SpyProject Class

CloseProject

See also

Declaration: CloseProject(*bDiscardChanges* as Boolean, *bCloseFiles* as Boolean, *bDialog* as Boolean)

Parameters
bDiscardChanges
Set bDiscardChanges to FALSE if you want to save the changes of the open project files and the project.

bCloseFiles
Set bCloseFiles to TRUE to close all open project files.

bDialog
Show dialogs for user input.

Description
CloseProject closes the current project.

ProjectFile

See also

Declaration: ProjectFile as String

Description
Path and filename of the project.

RootItems

See also

Declaration: RootItems as SpyProjectItems

Description
Root level of collection of project items.

SaveProject

See also

Declaration: SaveProject

Description
SaveProject saves the current project.

SaveProjectAs

See also

Declaration: SaveProjectAs (strPath as String, bDialog as Boolean)

Parameters
strPath
Full path with file name of new project file.

bDialog
If bDialog is TRUE, a file- dialog will be displayed.

Description
SaveProjectAs stores the project data into a new location.

3.2.25 SpyProjectItem

See also

Methods
Open

Properties
ChildItems
ParentItem
FileExtensions
ItemType
Name
Path
ValidateWith
XMLForXSLTransformation
XSLForXMLTransformation
XSLTransformationFileExtension
XSLTransformationFolder

Description
SpyProjectItem Class

ChildItems

See also

Declaration: ChildItems as SpyProjectItems

Description
If the item is a folder, ChildItems is the collection of the folder content.

FileExtensions

See also

Declaration: FileExtensions as String

Description
Used to set the file extensions if the project item is a folder.

ItemType

See also

Declaration: ItemType as SPYProjectItemTypes

Description
This property is read-only.

Name

See also

Declaration: Name as String

Description
Name of the project item. This property is read-only.

Open

See also

Declaration: Open as Document

Return Value
The project item opened as document.

Description
Opens the project item.

ParentItem

See also

Declaration: ParentItem as SpyProjectItem

Description
Parent item of the current project item. Can be NULL (Nothing) if the project item is a top-level item.

Path

See also

Declaration: Path as String

Description
Path of project item. This property is read-only.

ValidateWith

See also

Declaration: ValidateWith as String

Description
Used to set the schema/ DTD for validation.

XMLForXSLTransformation

See also

Declaration: XMLForXSLTransformation as String

Description
Used to set the XML for XSL transformation.

XSLForXMLTransformation

See also

Declaration: XSLForXMLTransformation as String

Description
Used to set the XSL for XML transformation.

XSLTransformationFileExtension

See also

Declaration: XSLTransformationFileExtension as String

Description
Used to set the file extension for XSL transformation output files.

XSLTransformationFolder

See also

Declaration: XSLTransformationFolder as String

Description
Used to set the destination folder for XSL transformation output files.

3.2.26 SpyProjectItems

See also

Methods
AddFile
AddFolder
AddURL
RemoveItem

Properties
Count
Item

Description
SpyProjectItems Class

AddFile

See also

Declaration: AddFile (*strPath* as String)

Parameters

strPath
Full path with file name of new project item

Description
The method adds a new file to the collection of project items.

AddFolder

See also

Declaration: AddFolder(*strName* as String)

Parameters

strName
Name of the new folder.

Description
The method AddFolder adds a folder with the name strName to the collection of project items.

AddURL

See also

Declaration: AddURL (*strURL* as String, *nURLType* as SPYURLTypes, *strUser* as String, *strPassword* as String, *bSave* as Boolean)

Description

strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentification. Can be empty.

bSave
Save user and password information.

Description
The method adds an URL item to the project collection.

Count

See also

Declaration: Count as long

Description
This property gets the count of project items in the collection. The property is read-only.

Item

See also

Declaration: Item (*n* as long) as SpyProjectItem

Description
Retrieves the n-th element of the collection of project items. The first item has index 1.

RemoveItem

See also

Declaration: RemoveItem (*pItem* as SpyProjectItem)

Description
RemoveItem deletes the item pItem from the collection of project items.

3.2.27 TextImportExportSettings

See also

Properties for import only
ImportFile

Properties for export only
DestinationFolder
FileExtension
CommentIncluded
RemoveDelimiter
RemoveNewline

Properties for import and export
HeaderRow
FieldDelimiter
EnclosingCharacter
Encoding
EncodingByteOrder

Description
TextImportExportSettings contains options common to text import and export functions.

CommentIncluded

See also

Property: CommentIncluded as Boolean

Description
This property tells whether additional comments are added to the generated text file. Default is true. This property is used only when exporting to text files.

DestinationFolder

See also

Property: DestinationFolder as String

Description

The property DestinationFolder sets the folder where the created files are saved during text export.

EnclosingCharacter

See also

Property: EnclosingCharacter as SPYTextEnclosing

Description
This property defines the character that encloses all field values for import and export. Default is spyNoEnclosing.

Encoding

See also

Property: Encoding as String

Description
The property Encoding sets the character encoding for the text files for importing and exporting.

EncodingByteOrder

See also

Property: EncodingByteOrder as SPYEncodingByteOrder

Description
The property EncodingByteOrder sets the byte order for Unicode characters. Default is spyNONE.

FieldDelimiter

See also

Property: FieldDelimiter as SPYTextDelimiters

Description
The property FieldDelimiter defines the delimiter between the fields during import and export. Default is spyTabulator.

FileExtension

See also

Property: FileExtension as String

Description
This property sets the file extension for files created on text export.

HeaderRow

See also

Property: HeaderRow as Boolean

Description

The property HeaderRow is used during import and export. Set HeaderRow true on import, if the first line of the text file contains the names of the columns. Set HeaderRow true on export, if the first line in the created text files should contain the name of the columns. Default value is true.

ImportFile

See also

Property: ImportFile as String

Description
This property is used to set the text file for import. The string has to be a full qualified path. See also Import and Export.

RemoveDelimiter

See also

Property: RemoveDelimiter as Boolean

Description
The property RemoveDelimiter defines whether characters in the text that are equal to the delimiter character are removed. Default is false. This property is used only when exporting to text files.

RemoveNewline

See also

Property: RemoveNewline as Boolean

Description
The property RemoveNewline defines whether newline characters in the text are removed. Default is false. This property is used only when exporting to text files.

3.2.28 TextView

See also

Properties and Methods

Application
Parent

LineFromPosition
PositionFromLine
LineLength
SelText
GetRangeText
ReplaceText
MoveCaret
GoToLineChar
SelectText
SelectionStart
SelectionEnd
Text
LineCount
Length

Description

Events

OnBeforeShowSuggestions

See also

Event: OnBeforeShowSuggestions () as Boolean

Description
This event gets fired before a suggestion window is shown. The Document property Suggestions contains a string array that is recommended to the user. It is possible to modify the displayed recommendations during this event. Before doing so you have to assign an empty array to the Suggestions property. The best location for this is the OnDocumentOpened event. To prevent the suggestion window to show up return false and true to continue its display.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_BeforeShowSuggestions()
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeShowSuggestions()
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent(33, ...)          // nEventId = 33
```

OnChar

See also

Event: OnChar (nChar as Long, bExistSuggestion as Boolean) as Boolean

Description
This event gets fired on each key stroke. The parameter nChar is the key that was pressed and bExistSuggestions tells whether a XMLSpy generated suggestions window is displayed after this key. The Document property Suggestions contains a string array that is recommended to the user. It is possible to modify the displayed recommendations during this event. Before doing so you have to assign an empty array to the Suggestions property. The best location for this is the OnDocumentOpened event. To prevent the suggestion window to show up return false and true to continue its display.
It is also possible to create a new suggestions window when none is provided by XMLSpy. Set the Document property Suggestions to a string array with your recommendations and return true. This event is fired before the OnBeforeShowSuggestions event. If you prevent to show the suggestion window by returning false then OnBeforeShowSuggestions is not fired.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:

```
Function On_Char (nChar,bExistSuggestions)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_Char (nChar,bExistSuggestions)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugInOnEvent(35, ...)          // nEventId = 35
```

Application

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

GetRangeText

Method: GetRangeText(nStart as Long, nEnd as Long) as String

Description
Returns the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

GoToLineChar

Method: GoToLineChar(nLine as Long, nChar as Long)

Description
Moves the caret to the specified line and character position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

Length

Property: Length as Long

Description
Returns the character count of the document.

Errors
3900 The object is no longer valid.

3901 Invalid address for the return parameter was specified.

LineCount

Property: LineCount as Long

Description
Returns the number of lines in the document.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

LineFromPosition

Method: LineFromPosition(nCharPos as Long) as Long

Description
Returns the line number of the character position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

LineLength

Method: LineLength(nLine as Long) as Long

Description
Returns the length of the line.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

MoveCaret

Method: MoveCaret(nDiff as Long)

Description
Moves the caret nDiff characters.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

Parent

Property: `Parent` as Document (read-only)

Description
Access the parent of the object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

PositionFromLine

Method: PositionFromLine(nLine as Long) as Long

Description
Returns the start position of the line.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ReplaceText

Method: ReplaceText(nPosFrom as Long, nPosTill as Long, sText as String)

Description
Replaces the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SelectionEnd

Property: SelectionEnd as Long

Description
Returns/sets the text selection end position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SelectionStart

Property: SelectionStart as Long

Description
Returns/sets the text selection start position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SelectText

Method: SelectText(nPosFrom as Long, nPosTill as Long)

Description
Selects the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SelText

Property: SelText as String

Description
Returns/sets the selected text.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

Text

Property: Text as String

Description
Returns/sets the document text.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

3.2.29 WSDLDocumentationDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Interaction and visibility properties
GlobalElementsAndTypesOnly
OptionsDialogAction
OutputFile
OutputFileDialogAction
SeparateSchemaDocument
ShowProgressBar
ShowResult

Document generation options and methods
OutputFormat
UseFixedDesign
SPSFile
EmbedDiagrams
DiagramFormat
MultipleOutputFiles
EmbedCSSInHTML
CreateDiagramsFolder

IncludeAll
IncludeBinding
IncludeImportedWSDLFiles
IncludeMessages
IncludeOverview
IncludePortType
IncludeService
IncludeTypes

AllDetails
ShowBindingDiagram
ShowExtensibility
ShowMessageParts
ShowPort
ShowPortTypeDiagram
ShowPortTypeOperations
ShowServiceDiagram
ShowSourceCode
ShowTypesDiagram
ShowUsedBy

Description
This object combines all options for WSDL document generation as they are available through user interface dialog boxes in XMLSpy. The document generation options are initialized with the values used during the last generation of WSDL documentation. However, before using the object you have to set the OutputFile property to a valid file path. Use OptionsDialogAction, OutputFileDialogAction and ShowProgressBar to specify the level of user interaction desired. You can use IncludeAll and AllDetails to set whole option groups at once or the individual properties to operate on a finer granularity.

AllDetails

See also

Method: AllDetails (ibDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

Errors
4300 The object is no longer valid.

Application

See also

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

CreateDiagramsFolder

See also

Property: CreateDiagramsFolder as Boolean

Description

Set this property to true, to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is false.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

DiagramFormat

See also

Property: DiagramFormat as SPYImageKind

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is PNG.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

EmbedCSSInHTML

See also

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

GlobalElementsAndTypesOnly

See also

Property: GlobalElementsAndTypesOnly as Boolean

Description
Returns/sets a value indicating whether a full Schema documentation is done or only Global Elements and Types are documented.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeAll

See also

Method: IncludeAll(ibInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
4300 The object is no longer valid.

IncludeBinding

See also

Property: IncludeBinding as Boolean

Description
Set this property to true, to include bindings in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeImportedWSDLFiles

See also

Property: IncludeImportedWSDLFiles as Boolean

Description
Set this property to true, to include imported WSDL finles in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeMessages

See also

Property: IncludeMessages as Boolean

Description
Set this property to true, to include messages in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeOverview

See also

Property: IncludeOverview as Boolean

Description
Set this property to true, to include an overview in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludePortType

See also

Property: IncludePortType as Boolean

Description
Set this property to true, to include port types in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeService

See also

Property: IncludeService as Boolean

Description
Set this property to true, to include services in the WSDL documentation. The property is

initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

IncludeTypes

See also

Property: IncludeTypes as Boolean

Description
Set this property to true, to include types in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

MultipleOutputFiles

See also

Property: MultipleOutputFiles as Boolean

Description
Set this property to true, to split the documentation files. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is false.

Errors
3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OptionsDialogAction

See also

Property: OptionsDialogAction as SPYDialogAction

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors
3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFile

See also

Property: OutputFile as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to DocumentGenerateWSDLDocumentation.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

OutputFileDialogAction

See also

Property: OutputFileDialogAction as SPYDialogAction

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors
3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFormat

See also

Property: OutputFormat as SPYSchemaDocumentationFormat

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is HTML.

Errors
3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent

See also

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SeparateSchemaDocument

See also

Property: SeparateSchemaDocument as Boolean

Description
Returns/sets a value indicating whether the Schema documentation should be placed in a separate document.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowBindingDiagram

See also

Property: ShowBindingDiagram as Boolean

Description
Set this property to true, to show binding diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowExtensibility

See also

Property: ShowExtensibility as Boolean

Description
Set this property to true, to show service and binding extensibilities in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowMessageParts

See also

Property: ShowMessageParts as Boolean

Description
Set this property to true, to show message parts of messges in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowPort

See also

Property: ShowPort as Boolean

Description
Set this property to true, to show service ports in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowPortTypeDiagram

See also

Property: ShowPortTypeDiagram as Boolean

Description
Set this property to true, to show port type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowPortTypeOperations

See also

Property: ShowPortTypeOperations as Boolean

Description
Set this property to true, to show port type operations in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowProgressBar

See also

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowResult

See also

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in XMLSpy. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowServiceDiagram

See also

Property: ShowServiceDiagram as Boolean

Description
Set this property to true, to show service diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowSourceCode

See also

Property: ShowSourceCode as Boolean

Description
Set this property to true, to show source code for the includes in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowTypesDiagram

See also

Property: ShowTypesDiagram as Boolean

Description
Set this property to true, to show type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowUsedBy

See also

Property: ShowUsedBy as Boolean

Description
Set this property to true, to show the used-by relation for types, bindings and messages definitions in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDLDocumentation. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

UseFixedDesign

See also

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

SPSFile

See also

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

3.2.30 WSDL20DocumentationDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Interaction and visibility properties
GlobalElementsAndTypesOnly
OptionsDialogAction
OutputFile
OutputFileDialogAction
SeparateSchemaDocument
ShowProgressBar
ShowResult

Document generation options and methods
OutputFormat
UseFixedDesign
SPSFile
EmbedDiagrams
DiagramFormat
MultipleOutputFiles
EmbedCSSInHTML
CreateDiagramsFolder

IncludeAll
IncludeBinding
IncludeImportedWSDLFiles
IncludeInterface
IncludeOverview
IncludeService
IncludeTypes

AllDetails
ShowBindingDiagram
ShowExtensibility
ShowEndpoint
ShowFault
ShowInterfaceDiagram
ShowOperation
ShowServiceDiagram
ShowSourceCode
ShowTypesDiagram
ShowUsedBy

Description
This object combines all options for WSDL document generation as they are available through user interface dialog boxes in XMLSpy. The document generation options are initialized with the

values used during the last generation of WSDL documentation. However, before using the object you have to set the `OutputFile` property to a valid file path. Use `OptionsDialogAction`, `OutputFileDialogAction` and `ShowProgressBar` to specify the level of user interaction desired. You can use `IncludeAll` and `AllDetails` to set whole option groups at once or the individual properties to operate on a finer granularity.

AllDetails

See also

Method: `AllDetails` (`ibDetailsOn` as `Boolean`)

Description
Use this method to turn all details options on or off.

Errors
4300 The object is no longer valid.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy application object.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

CreateDiagramsFolder

See also

Property: `CreateDiagramsFolder` as `Boolean`

Description
Set this property to `true`, to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to `DocumentGenerateWSD20LDocumentation`. The default for the first run is false.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

DiagramFormat

See also

Property: DiagramFormat as SPYImageKind

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is PNG.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

EmbedCSSInHTML

See also

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

GlobalElementsAndTypesOnly

See also

Property: GlobalElementsAndTypesOnly as Boolean

Description
Returns/sets a value indicating whether a full Schema documentation is done or only Global Elements and Types are documented.

Errors
3900 The object is no longer valid.

3901 Invalid address for the return parameter was specified.

IncludeAll

See also

Method: IncludeAll(ibInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
4300 The object is no longer valid.

IncludeBinding

See also

Property: IncludeBinding as Boolean

Description
Set this property to true, to include bindings in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

IncludeImportedWSDLFiles

See also

Property: IncludeImportedWSDLFiles as Boolean

Description
Set this property to true, to include imported WSDL finles in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

IncludeInterface

See also

Property: IncludeInterface as Boolean

Description
Set this property to true, to include interfaces in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

IncludeOverview

See also

Property: IncludeOverview as Boolean

Description
Set this property to true, to include an overview in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

IncludeService

See also

Property: IncludeService as Boolean

Description
Set this property to true, to include services in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

IncludeTypes

See also

Property: IncludeTypes as Boolean

Description
Set this property to true, to include types in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

MultipleOutputFiles

See also

Property: MultipleOutputFiles as Boolean

Description

Set this property to true, to split the documentation files. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is false.

Errors
4300 The object is no longer valid.
4301 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OptionsDialogAction

See also

Property: OptionsDialogAction as SPYDialogAction

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors
4300 The object is no longer valid.
4301 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFile

See also

Property: OutputFile as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to DocumentGenerateWSDL20Documentation.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

OutputFileDialogAction

See also

Property: OutputFileDialogAction as SPYDialogAction

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors
4300 The object is no longer valid.
4301 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFormat

See also

Property: OutputFormat as SPYSchemaDocumentationFormat

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is HTML.

Errors
4300 The object is no longer valid.
4301 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent

See also

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

SeparateSchemaDocument

See also

Property: SeparateSchemaDocument as Boolean

Description
Returns/sets a value indicating whether the Schema documentation should be placed in a separate document.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

ShowBindingDiagram

See also

Property: ShowBindingDiagram as Boolean

Description
Set this property to true, to show binding diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.

4301 Invalid address for the return parameter was specified.

ShowEndpoint

See also

Property: ShowEndpoint as Boolean

Description
Set this property to true, to show service endpoints in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowExtensibility

See also

Property: ShowExtensibility as Boolean

Description
Set this property to true, to show service and binding extensibilities in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowFault

See also

Property: ShowFault as Boolean

Description
Set this property to true, to show faults in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowInterfaceDiagram

See also

Property: ShowInterfaceDiagram as Boolean

Description
Set this property to true, to show interface diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowOperation

See also

Property: ShowOperation as Boolean

Description
Set this property to true, to show interface and binding operations in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowProgressBar

See also

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowResult

See also

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in XMLSpy. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowServiceDiagram

See also

Property: ShowServiceDiagram as Boolean

Description
Set this property to true, to show service diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowSourceCode

See also

Property: ShowSourceCode as Boolean

Description
Set this property to true, to show source code for the includes in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowTypesDiagram

See also

Property: ShowTypesDiagram as Boolean

Description
Set this property to true, to show type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

ShowUsedBy

See also

Property: ShowUsedBy as Boolean

Description
Set this property to true, to show the used-by relation for types, bindings and messages definitions in the WSDL documentation. The property is initialized with the value used during the last call to DocumentGenerateWSDL20Documentation. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

SPSFile

See also

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

UseFixedDesign

See also

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

3.2.31 XBRLDocumentationDlg

See also

Properties and Methods

Standard automation properties
Application
Parent

Interaction and visibility properties
OptionsDialogAction
OutputFile
OutputFileDialogAction
ShowProgressBar
ShowResult

Document generation options and methods
OutputFormat
UseFixedDesign
SPSFile
EmbedDiagrams
DiagramFormat
EmbedCSSInHTML

CreateDiagramsFolder

IncludeAll
IncludeOverview
IncludeNamespacePrefixes
IncludeGlobalElements
IncludeDefinitionLinkroles
IncludePresentationLinkroles
IncludeCalculationLinkroles

AllDetails
ShowDiagram
ShowSubstitutiongroup
ShowItemtype
ShowBalance
ShowPeriod
ShowAbstract
ShowNillable
ShowLabels
ShowReferences
ShowLinkbaseReferences

ShortQualifiedName
ShowImportedElements

Description
This object combines all options for XBRL document generation as they are available through user interface dialog boxes in XMLSpy. The document generation options are initialized with the values used during the last generation of XBRL documentation. However, before using the object you have to set the OutputFile property to a valid file path. Use OptionsDialogAction, OutputFileDialogAction and ShowProgressBar to specify the level of user interaction desired. You can use IncludeAll and AllDetails to set whole option groups at once or the individual properties to operate on a finer granularity.

AllDetails

See also

Method: AllDetails (ibDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

Errors
4400 The object is no longer valid.

Application

See also

Property: Application as Application (read-only)

Description
Access the XMLSpy application object.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

CreateDiagramsFolder

See also

Property: CreateDiagramsFolder as Boolean

Description
Set this property to true, to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is false.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

DiagramFormat

See also

Property: DiagramFormat as SPYImageKind

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is PNG.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

EmbedCSSInHTML

See also

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

EmbedDiagrams

See also

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

IncludeAll

See also

Method: IncludeAll(ibInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
4400 The object is no longer valid.

IncludeCalculationLinkroles

See also

Property: IncludeCalculationLinkroles as Boolean

Description
Set this property to true, to include calculation linkroles in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

IncludeDefinitionLinkroles

See also

Property: IncludeDefinitionLinkroles as Boolean

Description
Set this property to true, to include definition linkroles in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors

4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

IncludeGlobalElements

See also

Property: IncludeGlobalElements as Boolean

Description
Set this property to true, to include global elements in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

IncludeNamespacePrefixes

See also

Property: IncludeNamespacePrefixes as Boolean

Description
Set this property to true, to include namespace prefixes in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

IncludeOverview

See also

Property: IncludeOverview as Boolean

Description
Set this property to true, to include an overview in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

IncludePresentationLinkroles

See also

Property: IncludePresentationLinkroles as Boolean

Description
Set this property to true, to include presentation linkroles in the XBRL documentation. The property is initialized with the value used during the last call to

DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

OptionsDialogAction

See also

Property: OptionsDialogAction as SPYDialogAction

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors
4400 The object is no longer valid.
4401 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFile

See also

Property: OutputFile as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to DocumentGenerateXBRLDocumentation.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

OutputFileDialogAction

See also

Property: OutputFileDialogAction as SPYDialogAction

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors
4400 The object is no longer valid.
4401 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

OutputFormat

See also

Property: OutputFormat as SPYSchemaDocumentationFormat

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is HTML.

Errors
4400 The object is no longer valid.
4401 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent

See also

Property: Parent as Dialogs (read-only)

Description
Access the parent of the object.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShortQualifiedName

See also

Property: ShortQualifiedName as Boolean

Description
Set this property to true, to use short qualified names in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowAbstract

See also

Property: ShowAbstract as Boolean

Description
Set this property to true, to show abstracts in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors

4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowBalance

See also

Property: ShowBalance as Boolean

Description
Set this property to true, to show balances in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowDiagram

See also

Property: ShowDiagram as Boolean

Description
Set this property to true, to show diagrams in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowImportedElements

See also

Property: ShowImportedElements as Boolean

Description
Set this property to true, to show imported elements in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowItemtype

See also

Property: ShowItemtype as Boolean

Description
Set this property to true, to show item types in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowLabels

See also

Property: ShowLabels as Boolean

Description
Set this property to true, to show labels in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowLinkbaseReferences

See also

Property: ShowLinkbaseReferences as Boolean

Description
Set this property to true, to show linkbase references in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowNillable

See also

Property: ShowNillable as Boolean

Description
Set this property to true, to show nillable properties in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowPeriod

See also

Property: ShowPeriod as Boolean

Description
Set this property to true, to show periods in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowProgressBar

See also

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowReferences

See also

Property: ShowReferences as Boolean

Description
Set this property to true, to show references in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowResult

See also

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was

successful. HTML documentation will be opened in XMLSpy. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

ShowSubstitutiongroup

See also

Property: ShowSubstitutiongroup as Boolean

Description
Set this property to true, to show substitution groups in the XBRL documentation. The property is initialized with the value used during the last call to DocumentGenerateXBRLDocumentation. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

SPSFile

See also

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

UseFixedDesign

See also

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

3.2.32 XMLData

See also

Properties
Kind
Name
TextValue

HasChildren
MayHaveChildren
Parent

Methods
GetFirstChild
GetNextChild
GetCurrentChild

InsertChild
InsertChildAfter
InsertChildBefore
AppendChild

EraseAllChildren
EraseChild
EraseCurrentChild

IsSameNode

CountChildren
CountChildrenKind

GetChild
GetChildAttribute
GetChildElement
GetChildKind
GetNamespacePrefixForURI

HasChildrenKind
SetTextValueXMLEncoded

Description
The XMLData interface provides direct XML-level access to a document. You can read and directly modify the XML representation of the document. However, please, note the following restrictions:

- The XMLData representation is only valid when the document is shown in grid view or authentic view.
- When in authentic view, additional XMLData elements are automatically inserted as parents of each visible document element. Typically this is an XMLData of kind `spyXMLDataElement` with the Name property set to 'Text'.
- When you use the XMLData interface while in a different view mode you will not receive errors, but changes are not reflected to the view and might get lost during the next view switch.

Note also:

- Setting a new text value for an XML element is possible if the element does not have non-text children. A text value can be set even if the element has attributes.

- When setting a new text value for an XML element which has more than one text child, the latter will be deleted and replaced by one new text child.
- When reading the text value of an XML element which has more than one text child, only the value of the first text child will be returned.

Objects of this class represent the different atomic parts of an XML document. See the enumeration type SPYXMLDataKind for the available part types. Each part knows its children, thus forming a XMLData tree with DocumentRootElement at its top. To get the top element of the document content - ignoring the XML header - use DocumentDataRoot. For an examples on how to traverse the XMLData tree, see GetNextChild.

AppendChild

See also

Declaration: AppendChild (*pNewData* as XMLData)

Description
AppendChild appends pNewData as last child to the XMLData object.

Errors
1500 The XMLData object is no longer valid.
1505 Invalid XMLData kind was specified.
1506 Invalid address for the return parameter was specified.
1507 Element cannot have Children
1512 Cyclic insertion - new data element is already part of document
1514 Invalid XMLData kind was specified for this position.
1900 Document must not be modified

Example

```
Dim objCurrentParent As XMLData
Dim objNewChild As XMLData

Set objNewChild = objSpyActiveDocument.CreateChild(spyXMLDataElement)
Set objCurrentParent = objSpyActiveDocument.RootElement

objCurrentParent.AppendChild objNewChild

Set objNewChild = Nothing
```

CountChildren

See also

Declaration: CountChildren as long

Description
CountChildren gets the number of children.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.

CountChildrenKind

See also

Declaration: CountChildrenKind (*nKind* as SPYXMLDataKind) as long

Description
CountChildrenKind gets the number of children of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.

EraseAllChildren

See also

Declaration: EraseAllChildren

Description
EraseAllChildren deletes all associated children of the XMLData object.

Errors
1500 The XMLData object is no longer valid.
1900 Document must not be modified

Example
The sample erases all elements of the active document.

```
Dim objCurrentParent As XMLData

Set objCurrentParent = objSpy.ActiveDocument.RootElement
objCurrentParent.EraseAllChildren
```

EraseChild

Method: EraseChild (Child as XMLData)

Description
Deletes the given child node.

Errors
1500 Invalid object.
1506 Invalid input xml
1510 Invalid parameter.

EraseCurrentChild

See also

Declaration: EraseCurrentChild

Description
EraseCurrentChild deletes the current XMLData child object. Before you call EraseCurrentChild

you must initialize an internal iterator with XMLData.GetFirstChild. After deleting the current child, EraseCurrentChild increments the internal iterator of the XMLData element. No error is returned when the last child gets erased and the iterator is moved past the end of the child list. The next call to EraseCurrentChild however, will return error 1503.

Errors

1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object, or the iterator points past the last child.
1900 Document must not be modified

Examples

```
// ----------------------------------------
// XMLSpy scripting environment - JScript
// erase all children of XMLData
// ----------------------------------------
// let's get an XMLData element, we assume that the
// cursor selects the parent of a list in grid view
var objList = Application.ActiveDocument.GridView.CurrentFocus;

// the following line would be shorter, of course
// objList.EraseAllChildren();

// but we want to demonstrate the usage of EraseCurrentChild
if ((objList != null) && (objList.HasChildren))
{
 try
 {
  objEle = objList.GetFirstChild(-1);
  while (objEle != null)
   objList.EraseCurrentChild();
   // no need to call GetNextChild
 }
 catch (err)
  // 1503 - we reached end of child list
  {if ((err.number & 0xffff) != 1503) throw (err);}
}
```

GetChild

See also

Declaration: GetChild(*position* as long) as XMLData

Return Value

Returns an XML element as XMLData object.

Description

GetChild() returns a reference to the child at the given index (zero-based).

Available with TypeLibrary version 1.5

Errors

1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

GetChildAttribute

Method: GetChildAttribute (strName as string) child as XMLData object (NULL on error)

Description
Retrieves the attribute having the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

GetChildElement

Method: GetChildElement(strName as string, nIndex as long) child as XMLData object (NULL on error)

Description
Retrieves the Nth child element with the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

GetChildKind

See also

Declaration: GetChildKind(*position* as long, *nKind* as SPYXMLDataKind) as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetChildKind() returns a reference to a child of this kind at the given index (zero-based). The position parameter is relative to the number of children of the specified kind and not to all children of the object.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

GetCurrentChild

See also

Declaration: GetCurrentChild as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetCurrentChild gets the current child. Before you call GetCurrentChild you must initialize an internal iterator with XMLData.GetFirstChild.

Errors

1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

GetFirstChild

See also

Declaration: GetFirstChild (*nKind* as SPYXMLDataKind) as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetFirstChild initializes a new iterator and returns the first child. Set nKind = -1 to get an iterator for all kinds of children.
REMARK: The iterator is stored inside the XMLData object and gets destroyed when the XMLData object gets destroyed. Be sure to keep a reference to this object as long as you want to use GetCurrentChild, GetNextChild or EraseCurrentChild.

Errors
1500 The XMLData object is no longer valid.
1501 Invalid XMLData kind was specified.
1504 Element has no children of specified kind.
1510 Invalid address for the return parameter was specified.

Example
See the example at XMLData.GetNextChild.

GetNamespacePrefixForURI

Method: GetNamespacePrefixForURI(strURI as string) strNS as string

Description
Returns the namespace prefix of the supplied URI.

Errors
1500 Invalid object.
1510 Invalid parameter.

GetNextChild

See also

Declaration: GetNextChild as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetNextChild steps to the next child of this element. Before you call GetNextChild you must initialize an internal iterator with XMLData.GetFirstChild.

Check for the last child of the element as shown in the sample below.

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

Examples

```
'-----------------------------------------------
'VBA code snippet - iterate XMLData children
'-----------------------------------------------
On Error Resume Next
Set objParent = objSpyActiveDocument.RootElement

'get elements of all kinds
Set objCurrentChild = objParent.GetFirstChild(-1)

Do
 'do something useful with the child

 'step to next child
 Set objCurrentChild = objParent.GetNextChild
Loop Until (Err.Number - vbObjectError = 1503)
```

```
// ----------------------------------------
// XMLSpy scripting environment - JScript
// iterate through children of XMLData
// ----------------------------------------
try
{
   var objXMLData = ... // initialize somehow
   var objChild = objXMLData.GetFirstChild(-1);

   while (true)
   {
      // do something useful with objChild

      objChild = objXMLData.GetNextChild();
   }
}
catch (err)
{
   if ((errnumber & 0xffff) == 1504)
    ;// element has no children
   else if ((errnumber & 0xffff) == 1503)
    ;// last child reached
   else
    throw (err);
}
```

GetTextValueXMLDecoded

Method: GetTextValueXMLDecoded ()as string

Description
Gets the decoded text value of the XML.

Errors
1500 Invalid object.
1510 Invalid parameter.

HasChildren

See also

Declaration: HasChildren as Boolean

Description
The property is true if the object is the parent of other XMLData objects. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

HasChildrenKind

See also

Declaration: HasChildrenKind (*nKind* as SPYXMLDataKind) as Boolean

Description
The method returns true if the object is the parent of other XMLData objects of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

InsertChild

See also

Declaration: InsertChild (*pNewData* as XMLData)

Description
InsertChild inserts the new child before the current child (see also XMLData.GetFirstChild, XMLData.GetNextChild to set the current child).

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1505 Invalid XMLData kind was specified.
1506 Invalid address for the return parameter was specified.
1507 Element cannot have Children
1512 Cyclic insertion - new data element is already part of document
1514 Invalid XMLData kind was specified for this position.
1900 Document must not be modified

InsertChildAfter

Method: InsertChildBefore (Node as XMLData, NewData as XMLData)

Description
Inserts a new XML node (supplied with the second parameter) after the specified node (first parameter).

Errors
1500 Invalid object.
1506 Invalid input xml
1507 No children allowed
1510 Invalid parameter.
1512 Child is already added
1514 Invalid kind at position

InsertChildBefore

Method: InsertChildBefore (Node as XMLData, NewData as XMLData)

Description
Inserts a new XML node (supplied with the second parameter) before the specified node (first parameter).

Errors
1500 Invalid object.
1506 Invalid input xml
1507 No children allowed
1510 Invalid parameter.
1512 Child is already added
1514 Invalid kind at position

IsSameNode

See also

Declaration: IsSameNode (*pNodeToCompare* as XMLData) as Boolean

Description

Returns true if pNodeToCompare references the same node as the object itself.

Errors
1500 The XMLData object is no longer valid.
1506 Invalid address for the return parameter was specified.

Kind

See also

Declaration: Kind as SPYXMLDataKind

Description
Kind of this XMLData object. This property is read-only.

Errors
1500 The XMLData object is no longer valid.

1510 Invalid address for the return parameter was specified.

MayHaveChildren

See also

Declaration: MayHaveChildren as Boolean

Description
Indicates whether it is allowed to add children to this XMLData object.
This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

Name

See also

Declaration: Name as String

Description
Used to modify and to get the name of the XMLData object.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

Parent

See also

Declaration: Parent as XMLData

Return value
Parent as XMLData object. Nothing (or NULL) if there is no parent element.

Description
Parent of this element. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

SetTextValueXMLEncoded

Method: SetTextValueXMLEncoded (*strVal* as String)

Description
Sets the encoded text value of the XML.

Errors
1500 Invalid object.
1513 Modification not allowed.

TextValue

See also

Declaration: TextValue as String

Description
Used to modify and to get the text value of this XMLData object.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

3.3 Interfaces (obsolete)

Interfaces contained in this book are obsolete. It is recommended to migrate your applications to the new interfaces. See the different properties and methods in this book for migration hints.

3.3.1 AuthenticEvent (obsolete)

Superseded by AuthenticView and AuthenticRange

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interface. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Properties

altKey
altLeft
ctrlKey
ctrlLeft
shiftKey
shiftLeft

keyCode
repeat

button

clientX
clientY

dataTransfer

srcElement
fromElement

propertyName

cancelBubble
returnValue

type

Description
`DocEditEvent` interface.

altKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//    if (Application.ActiveDocument.DocEditView.eventaltKey ||
//       Application.ActiveDocument.DocEditView.eventaltLeft)
//      MsgBox ("alt key is down");
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent ieKeyEvent, long inKeyCode, SPYVirtualKeyMask
inVirtualKeyStatus)
{
   if (inVirtualKeyStatus & spyAltKeyMask)
      MsgBox ("alt key is down");
}
```

See also

Declaration: altKey as Boolean

Description
True if the right ALT key is pressed.

altLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyDown ()
// {
//    if (Application.ActiveDocument.DocEditView.eventAltKey ||
//       Application.ActiveDocument.DocEditView.eventAltLeft)
//      MsgBox ("alt key is down");
// }
// use now:
function On_AuthenticView_KeyDown (SPYKeyEvent ieKeyEvent, long inKeyCode, SPYVirtualKeyMask
inVirtualKeyStatus)
{
   if (inVirtualKeyStatus & spyAltKeyMask)
      MsgBox ("alt key is down");
}
```

See also

Declaration: altLeft as Boolean

Description
True if the left ALT key is pressed.

button (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditButtonDown ()
// {
//    if (Application.ActiveDocument.DocEditView.eventbutton == 1)
//       MsgBox ("left mouse button down detected");
// }
// use now:
function On_AuthenticView_MouseEvent (long inXPos, long inYPos, SPYMouseEvent ieMouseEvent,
IAuthenticRange *ipRange)
{
   if (ieMouseEvent & spyLeftButtonDownMask)
      MsgBox ("left mouse button down detected");
}
```

See also

Declaration: button as long

Description
Specifies which mouse button is pressed:

0	No button is pressed.
1	Left button is pressed.
2	Right button is pressed.
3	Left and right buttons are both pressed.
4	Middle button is pressed.
5	Left and middle buttons both are pressed.
6	Right and middle buttons are both pressed.
7	All three buttons are pressed.

cancelBubble (obsolete)

Superseded by the boolean return value of following event handler functions
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

Returning *true* from an event handler function signals that the event has beend handled and normal event handling should be aborted.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//    if (Application.ActiveDocument.DocEditView.eventkeyCode == 0x20)
//    {
//      // cancelkey processing, swallow spaces :-)
//      Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//    }
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent ieKeyEvent, long inKeyCode, SPYVirtualKeyMask
inVirtualKeyStatus)
{
   if (inKeyCode == 0x20)
      return true; // cancelkey processing, swallow spaces :-)
}
```

See also

Declaration: cancelBubble as Boolean

Description

Set cancelBubble to TRUE if the default event handler should not be called.

clientX (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//    MsgBox ("moving over " + Application.ActiveDocument.DocEditView.eventclientX +
//          "/" + Application.ActiveDocument.DocEditView.eventclientY);
// }
// use now:
function On_AuthenticView_MouseEvent (long inXPos, long inYPos, SPYMouseEvent ieMouseEvent,
IAuthenticRange *ipRange)
{
   if (ieMouseEvent & spyMouseMoveMask)
      MsgBox ("moving over " + inXPos + "/" + inYPos);
}
```

See also

Declaration: clientX as long

Description
X value of the current mouse position in client coordinates.

clientY (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//    MsgBox ("moving over " + Application.ActiveDocument.DocEditView.eventclientX +
//          "/" + Application.ActiveDocument.DocEditView.eventclientY);
// }
// use now:
function On_AuthenticView_MouseEvent (long inXPos, long inYPos, SPYMouseEvent ieMouseEvent,
IAuthenticRange *ipRange)
{
   if (ieMouseEvent & spyMouseMoveMask)
      MsgBox ("moving over " + inXPos + "/" + inYPos);
}
```

See also

Declaration: clientY as long

Description
Y value of the current mouse position in client coordinates.

ctrlKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//    if (Application.ActiveDocument.DocEditView.event.ctrKey ||
//       Application.ActiveDocument.DocEditView.event.altLeft)
//      MsgBox ("controlkey is down");
// }
// use now:
function On_AuthenticView_MouseEvent (long inXPos, long inYPos, SPYMouseEvent ieMouseEvent,
IAuthenticRange *ipRange)
{
   if (ieMouseEvent & spyCtrlKeyMask)
      MsgBox ("controlkey is down");
}
```

See also

Declaration: ctrlKey as Boolean

Description
True if the right CTRL key is pressed.

ctrlLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//    if (Application.ActiveDocument.DocEditView.event.ctrKey ||
//       Application.ActiveDocument.DocEditView.event.altLeft)
//      MsgBox ("controlkey is down");
// }
// use now:
function On_AuthenticView_MouseEvent (long inXPos, long inYPos, SPYMouseEvent ieMouseEvent,
IAuthenticRange *ipRange)
{
   if (ieMouseEvent & spyCtrlKeyMask)
      MsgBox ("controlkey is down");
}
```

See also

Declaration: ctrlLeft as Boolean

Description
True if the left CTRL key is pressed.

dataTransfer (obsolete)

Superseded by parameters to
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDrop ()
// {
//    if (Application.ActiveDocument.DocEditView.eventdataTransfer != null)
//       if (!Application.ActiveDocument.DocEditView.eventdataTransfer.ownDrag)
//       {
//         // cancel key processing, don't drop foreign objects :-)
//         Application.ActiveDocument.DocEditView.eventcancelBubble = true;
//       }
// }
// use now:
function On_AuthenticView_BeforeDrop (long inXPos, long inYPos,
                                      IAuthenticRange *ipRange,
                                      IAuthenticDataTransfer *ipData)
{
   if (ipRange != null)
      if (!ipRange.ownDrag)
         return true; // cancel key processing, don't drop foreign objects :-)

   return false;
}
```

See also

Declaration: dataTransfer as Variant

Description
Property dataTransfer.

fromElement (obsolete)

Not supported

See also

Declaration: fromElement as Variant (not supported)

Description
Currently no event sets this property.

keyCode (obsolete)

Superseded by a parameter to AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//    if (Application.ActiveDocument.DocEditView.event.keyCode == 0x20)
//    {
//      // cancel key processing, swallow spaces :-)
//      Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//    }
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent ieKeyEvent, long inKeyCode, SPYVirtualKeyMask
inVirtualKeyStatus)
{
   if (inKeyCode == 0x20)
      return true; // cancel key processing, swallow spaces :-)
}
```

See also

Declaration: keyCode as long

Description
Keycode of the currently pressed key. This property is read-write.

propertyName (obsolete)

Not supported

See also

Declaration: propertyName as String (not supported)

Description
Currently no event sets this property.

repeat (obsolete)

Not supported

See also

Declaration: repeat as Boolean (not supported)

Description
True if the onkeydown event is repeated.

returnValue (obsolete)

No longer supported

See also

Declaration: returnValue as Variant

Description
Use returnValue to set a return value for your event handler.

shiftKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDragOver ()
// {
//    if (Application.ActiveDocument.DocEditView.event.shiftKey ||
//       Application.ActiveDocument.DocEditView.event.shiftLeft)
//      MsgBox ("shift key is down");
// }
// use now:
function On_AuthenticView_DragOver (long inXPos, long inYPos,
                                    SPYMouseEvent ieMouseEvent,
                                    IAuthenticRange *ipRange,
                                    IAuthenticDataTransfer *ipData)
{
   if (ieMouseEvent & spyShiftKeyMask)
      MsgBox ("shift key is down");
}
```

See also

Declaration: shiftKey as Boolean

Description
True if the right SHIFT key is pressed.

shiftLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDragOver ()
// {
//    if (Application.ActiveDocument.DocEditView.event.shiftKey ||
//       Application.ActiveDocument.DocEditView.event.shiftLeft)
//      MsgBox ("shift key is down");
// }
// use now:
function On_AuthenticView_DragOver (long inXPos, long inYPos,
                                    SPYMouseEvent ieMouseEvent,
                                    IAuthenticRange *ipRange,
                                    IAuthenticDataTransfer *ipData)
{
   if (ieMouseEvent & spyShiftKeyMask)
      MsgBox ("shift key is down");
}
```

See also

Declaration: shiftLeft as Boolean

Description
True if the left SHIFT key is pressed.

srcElement (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

With the new event handler function, a range object selecting this element is provided instead of the XMLData element currently below the mouse cursor.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//    var objEvent = Application.ActiveDocument.DocEditView.event;
//    if (objEvent.srcElement != null)
//       MsgBox ("moving over " + objEvent.srcElement.ParentName);
// }
// use now:
function On_AuthenticView_MouseEvent (long inXPos, long inYPos, SPYMouseEvent ieMouseEvent,
IAuthenticRange *ipRange)
{
   if ((ieMouseEvent & spyMouseMoveMask) &&
      (ipRange != null))
      MsgBox ("moving over " + ipRange.FirstXMLData.ParentName);
}
```

See also

Declaration: srcElement as Variant

Description
Element which fires the current event. This is usually an XMLData object.

type (obsolete)

Not supported

See also

Declaration: type as String (not supported)

Description
Currently no event sets this property.

3.3.2 AuthenticSelection (obsolete)

Superseded by AuthenticRange

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interface. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Properties

Start
StartTextPosition
End
EndTextPosition

End (obsolete)

Superseded by AuthenticRange.LastXMLData

```
// ----- javascript sample -----
// instead of:
// var objXMLData = Application.ActiveDocument.DocEditView.CurrentSelection.End;
// use now:
var objXMLData = Application.ActiveDocument.AuthenticView.Selection.LastXMLData;
```

See also

Declaration: End as XMLData

Description
XML element where the current selection ends.

EndTextPosition (obsolete)

Superseded by AuthenticRange.LastXMLDataOffset

```
// ----- javascript sample -----
// instead of:
// var nOffset = Application.ActiveDocument.DocEditView.CurrentSelection.EndTextPosition;
// use now:
var nOffset = Application.ActiveDocument.AuthenticView.Selection.LastXMLDataOffset;
```

See also

Declaration: EndTextPosition as long

Description
Position in DocEditSelection.EndTextValue where the selection ends.

Start (obsolete)

Superseded by AuthenticRange.FirstXMLData

```
// ----- javascript sample -----
// instead of:
// var objXMLData = Application.ActiveDocument.DocEditView.CurrentSelection.Start;
// use now:
var objXMLData = Application.ActiveDocument.AuthenticView.Selection.FirstXMLData;
```

See also

Declaration: Start as XMLData

Description
XML element where the current selection starts.

StartTextPosition (obsolete)

Superseded by AuthenticRange.FirstXMLDataOffset

```
// ----- javascript sample -----
// instead of:
// var nOffset = Application.ActiveDocument.DocEditView.CurrentSelection.StartTextPosition;
// use now:
var nOffset = Application.ActiveDocument.AuthenticView.Selection.FirstXMLDataOffset;
```

See also

Declaration: StartTextPosition as long

Description
Position in DocEditSelection.StartTextValue where the selection starts.

3.3.3 OldAuthentictView (obsolete)

Superseded by AuthenticView and AuthenticRange

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interfaces. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Methods

LoadXML
SaveXML

EditClear
EditCopy
EditCut
EditPaste
EditRedo
EditSelectAll
EditUndo

RowAppend
RowDelete
RowDuplicate
RowInsert
RowMoveDown
RowMoveUp

ApplyTextState
IsTextStateApplied
IsTextStateEnabled

MarkUpView

SelectionSet
SelectionMoveTabOrder

GetNextVisible
GetPreviousVisible

GetAllowedElements

Properties

CurrentSelection

event

XMLRoot

IsEditClearEnabled
IsEditCopyEnabled
IsEditCutEnabled
IsEditPasteEnabled
IsEditRedoEnabled
IsEditUndoEnabled

IsRowAppendEnabled
IsRowDeleteEnabled
IsRowDuplicateEnabled
IsRowInsertEnabled
IsRowMoveDownEnabled
IsRowMoveUpEnabled

Description

Interface for Authentic View.

ApplyTextState (obsolete)

Superseded by AuthenticRange.PerformAction

Use spyAuthenticApply for the eAction parameter. The PerformAction method allows to apply text state attributes to any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.ApplyTextState ("bold");
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.PerformAction (spyAuthenticApply,"bold"))
   MsgBox ("Error: can't set current selection to bold");
```

See also

Declaration: ApplyTextState (*elementName* as String)

Description
Applies or removes the text state defined by the parameter elementName. Common examples for the parameter elementName would be strong and italic.

In an XML document there are segments of data, which may contain sub-elements. For example consider the following HTML:

```
<b>fragment</b>
```

The HTML tag <b> will cause the word fragment to be bold. However, this only happens because the HTML parser knows that the tag <b> is bold. With XML there is much more flexibility. It is possible to define any XML tag to do anything you desire. The point is that it is possible to apply a Text state using XML. But the Text state that is applied must be part of the schema. For example in the OrgChart.xml, OrgChart.sps, OrgChart.xsd example the tag is the same as bold. And to apply bold the method ApplyTextState() is called. But like the row and edit operations it is necessary to test if it is possible to apply the text state.

See also IsTextStateEnabled and IsTextStateApplied.

CurrentSelection (obsolete)

Superseded by AuthenticView.Selection

The returned AuthenticRange object supports navigation via XMLData elements as well as navigation by document elements (e.g. characters, words, tags) or text cursor positions.

```
// ----- javascript sample -----
// instead of:
// var objDocEditSel = Application.ActiveDocument.DocEditView.CurrentSelection;
// use now:
   var objRange = Application.ActiveDocument.AuthenticView.Selection;
```

See also

Declaration: CurrentSelection as DocEditSelection

Description
The property provides access to the current selection in the Authentic View.

EditClear (obsolete)

Superseded by AuthenticRange.Delete

The Delete method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditClear();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.Delete())
   MsgBox ("Error: can't delete current selection");
```

See also

Declaration: EditClear

Description
Deletes the current selection.

EditCopy (obsolete)

Superseded by AuthenticRange.Copy

The Copy method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditCopy();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.Copy())
   MsgBox ("Error: can't copy current selection");
```

See also

Declaration: EditCopy

Description
Copies the current selection to the clipboard.

EditCut (obsolete)

Superseded by AuthenticRange.Cut

The Cut method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditCut();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.Cut())
   MsgBox ("Error: can't cut out current selection");
```

See also

Declaration: EditCut

Description
Cuts the current selection from the document and copies it to the clipboard.

EditPaste (obsolete)

Superseded by AuthenticRange.Paste

The Paste method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditPaste();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.Paste())
   MsgBox ("Error: can't paste to current selection");
```

See also

Declaration: EditPaste

Description
Pastes the content from the clipboard into the document.

EditRedo (obsolete)

Superseded by AuthenticView.Redo

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditRedo();
// use now:
if (!Application.ActiveDocument.AuthenticView.Redo())
   MsgBox ("Error: no redo step available");
```

See also

Declaration: EditRedo

Description
Redo the last undo step.

EditSelectAll (obsolete)

Superseded by AuthenticView.WholeDocument and AuthenticRange.Select

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditSelectAll();
// use now:
Application.ActiveDocument.AuthenticView.WholeDocument.Select();
```

See also

Declaration: EditSelectAll

Description
The method selects the complete document.

EditUndo (obsolete)

Superseded by AuthenticView.Undo

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditUndo();
// use now:
if (!Application.ActiveDocument.AuthenticView.Undo())
   MsgBox ("Error: no undo step available");
```

See also

Declaration: EditUndo

Description
Undo the last action.

event (obsolete)

Superseded by parameters to AuthenticView events.

See also

Declaration: event as DocEditEvent

Description

The event property holds a DocEditEvent object which contains information about the current event.

GetAllowedElements (obsolete)

Superseded by AuthenticRange.CanPerformActionWith

AuthenticRange now supports all functionality of the 'elements' entry helper. Besides querying the elements that can be inserted, appended, etc., you can invoke the action as well. See AuthenticRange.PerformAction for more information.

```
// ----- javascript sample -----
// instead of:
// var anElements = New Array();
// var objDocEditView = Application.ActiveDocument.DocEditView;
// var objStartElement = objDocEditView.CurrentSelection.Start;
// var objEndElement = objDocEditView.CurrentSelection.End;
// objDocEditView.GetAllowedElements(k_ActionInsertBefore,objStartElement,objEndElement,anElements);
// use now:
var anElements = New Array();
Application.ActiveDocument.AuthenticView.Selection.CanPerformActionWith (spyAuthenticInsertBefore,
anElements);
```

See also

Declaration: GetAllowedElements (*nAction* as SpyAuthenticElementActions, *pStartElement* as XMLData, *pEndElement* as XMLData, *pElements* as Variant)

Description

GetAllowedElements() returns the allowed elements for the various actions specified by nAction.

JavaScript example:

```
Function GetAllowed()
{
 var objView = Application.ActiveDocument.DocEditView;

 var anElements = New Array(1);

 var objStart = objView.CurrentSelection.Start;
 var objEnd = objView.CurrentSelection.End;

 var strText;
```

```
 strText = "valid elements at current selection:\n\n";

 For(var i= 1;i<= 4;i++) {
  objPlugIn.GetAllowedElements(i,objStart,objEnd,arrElements);
  strText = strText + ListArray(arrElements) + "------------------\n";
 }

 Return strText;
}

Function ListArray(arrIn)
{
 var strText = "";

 If(TypeOf(arrIn) == "object") {
  For(var i= 0;i<= (arrIn.length - 1);i++)
   strText = strText + arrIn[i] + "\n";
 }

 Return strText;
}
```

VBScript example:

```
Sub DisplayAllowed
 Dim objView
 Set objView = Application.ActiveDocument.DocEditView

 Dim arrElements()

 Dim objStart
 Dim objEnd
 Set objStart = objView.CurrentSelection.Start
 Set objEnd = objView.CurrentSelection.End

 Dim strText
 strText = "valid elements at current selection:" & chr(13) & chr(13)

 Dim i

 For i= 1 To 4
  objView.GetAllowedElements i,objStart,objEnd,arrElements
  strText = strText & ListArray(arrElements) & "----------------" & chr(13)
 Next

 msgbox strText
End Sub

Function ListArray(arrIn)
 Dim strText

 If IsArray(arrIn) Then
  Dim i

  For i= 0 To UBound(arrIn)
   strText = strText & arrIn(i) & chr(13)
  Next
 End If

 ListArray = strText
End Function
```

GetNextVisible (obsolete)

Superseded by AuthenticRange.SelectNext

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the text passage that represents the content of the next XML element is just one of them.

```
// ----- javascript sample -----
// instead of:
// var objCurXMLData = ...
// var objXMLData = Application.ActiveDocument.DocEditView.GetNextVisible(objCurXMLData);
// Application.ActiveDocument.DocEditView.SelectionSet(objXMLData,0,objXMLData,-1);
// use now:
var objRange = ...
try
   {objRange.SelectNext(spyAuthenticTag).Select();}
catch(err)
{
   if((errnumber & 0xffff) == 2003)
      MsgBox("end of document reached");
   else
      throw(err);
}
```

See also

Declaration: GetNextVisible(*pElement* as XMLData) as XMLData

Description
The method gets the next visible XML element in the document.

GetPreviousVisible (obsolete)

Superseded by AuthenticRange.SelectPrevious

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the text passage that represents the content of the previous XML element is just one of them.

```
// ----- javascript sample -----
// instead of:
// var objCurXMLData = ...
// var objXMLData = Application.ActiveDocument.DocEditView.GetPreviousVisible(objCurXMLData);
// Application.ActiveDocument.DocEditView.SelectionSet(objXMLData,0,objXMLData,-1);
// use now:
var objRange = ...
try
   {objRange.SelectPrevious(spyAuthenticTag).Select();}
catch (err)
{
   if ((err.number & 0xffff) == 2004)
      MsgBox("begin of document reached");
   else
      throw (err);
}
```

See also

Declaration: GetPreviousVisible (*pElement* as XMLData) as XMLData

Description
The method gets the previous visible XML element in the document.

IsEditClearEnabled (obsolete)

Superseded by AuthenticRange.IsDeleteEnabled

The IsDeleteEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditClearEnabled)
//   Application.ActiveDocument.DocEditView.EditClear();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsDeleteEnabled)
   objCurrSelection.Delete();
```

See also

Declaration: IsEditClearEnabled as Boolean

Description
True if EditClear is possible. See also Editing operations.

IsEditCopyEnabled (obsolete)

Superseded by AuthenticRange.IsCopyEnabled

The IsCopyEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditCopyEnabled)
//   Application.ActiveDocument.DocEditView.EditCopy();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsCopyEnabled)
   objCurrSelection.Copy();
```

See also

Declaration: IsEditCopyEnabled as Boolean

Description
True if copy to clipboard is possible. See also EditCopy and Editing operations.

IsEditCutEnabled (obsolete)

Superseded by AuthenticRange.IsCutEnabled

The IsCutEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditCutEnabled)
//   Application.ActiveDocument.DocEditView.EditCut();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsCutEnabled)
   objCurrSelection.Cut();
```

See also

Declaration: IsEditCutEnabled as Boolean

Description
True if EditCut is currently possible. See also Editing operations.

IsEditPasteEnabled (obsolete)

Superseded by AuthenticRange.IsPasteEnabled

The IsPasteEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditPasteEnabled)
//   Application.ActiveDocument.DocEditView.EditPaste();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsPasteEnabled)
   objCurrSelection.Paste();
```

See also

Declaration: IsEditPasteEnabled as Boolean

Description
True if EditPaste is possible. See also Editing operations.

IsEditRedoEnabled (obsolete)

Superseded by AuthenticView.IsRedoEnabled

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditRedoEnabled)
//   Application.ActiveDocument.DocEditView.EditRedo();
// use now:
if (Application.ActiveDocument.AuthenticView.IsRedoEnabled)
   Application.ActiveDocument.AuthenticView.Redo();
```

See also

Declaration: IsEditRedoEnabled as Boolean

Description
True if EditRedo is currently possible. See also Editing operations.

IsEditUndoEnabled (obsolete)

Superseded by AuthenticView.IsUndoEnabled

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditUndoEnabled)
//   Application.ActiveDocument.DocEditView.EditUndo();
// use now:
if (Application.ActiveDocument.AuthenticView.IsUndoEnabled)
   Application.ActiveDocument.AuthenticView.Undo();
```

See also

Declaration: IsEditUndoEnabled as Boolean

Description
True if EditUndo is possible. See also Editing operations.

IsRowAppendEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowAppendEnabled)
//   Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
   Application.ActiveDocument.AuthenticView.Selection.AppendRow();
```

See also

Declaration: IsRowAppendEnabled as Boolean

Description
True if RowAppend is possible. See also Row operations.

IsRowDeleteEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowDeleteEnabled)
//   Application.ActiveDocument.DocEditView.Rowdelete();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
   Application.ActiveDocument.AuthenticView.Selection.DeleteRow();
```

See also

Declaration: IsRowDeleteEnabled as Boolean

Description
True if RowDelete is possible. See also Row operations.

IsRowDuplicateEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowDuplicateEnabled)
//   Application.ActiveDocument.DocEditView.RowDuplicate();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
   Application.ActiveDocument.AuthenticView.Selection.DuplicateRow();
```

See also

Declaration: IsRowDuplicateEnabled as Boolean

Description
True if RowDuplicate is currently possible. See also Row operations.

IsRowInsertEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowInsertEnabled)
//   Application.ActiveDocument.DocEditView.RowInsert();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
   Application.ActiveDocument.AuthenticView.Selection.InsertRow();
```

See also

Declaration: IsRowInsertEnabled as Boolean

Description
True if RowInsert is possible. See also Row operations.

IsRowMoveDownEnabled (obsolete)

Superseded by AuthenticRange.IsLastRow

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.OldAuthenticView.IsRowMoveDownEnabled)
//   Application.ActiveDocument.DocEditView.RowMoveDown();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsLastRow)
  Application.ActiveDocument.AuthenticView.Selection.MoveRowDown();
```

See also

Declaration: IsRowMoveDownEnabled as Boolean

Description
True if RowMoveDown is currently possible. See also Row operations.

IsRowMoveUpEnabled (obsolete)

Superseded by AuthenticRange.IsFirstRow

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowMoveUpEnabled)
//   Application.ActiveDocument.DocEditView.RowMoveUp();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsFirstRow)
  Application.ActiveDocument.AuthenticView.Selection.MoveRowUp();
```

See also

Declaration: IsRowMoveUpEnabled as Boolean

Description
True if RowMoveUp is possible. See also Row operations.

IsTextStateApplied (obsolete)

Superseded by AuthenticRange.IsTextStateApplied

```
// ----- javascript sample -----
// instead of:
// ApplicationActiveDocumentDocEditViewIsTextStateApplied ("bold");
// use now:
if (ApplicationActiveDocumentAuthenticViewSelectionIsTextStateApplied ("bold"))
   MsgBox ("bold on");
else
   MsgBox ("bold off");
```

See also

Declaration: IsTextStateApplied (*elementName* as String) as Boolean

Description
Checks to see if the it the text state has already been applied. Common examples for the parameter elementName would be strong and italic.

IsTextStateEnabled (obsolete)

Superseded by AuthenticRange.CanPerformAction

Use spyAuthenticApply for the eAction parameter. The CanPerformAction method allows to operate on any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// ApplicationActiveDocumentDocEditViewIsTextStateEnabled ("bold");
// use now:
if (ApplicationActiveDocumentAuthenticViewSelectionCanPerformAction (spyAuthenticApply,"bold"))
   ...// e.g.enable 'bold'button
```

See also

Declaration: IsTextStateEnabled (*istrElementName* as String) as Boolean

Description
Checks to see if it is possible to apply a text state. Common examples for the parameter elementName would be strong and italic.

LoadXML (obsolete)

Superseded by AuthenticView.AsXMLString

AuthenticView now supports the property AsXMLString that can be used to directly access and replace the document content as an XMLString.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.LoadXML (strDocAsXMLString);
// use now:
try
   {Application.ActiveDocument.AuthenticView.AsXMLString = strDocAsXMLString;}
catch (err)
   {MsgBox ("Error: invalid XML string");}
```

See also

Declaration: LoadXML (*xmlString* as String)

Description
Loads the current XML document with the XML string applied. The new content is displayed immediately.
The *xmlString* parameter must begin with the XML declaration, e.g.,
objPlugIn.LoadXML("<?xml version='1.0' encoding='UTF-8'?><root></root>");

MarkUpView (obsolete)

Superseded by AuthenticView.MarkupVisibility

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.MarkupView = 2;
// use now:
Application.ActiveDocument.AuthenticView.MarkupVisibility = spyAuthenticMarkupLarge;
```

See also

Declaration: MarkUpView (*kind* as long)

Description
By default the document displayed is using HTML techniques. But sometimes it is desirable to show the editing tags. Using this method it is possible to display three different types of markup tags:

0	hide the markup tags
2	show the large markup tags
3	show the mixed markup tags.

RowAppend (obsolete)

Superseded by AuthenticRange.AppendRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.AppendRow())
   MsgBox ("Error: can't append row");
```

See also

Declaration: RowAppend

Description
Appends a row at the current position.

See also Row operations.

RowDelete (obsolete)

Superseded by AuthenticRange.DeleteRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowDelete();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.DeleteRow())
   MsgBox ("Error: can't delete row");
```

See also

Declaration: RowDelete

Description
Deletes the currently selected row(s).

See also Row operations.

RowDuplicate (obsolete)

Superseded by AuthenticRange.DuplicateRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowDuplicate();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.DuplicateRow())
   MsgBox("Error: can't duplicate row");
```

See also

Declaration: RowDuplicate

Description
The method duplicates the currently selected rows.

See also Row operations.

RowInsert (obsolete)

Superseded by AuthenticRange.InsertRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowInsert();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.InsertRow())
   MsgBox("Error: can't insert row");
```

See also

Declaration: RowInsert

Description
Inserts a new row immediately above the current selection.

See also Row operations.

RowMoveDown (obsolete)

Superseded by AuthenticRange.MoveRowDown

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowMoveDown();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.MoveRowDown())
   MsgBox ("Error: can't move row down");
```

See also

Declaration: RowMoveDown

Description
Moves the current row one position down.

See also Row operations.

RowMoveUp (obsolete)

Superseded by AuthenticRange.MoveRowUp

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.MoveRowUp())
   MsgBox ("Error: can't move row up");
```

See also

Declaration: RowMoveUp

Description
Moves the current row one position up.

See also Row operations.

SaveXML (obsolete)

Superseded by AuthenticView.AsXMLString

AuthenticView now supports the property XMLString that can be used to directly access and replace the document content as an XMLString.

```
// ----- javascript sample -----
// instead of:
// var strDocAsXMLString = Application.ActiveDocument.DocEditView.SaveXML();
// use now:
try
{
   var strDocAsXMLString = Application.ActiveDocument.AuthenticView.AsXMLString;
   ... // do something here
}
catch (err)
   {MsgBox ("Error: invalid XML string");}
```

See also

Declaration: SaveXML as String

Return Value
XML structure as string

Description
Saves the current XML data to a string that is returned to the caller.

SelectionMoveTabOrder (obsolete)

Superseded by AuthenticRange.SelectNext

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the next paragraph is just one of them, and navigation is not necessarily bound to the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.SelectionMoveTabOrder(true,true);
// use now:
Application.ActiveDocument.AuthenticView.Selection.SelectNext("spyAuthenticParagraph").Select();
// to append a row to a table use AuthenticRange.AppendRow
```

See also

Declaration: SelectionMoveTabOrder(*bForward* as Boolean, *bTag* as Boolean)

Description

SelectionMoveTabOrder() moves the current selection forwards or backwards.

If bTag is false and the current selection is at the last cell of a table a new line will be added.

SelectionSet (obsolete)

Superseded by AuthenticRange.FirstXMLData and related properties

AuthenticRange supports navigation via XMLData elements as well as navigation by document elements (e.g. characters, words, tags) or text cursor positions.

```
// ----- javascript sample -----
// instead of:
// if (!Application.ActiveDocument.DocEditView.SelectionSet(varXMLData1,0,varXMLData2,-1))
//   MsgBox ("Error: invalid data position");
// use now:
try
{
   var objSelection = Application.ActiveDocument.AuthenticView.Selection;
   objSelection.FirstXMLData = varXMLData1;
   objSelection.FirstXMLdataOffset = 0;
   objSelection.LastXMLData = varXMLData2;
   objSelection.LastXMLDataOffset = -1;
   objSelection.Select();
}
catch (err)
   {MsgBox ("Error: invalid data position");}
// to select all text between varXMLData1 and varXMLdata2, inclusive
```

See also

Declaration: SelectionSet(*pStartElement* as XMLData, *nStartPos* as long, *pEndElement* as XMLData, *nEndPos* as long) as Boolean

Description

Use SelectionSet() to set a new selection in the Authentic View. Its possible to set pEndElement to null (nothing) if the selection should be just over one (pStartElement) XML element.

XMLRoot (obsolete)

Superseded by AuthenticView.XMLDataRoot

```
// ----- javascript sample -----
// instead of:
// var objXMLData = Application.ActiveDocument.DocEditView.XMLRoot;
// use now:
var objXMLData = Application.ActiveDocument.AuthenticView.XMLDataRoot;
```

See also

Declaration: XMLRoot as XMLData

Description

XMLRoot is the parent element of the currently displayed XML structure. Using the XMLData interface you have full access to the complete content of the file.

3.4 Enumerations

This is a list of all enumerations used by the XMLSpy API. If your scripting environment does not support enumerations use the number-values instead.

3.4.1 ENUMApplicationStatus

Description
Enumeration to specify the current Application status.

Possible values:

eApplicationRunning = 0
eApplicationAfterLicenseCheck = 1
eApplicationBeforeLicenseCheck = 2
eApplicationConcurrentLicenseChecKFailed = 3
eApplicationProcessingCommandLine = 4

3.4.2 SPYAttributeTypeDefinition

Description
Attribute type definition that can be selected for generation of Sample XML.
This type is used with the method GenerateDTDOrSchema and GenerateDTDOrSchemaEx.

Possible values:

spyMergedGlobal = 0
spyDistinctGlobal = 1
spyLocal = 2

3.4.3 SPYAuthenticActions

Description
Actions that can be performed on AuthenticRange objects.

Possible values:

spyAuthenticInsertAt = 0
spyAuthenticApply = 1
spyAuthenticClearSurr = 2
spyAuthenticAppend = 3
spyAuthenticInsertBefore = 4
spyAuthenticRemove = 5

3.4.4 SPYAuthenticDocumentPosition

Description
Relative and absolute positions used for navigating with AuthenticRange objects.

Possible values:

spyAuthenticDocumentBegin = 0
spyAuthenticDocument = 1

End
spyAuthenticRangeBeg = 2
in
spyAuthenticRangeEnd = 3

3.4.5 SPYAuthenticElementActions

Description
Actions that can be used with the obsolete object GetAllowedElements (superseded by AuthenticRangeCanPerformActionWith).

Possible values:

k_ActionInsertAt = 0
k_ActionApply = 1
k_ActionClearSurr = 2
k_ActionAppend = 3
k_ActionInsertBefore = 4
k_ActionRemove = 5

3.4.6 SPYAuthenticElementKind

Description
Enumeration of the different kinds of elements used for navigation and selection within the AuthenticRange and AuthenticView objects.

Possible values:

spyAuthenticChar = 0
spyAuthenticWord = 1
spyAuthenticLine = 3
spyAuthenticParagrap = 4
h
spyAuthenticTag = 6
spyAuthenticDocumen = 8
t
spyAuthenticTable = 9
spyAuthenticTableRow = 10
spyAuthenticTableCol = 11
umn

3.4.7 SPYAuthenticMarkupVisibility

Description
Enumeration values to customize the visibility of markup with MarkupVisibility.

Possible values:

spyAuthenticMarkupHid = 0
den
spyAuthenticMarkupSm = 1
all
spyAuthenticMarkupLar = 2
ge
spyAuthenticMarkupMix = 3
ed

3.4.8 SPYAuthenticToolbarButtonState

Description
Authentic toolbar button states are given by the following enumeration:

Possible values:

authenticToolbarButtonDefault	= 0
authenticToolbarButtonEnabled	= 1
authenticToolbarButtonDisabled	= 2

3.4.9 SPYDatabaseKind

Description
Values to select different kinds of databases for import. See DatabaseConnection.DatabaseKind for its use.

Possible values:

spyDB_Access	= 0
spyDB_SQLServer	= 1
spyDB_Oracle	= 2
spyDB_Sybase	= 3
spyDB_MySQL	= 4
spyDB_DB2	= 5
spyDB_Other	= 6
spyDB_Unspecified	= 7
spyDB_PostgreSQL	= 8
spyDB_iSeries	= 9

3.4.10 SPYDialogAction

Description
Values to simulate different interactions on dialogs. See Dialogs for all dialogs available.

Possible values:

spyDialogOK	= 0	// simulate click on OK button
spyDialogCancel	= 1	// simulate click on Cancel button
spyDialogUserInput	= 2	// show dialog and allow user interaction

3.4.11 SPYDOMType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:

spyDOMType_msxml4	= 0	Obsolete
spyDOMType_xerces	= 1	
spyDOMType_xerces3	= 2	
spyDOMType_msxml6	= 3	

spyDOMType_xerces indicates Xerces 2.x usage; spyDOMType_xerces3 indicates Xerces 3.x usage.

3.4.12 SPYDTDSchemaFormat

Description
Enumeration to identify the different schema formats.

Possible values:

spyDTD	= 0
spyW3C	= 1

3.4.13 SPYEncodingByteOrder

Description
Enumeration values to specify encoding byte ordering for text import and export.

Possible values:

spyNONE	= 0
spyLITTLE_ENDIAN	= 1
spyBIG_ENDIAN	= 2

3.4.14 SPYExportNamespace

Description
Enumeration type to configure handling of namespace identifiers during export.

Possible values:

spyNoNamespace	= 0
spyReplaceColonWithUnderscore	= 1

3.4.15 SPYFindInFilesSearchLocation

Description
The different locations where a search can be performed. This type is used with the FindInFilesDlg dialog.

Possible values:

spyFindInFiles Documents	= 0
spyFindInFiles Project	= 1
spyFindInFiles Folder	= 2

3.4.16 SPYFrequentElements

Description
Enumeration value to parameterize schema generation.

Possible values:

spyGlobalElements	= 0
spyGlobalComplexType	= 1

3.4.17 SPYImageKind

Description
Enumeration values to parameterize image type of the generated documentation. These values are used in SchemaDocumentationDialogDiagramFormat and WSDLDocumentationDlgDiagramFormat.

Possible values:

spyImageType_PNG = 0
spyImageType_EMF = 1

3.4.18 SPYImportColumnsType

Description
Enumeration to specify different Import columns types.

Possible values:

spyImportColumns Element = 0
spyImportColumns Attribute = 1

3.4.19 SPYKeyEvent

Description
Enumeration type to identify the different key events. These events correspond with the equally named windows messages.

Possible values:

spyKeyDown = 0
spyKeyUp = 1
spyKeyPressed = 2

3.4.20 SPYKeyStatus

Description
Enumeration type to identify the key status.

Possible values:

spyLeftShiftKeyMask = 1
spyRightShiftKeyMask = 2
spyLeftCtrlKeyMask = 4
spyRightCtrlKeyMask = 8
spyLeftAltKeyMask = 16
spyRightAltKeyMask = 32

3.4.21 SPYLibType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:

spyLibType_static = 0
spyLibType_dll = 1

3.4.22 SPYLoading

Description
Enumeration values to define loading behaviour of URL files.

Possible values:

spyUseCacheProxy = 0
spyReload = 1

3.4.23 SPYMouseEvent

Description
Enumeration type that defines the mouse status during a mouse event. Use the enumeration values as bitmasks rather then directly comparing with them.

Examples

```
'to check for ctrl-leftbutton-down in VB
If (ieMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or XMLSpyLib.spyCtrlKeyDownMask)) Then
      'react on ctrl-leftbutton-down
End If

'to check for double-click with any button in VBScript
If (((ieMouseEvent And spyDoubleClickMask) <> 0) Then
      'react on double-click
End If
```

Possible values:

spyNoButtonMask	= 0
spyMouseMoveMask	= 1
spyLeftButtonMask	= 2
spyMiddleButtonMask	= 4
spyRightButtonMask	= 8
spyButtonUpMask	= 16
spyButtonDownMask	= 32
spyDoubleClickMask	= 64
spyShiftKeyDownMask	= 128
spyCtrlKeyDownMask	= 256
spyLeftButtonDownMask	= 34 // spyLeftButtonMask | spyButtonDownMask
spyMiddleButtonDownMask	= 36 // spyMiddleButtonMask | spyButtonDownMask
spyRightButtonDownMask	= 40 // spyRightButtonMask | spyButtonDownMask
spyLeftButtonUpMask	= 18 // spyLeftButtonMask | spyButtonUpMask
spyMiddleButtonUpMask	= 20 // spyMiddleButtonMask | spyButtonUpMask
spyRightButtonUpMask	= 24 // spyRightButtonMask | spyButtonUpMask
spyLeftDoubleClickMask	= 66 // spyRightButtonMask | spyButtonUpMask
spyMiddleDoubleClickMask	= 68 // spyMiddleButtonMask | spyDoubleClickMask
spyRightDoubleClickMask	= 72 // spyRightButtonMask | spyDoubleClickMask

3.4.24 SPYNumberDateTimeFormat

Description
Enumeration value to configure database connections.

Possible values:

spySystemLocale = 0
spySchemaCompatible = 1

3.4.25 SPYProgrammingLanguage

Description
Enumeration values to select the programming language for code generation from schema definitions.

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Possible values:

spyUndefinedLanguage	= -1
spyJava	= 0
spyCpp	= 1
spyCSharp	= 2

3.4.26 SPYProjectItemTypes

Description
Enumeration values to identify the different elements in project item lists. See SpyProjectItem.ItemType.

Possible values:

spyUnknownItem	= 0
spyFileItem	= 1
spyFolderItem	= 2
spyURLItem	= 3

3.4.27 SPYProjectType

Description
Enumeration values to parameterize generation of C# from schema definitions.

Possible values:

spyVisualStudioProject	= 0	Obsolete
spyVisualStudio2003Project	= 1	Obsolete
spyBorlandProject	= 2	Obsolete
spyMonoMakefile	= 3	Obsolete
spyVisualStudio2005Project	= 4	For C++ code also
spyVisualStudio2008Project	= 5	For C++ code also
spyVisualStudio2010Project	= 6	For C++ code also

3.4.28 SpySampleXMLGenerationChoiceMode

Description
This enumeration is used in GenerateSampleXMLDlg.ChoiceMode:

spySampleXMLGen_FirstBranch = 0
spySampleXMLGen_AllBranches = 1
spySampleXMLGen_ShortestBranch = 2

3.4.29 SPYSampleXMLGenerationOptimization (Obsolete)

This enumeration is OBSOLETE since v2014.

Description
Specify the elements that will be generated in the Sample XML.
This enumeration is used in GenerateSampleXMLDlg.

Possible values:

spySampleXMLGen_Optimized	= 0
spySampleXMLGen_NonMandatoryElements	= 1
spySampleXMLGen_Everything	= 2

3.4.30 SpySampleXMLGenerationSampleValueHints

Description
This enumeration is used in `GenerateSampleXMLDlg.SampleValueHints`

spySampleXMLGen_FirstFit = 0
spySampleXMLGen_RandomFit = 1
spySampleXMLGen_CycleThrough = 2

3.4.31 SPYSampleXMLGenerationSchemaOrDTDAssignment

Description
Specifies what kind of reference to the schema/DTD should be added to the generated Sample XML.
This enumeration is used in GenerateSampleXMLDlg.

Possible values:

spySampleXMLGen_AssignRelatively	= 0
spySampleXMLGen_AssignAbsolutely	= 1
spySampleXMLGen_DoNotAssign	= 2

3.4.32 SPYSchemaDefKind

Description
Enumeration type to select schema diagram types.

Possible values:

spyKindElement	= 0
spyKindComplexType	= 1
spyKindSimpleType	= 2
spyKindGroup	= 3
spyKindModel	= 4
spyKindAny	= 5
spyKindAttr	= 6
spyKindAttrGroup	= 7
spyKindAttrAny	= 8
spyKindIdentityUnique	= 9
spyKindIdentityKey	= 10
spyKindIdentityKeyRef	= 11
spyKindIdentitySelector	= 12
spyKindIdentityField	= 13
spyKindNotation	= 14
spyKindInclude	= 15
spyKindImport	= 16
spyKindRedefine	= 17
spyKindFacet	= 18
spyKindSchema	= 19
spyKindCount	= 20

3.4.33 SPYSchemaDocumentationFormat

Description
Enumeration values to parameterize generation of schema documentation. These values are used in SchemaDocumentationDialogOutputFormat and WSDLDocumentationDlgOutputFormat.

Possible values:

spySchemaDoc_HTML	= 0
spySchemaDoc_MSWord	= 1
spySchemaDoc_RTF	= 2
spySchemaDoc_PDF	= 3

3.4.34 SPYSchemaExtensionType

Description
Enumeration to specify different Schema Extension types.

Possible values:

spySchemaExtension None	= 0

spySchemaExtension SQL XML = 1
spySchemaExtension_MS_SQL_ = 2
Server
spySchemaExtension_Oracle = 3

3.4.35 SPYSchemaFormat

Description
Enumeration to specify different Schema Format types.

Possible values:

spySchemaFormat Hierarchical = 0
spySchemaFormat Flat = 1

3.4.36 SPYTextDelimiters

Description
Enumeration values to specify text delimiters for text export.

Possible values:

spyTabulator = 0
spySemicolon = 1
spyComma = 2
spySpace = 3

3.4.37 SPYTextEnclosing

Description
Enumeration value to specify text enclosing characters for text import and export.

Possible values:

spyNoEnclosing = 0
spySingleQuote = 1
spyDoubleQuote = 2

3.4.38 SPYTypeDetection

Description
Enumeration to select how type detection works during GenerateDTDOrSchema and GenerateDTDOrSchemaEx.

Possible values:

spyBestPossibl = 0
e
spyNumbersOnl= 1
y
spyNoDetection= 2

3.4.39 SPYURLTypes

Description
Enumeration to specify different URL types.

Possible values:

spyURLTypeAu= -1
to

spyURLTypeX = 0
ML
spyURLTypeDT= 1
D

3.4.40 SPYValidateXSDVersion

Description
Enumeration values that select what XSD version to use. The XSD version that is selected depends on both (i) the presence/absence—and, if present, the value—of the `/xs:schema/@vc:minVersion` attribute of the XSD document, and (ii) the value of this enumeration.

`spyValidateXSDVersion_1_0` selects `XSD 1.0` if `vc:minVersion` is absent, or is present with any value.
`spyValidateXSDVersion_1_1` selects `XSD 1.1` if `vc:minVersion` is absent, or is present with any value.
`spyValidateXSDVersion_AutoDetect` selects `XSD 1.1` if `vc:minVersion=1.1`. If the `vc:minVersion` attribute is absent, or is present with a value other than `1.1`, then `XSD 1.0` is selected.

Possible values

```
spyValidateXSDVersion_AutoDet = 0
ect
spyValidateXSDVersion_1_1     = 1
spyValidateXSDVersion_1_0     = 2
```

3.4.41 SPYValidateErrorFormat

Description
Enumeration values that select the format of the error message.

Possible values

```
spyValidateErrorFormat_Text   = 0
spyValidateErrorFormat_ShortX = 1
ML
spyValidateErrorFormat_LongXM = 2
L
```

3.4.42 SPYViewModes

Description
Enumeration values that define the different view modes for XML documents. The mode *spyViewAuthentic(4)* identifies the mode that was intermediately called DocEdit mode and is now called Authentic mode. The mode *spyViewJsonSchema* identifies a mode which is mapped to the Schema Design View on the GUI but is distinguished internally.

Possible values:

```
spyViewGrid     = 0
spyViewText     = 1
spyViewBrowser  = 2
spyViewSchema   = 3
spyViewContent  = 4    // obsolete
```

spyViewAuthentic = 4
spyViewWSDL = 5
spyViewZIP = 6
spyViewEditionInfo = 7
spyViewXBRL = 8
spyViewJsonSchema = 9

3.4.43 SPYVirtualKeyMask

Description
Enumeration type for the most frequently used key masks that identify the status of the virtual keys. Use these values as bitmasks rather then directly comparing with them. When necessary, you can create further masks by using the 'logical or' operator.

Examples

```
' VBScript sample: check if ctrl-key is pressed
If ((inVirtualKeyStatus And spyCtrlKeyMask) <> 0)) Then
      'ctrl-key is pressed
End If

' VBScript sample: check if ONLY ctrl-key is pressed
If (inVirtualKeyStatus == spyCtrlKeyMask) Then
      'exactly ctrl-key is pressed
End If

// JScript sample: check if any of the right virtual keys is pressed
if ((inVirtualKeyStatus & (spyRightShiftKeyMask | spyRightCtrlKeyMask | spyRightAltKeyMask)) != 0)
{
      ;'right virtual key is pressed
}
```

Possible values:

spyNoVirtualKeyMask	= 0	
spyLeftShiftKeyMask	= 1	
spyRightShiftKeyMask	= 2	
spyLeftCtrlKeyMask	= 4	
spyRightCtrlKeyMask	= 8	
spyLeftAltKeyMask	= 16	
spyRightAltKeyMask	= 32	
spyShiftKeyMask	= 3	// spyLeftShiftKeyMask | spyRightShiftKeyMask
spyCtrlKeyMask	= 12	// spyLeftCtrlKeyMask | spyRightCtrlKeyMask
spyAltKeyMask	= 48	// spyLeftAltKeyMask | spyRightAltKeyMask

3.4.44 SPYXMLDataKind

Description
The different types of XMLData elements available for XML documents.

Possible values:

spyXMLDataXMLDocStruct = 0
spyXMLDataXMLEntityDocStruct = 1
spyXMLDataDTDDocStruct = 2

spyXMLDataXML	= 3
spyXMLDataElement	= 4
spyXMLDataAttr	= 5
spyXMLDataText	= 6
spyXMLDataCData	= 7
spyXMLDataComment	= 8
spyXMLDataPI	= 9
spyXMLDataDefDoctype	= 10
spyXMLDataDefExternalID	= 11
spyXMLDataDefElement	= 12
spyXMLDataDefAttlist	= 13
spyXMLDataDefEntity	= 14
spyXMLDataDefNotation	= 15
spyXMLDataKindsCount	= 16

3.5 Application API for Java (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

The Application API in Java has an interface built up of Java classes, each of which corresponds to an object in the Application API. Developers can use these Java classes to interact with the COM API. These classes are listed below and described in subsequent sections. For a description of the Application API objects themselves, see the Application API documentation. Bear in mind that some API features are only available in scripting environments; these have therefore not been ported to Java.

Java classes

SpyApplication
- SpyProject
  - SpyProjectItems
    - SpyProjectItem
  - SpyDocuments
    - SpyDoc
      - SpyAuthenticView
        - SpyAuthenticRange
      - SpyDocEditView
        - SpyDocEditSelection
      - SpyGridView
      - SpyTextView
      - SpyXMLData
  - SpyDialogs
    - SpyCodeGeneratorDlg
    - SpyDTDSchemaGeneratorDlg
    - SpyFileSelectionDlg
    - SpyFindInFilesDlg
    - SpyGenerateSampleXMLDlg
    - SpySchemaDocumentationDlg
    - SpyWSDL20DocumentationDlg
    - SpyWSDLDocumentationDlg
    - SpyXBRLDocumentationDlg
  - SpyDatabaseConnection
  - SpyElementList
  - SpyElementListItem
  - SpyExportSettings
  - SpyFindInFilesResults
    - SpyFindInFilesResult
      - SpyFindInFilesMatch
  - SpyTextImportExportSettings

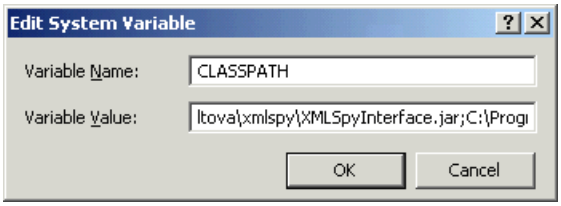
Implementation of COM properties in Java
Properties in Java have been defined to include both a **set** and **get** method (`set` if it is allowed by the COM implementation). For example, the COM class `Document` contains the **GridView** property. In Java the method is called `SpyDoc` and the property is defined as a **GetGridView** method.

If you encounter compiling problems, please check the following points:

- The `xmlspylibdll` must be available in `..\windows\system32`.
- The `XMLSpyInterface.jar` file must be inserted in the ClassPath environment variable.

Setting the `ClassPath` variable in Windows XP

1. Click **Start | Settings | Control panel | System | Advanced | Environment Variables**. This opens the Environment Variables dialog box.
2. If a `ClassPath` entry already exists in the **System variables** group, select the `ClassPath` entry, and click the **Edit** button. Edit the path to: `"C:\Program Files\Altova\xmlspy\XMLSpyInterface.jar"`.



If a `ClassPath` entry does not exist in the System variables group, click the **New** button. The New System Variable dialog pops up. Enter `CLASSPATH` as the variable name, and `"C:\Program Files\Altova\xmlspy\XMLSpyInterface.jar"` as the `ClassPath` variable (alter the path to match your installation, if necessary).

3.5.1 Sample source code (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

The "`SpyDoc doc = app.GetDocuments().OpenFile(...)`" command parameter must be altered to suit your environment.

What the sample does:

- Starts a new XMLSpy instance
- Opens the Datasheet.xml file (alter the path here...)
- Switches to the Enhanced Grid view
- Appends a new child element called "NewChild" with the text value "NewValuE" element to the root element
- Checks if the document is valid and outputs a message to the Java console
- Quits and releases the XMLSpy application

```
import XMLSpyInterface.*;

public class TestSpyInterface
{
```

```
    public TestSpyInterface() {}

    public static void main(String[] args)
    {
        SpyApplication app = null;
        SpyDoc oDoc = null;
        SpyXMLData oData = null;
        SpyXMLData oNewChild = null;

        try
        {
            app = new SpyApplication();
            app.ShowApplication( true );

            oDoc = app.GetDocuments().OpenFile("C:\\FilePath\\OrgChart.xml",true );

            // OrgChart.xml is in the folder C:\Documents and Settings\<username>\My Documents\Altova
            \XMLSpy2017. The filepath should be in
            // the form: C:\\Documents and Settings\\Username\\Folder\\Filename.xml

            if (oDoc != null)
            {
                oDoc.SwitchViewMode(SPYViewModes.spyViewGrid);
                oData = oDoc.GetRootElement();
                oNewChild = oDoc.CreateChild(SPYXMLDataKind.spyXMLDataElement);

                oNewChild.SetName( "NewChild" );
                oNewChild.SetTextValue("newVaLuE");
                oData.AppendChild(oNewChild);

                if (oDoc.IsValid() == false )
                {
                    // is to be expected after above insertion
                    System.out.println( "!!!!!validation error:" + oDoc.GetErrorString() );
                    System.out.println( "!!!!!validation error:" + oDoc.GetErrorPos() );
                    System.out.println( "!!!!!validation error:" + oDoc.GetBadData() );
                }
            }

            app.Quit();
        }
        finally
        {
            // Free any allocated resources by calling ReleaseInstance().
            if (oNewChild != null)
                oNewChild.ReleaseInstance();

            if (oData != null)
                oData.ReleaseInstance();

            if (oDoc != null)
                oDoc.ReleaseInstance();

            if (app != null)
                app.ReleaseInstance();
        }
    }
}
```

If you have difficulties compiling this sample, please try the following commands on the (**Start | Run | cmd**) command line. Please make sure you are currently in the folder that contains the sample java file.

compilation
javac - classpath c:\ *yourpathhere*\ XMLSpyInterface.jar *testspyinterface.java*

Execution

java - classpath c:\ *yourpathhere*\ XMLSpyInterface.jar *testspyinterface*

3.5.2 SpyApplication (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyApplication
{
  public void ReleaseInstance();
  public void ShowApplication(boolean bShow );
  public void Quit();
  public void AddMacroMenuItem (String sMacro, String sDisplayText );
  public void ClearMacroMenu();
  public SpyDoc GetActiveDocument();
  public SpyProject GetCurrentProject();
  public SpyDocuments GetDocuments();
  public SpyElementList GetDatabaseImportElementList(SpyDatabaseConnection oImportSettings );
  public SpyDatabaseConnection GetDatabaseSettings();
  public SpyElementList GetDatabaseTables(SpyDatabaseConnection oImportSettings );
  public SpyExportSettings GetExportSettings();
  public SpyElementList GetTextImportElementList(SpyTextImportExportSettings oImportSettings );
  public SpyTextImportExportSettings GetTextImportExportSettings();
  public SpyDoc ImportFromDatabase(SpyDatabaseConnection oImportSettings, SpyElementList oElementList
  );
  public SpyDoc ImportFromSchema(SpyDatabaseConnection oImportSettings, String strTable, SpyDoc
  oSchemaDoc );
  public SpyDoc ImportFromText(SpyTextImportExportSettings oImportSettings, SpyElementList oElementList
  );
  public SpyDoc ImportFromWord(String sFile );
  public void NewProject(String sPath, boolean bDiscardCurrent );
  public void OpenProject(String sPath , boolean bDiscardCurrent, boolean bDialog );
  public long ShowForm (String sName );
  public void URLDelete(String sURL, String sUser, String sPassword );
  public void URLMakeDirectory(String sURL, String sUser, String sPassword );
  public int GetWarningNumber();
  public String GetWarningText();

  // since Version 2004R4
  public SpyApplication GetApplication();
  public SpyApplication GetParent();
  public SpyDialogs GetDialogs();
  public boolean GetVisible();
  public void SetVisible(boolean ibVisibility );
  public long GetWindowHandle();

  public void ReloadSettings();
  public SpyFindInFilesResults FindInFiles(SpyFindInFilesDlg dlgSettings );
  public boolean ShowFindInFiles(SpyFindInFilesDlg dlgSettings );
  public void Selection(String sVal);

  public long Status();
  public int MajorVersion();
  public int MinorVersion();
  public String Edition();
  public boolean IsAPISupported();
  public long ServicePackVersion();
  public void CreateXMLSchemaFromDBStructure(SpyDatabaseConnection oConnection, SpyElementList
oTables );
}
```

3.5.3 SpyCodeGeneratorDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

```
// since version 2004R4
public class SpyCodeGeneratorDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public SpyDialogs GetParent();
  public long GetProgrammingLanguage();
  public void SetProgrammingLanguage(long ieVal);
  public String GetTemplateFileName();
  public void SetTemplateFileName(String istrVal);
  public String GetOutputPath();
  public void SetOutputPath(String istrVal);
  public long GetOutputPathDialogAction();
  public void SetOutputPathDialogAction(long ieVal);
  public long GetPropertySheetDialogAction();
  public void SetPropertySheetDialogAction(long ieVal);
  public long GetOutputResultDialogAction();
  public void SetOutputResultDialogAction(long ieVal);
  public long GetCPPSettings_DOMType();
  public void SetCPPSettings_DOMType(long ieVal);
  public long GetCPPSettings_LibraryType();
  public void SetCPPSettings_LibraryType(long ieVal);
  public boolean GetCPPSettings_UseMFC();
  public void SetCPPSettings_UseMFC(boolean ibVal);
  public long GetCSharpSettings_ProjectType();
  public void SetCSharpSettings_ProjectType(long ieVal);
}
```

3.5.4 SpyDatabaseConnection (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyDatabaseConnection
{
  public void ReleaseInstance();
  public String GetADOConnection();
  public void SetADOConnection(String sValue );
```

```
  public boolean GetAsAttributes();
  public void SetAsAttributes(boolean bValue );
  public boolean GetCreateMissingTables();
  public void SetCreateMissingTables(boolean bValue );
  public boolean GetCreateNew();
  public void SetCreateNew(boolean bValue );
  public boolean GetExcludeKeys();
  public void SetExcludeKeys(boolean bValue );
  public String GetFile();
  public void SetFile(String sValue );
  public boolean GetIncludeEmptyElements();
  public void SetIncludeEmptyElements(boolean bValue );
  public long GetNumberDateTimeFormat();
  public void SetNumberDateTimeFormat(long nValue );
  public String GetODBCConnection();
  public void SetODBCConnection(String sValue );
  public String GetSQLSelect();
  public void SetSQLSelect(String sValue );
  public long GetTextFieldLen();
  public void SetTextFieldLen(long nValue );

  // since version 2004R4
  public long GetDatabaseKind();
  public void SetDatabaseKind(long nValue );

  // since version 2008R2
  public boolean GetCommentIncluded();
  public void SetCommentIncluded(boolean bValue );
  public String GetNulReplacement();
  public void SetNulReplacement(String sValue );
  public String GetDatabaseSchema();
  public void SetDatabaseSchema(String sValue );

  // since version 2010r3
  public boolean GetPrimaryKeys()
  public void SetPrimaryKeys(boolean bValue )
  public boolean GetForeignKeys()
  public void SetForeignKeys(boolean bValue )
  public boolean GetUniqueKeys()
  public void SetUniqueKeys(boolean bValue )
  public long GetSchemaExtensionType()
  public void SetSchemaExtensionType(long nValue )
  public long GetSchemaFormat()
  public void SetSchemaFormat(long nValue )
  public long GetImportColumnsType()
  public void SetImportColumnsType(long nValue )
}
```

3.5.5 SpyDialogs (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2004R4
public class SpyDialogs
{
  public SpyApplication GetApplication();
  public SpyApplication GetParent();
  public SpyCodeGeneratorDlg GetCodeGeneratorDlg();
  public SpyFileSelectionDlg GetFileSelectionDlg();
  public SpySchemaDocumentationDlg GetSchemaDocumentationDlg();
```

```
  public SpyGenerateSampleXMLDlg GetGenerateSampleXMLDlg();
  public SpyDTDSchemaGeneratorDlg GetDTDSchemaGeneratorDlg();
  public SpyFindInFilesDlg GetFindInFilesDlg();
  public SpyWSDLDocumentationDlg GetWSDLDocumentationDlg();

  // Since version 2010
  public SpyWSDL20DocumentationDlg GetWSDL20DocumentationDlg();
  public SpyXBRLDocumentationDlg GetXBRLDocumentationDlg();
}
```

3.5.6 SpyDoc (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyDoc
{
  public void ReleaseInstance();
  public void SetEncoding(String strEncoding);
  public void SetPathName(String strPath);
  public String GetPathName();
  public String GetTitle();
  public boolean IsModified();
  public void Save();
  public void Close(boolean bDiscardChanges);
  public void UpdateViews();
  public long GetCurrentViewMode();
  public boolean SwitchViewMode(long nMode);
  public SpyGridView GetGridView();
  public void SetActiveDocument();
  public void StartChanges();
  public void EndChanges();
  public void TransformXSL();
  public void AssignDTD(String sDTDFile,boolean bDialog);
  public void AssignSchema(String sSchemaFile,boolean bDialog);
  public void AssignXSL(String sXSLFile,boolean bDialog);
  public void ConvertDTDOrSchema(long nFormat,long nFrequentElements);
  public SpyXMLData CreateChild(long nKind);
  public void CreateSchemaDiagram(long nKind,String sName,String sFile);
  public SpyDocEditView GetDocEditView();
  public void ExportToDatabase(SpyXMLData oFromChild,SpyExportSettings oExportSettings,
  SpyDatabaseConnection oDatabaseConnection);
  public void ExportToText(SpyXMLData oFromChild,SpyExportSettings oExportSettings,
  SpyTextImportExportSettings oTextSettings);
  public void GenerateDTDOrSchema(long nFormat,int nValuesList,long nDetection,long
  nFrequentElements);
  public SpyElementList GetExportElementList(SpyXMLData oFromChild,SpyExportSettings oExportSettings);
  public SpyXMLData GetRootElement();
  public String SaveInString(SpyXMLData oData,boolean bMarked);
  public void SaveToURL(String sUrl,String sUser,String sPassword);
  public String GetErrorString();// See IsValid() or IsWellFormed()
  public int GetErrorPos();// See IsValid() or IsWellFormed()
  public SpyXMLData GetBadData();// See IsValid() or IsWellFormed()
  public boolean IsValid();
  public boolean IsWellFormed(SpyXMLData oData,boolean bWithChildren);

  // Since version 2004R3
  public SpyAuthenticView GetAuthenticView()
```

```
  // Since version 2004R4
  public SpyApplication GetApplication();
  public SpyDocuments GetParent();
  public String GetFullName();
  public void SetFullName(String lstrName);
  public String GetName();
  public String GetPath();
  public boolean GetSaved();
  public void SaveAs(String lstrFileNameOrPath);
  public String GetEncoding();
  public SpyXMLData GetDataRoot();
  public void GenerateProgramCode(SpyCodeGeneratorDlg idlg);
  public void AssignXSLFO(String lstrFile,boolean lbUseDialog);
  public void TransformXSLFO();
  public void GenerateSchemaDocumentation(SpySchemaDocumentationDlg idlg);

  public void ExecuteXQuery(String lstrXMLSourceFile);
  public void SetExternalIsValid(boolean bIsValid);
  public SpyDoc GenerateSampleXML(SpyGenerateSampleXMLDlg ipGenerateXMLDlg);
  public boolean UpdateXMLData();
  public String GetAsXMLString();
  public void SetAsXMLString(String newVal);
  public SpyDoc GenerateDTDOrSchemaEx(SpyDTDSchemaGeneratorDlg ipDTDSchemaGeneratorDlg);
  public SpyDoc ConvertDTDOrSchemaEx(long nFormat,long nFrequentElements,String sOutputPath,
  long nOutputPathDialogAction);
  public SpyTextView GetTextView();
  public String[] GetSuggestions();
  public void SetSuggestions(String[] aList);
  public void SetSelection(String sVal);

  // Since version 2009
  public void GenerateWSDLDocumentation(SpyWSDLDocumentationDlg ipWSDLDocumenationDlg);
  public void TransformXSLEx(long nDialogAction);

  // Since version 2010
  public void GenerateWSDL20Documentation(SpyWSDL20DocumentationDlg ipWSD20DocumenationDlg);
  public void GenerateXBRLDocumentation(SpyXBRLDocumentationDlg ipXBRLDocumentationDlg);
  public SpyDoc ConvertToWSDL20(String sFilePath,boolean bShowDialogs);

  // Since version 2010r3
  public String CreateDBStructureFromXMLSchema(SpyDatabaseConnection oConnection,SpyElementList
  oTables,boolean bDropTableWithExistingName);
  public SpyElementList GetDBStructureList(SpyDatabaseConnection oConnection);
}
```

3.5.7 SpyDocuments (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyDocuments
{
  public void ReleaseInstance();
  public long Count();
  public SpyDoc GetItem(long nNo);
  public SpyDoc NewFile(String strFile,String strType);
  public SpyDoc NewFileFromText(String nSource,String strType);
  public SpyDoc OpenFile(String sPath,boolean bDialog);
  public SpyDoc OpenURL(String sUrl,long nURLType,long nLoading,String sUser,String sPassword);
  public SpyDoc OpenURLDialog(String sURL,long nURLType,long nLoading,String sUser,String sPassword
  );
```

```
  // Since version 2011r2
  public SpyDoc NewAuthenticFile(String strSPSPath,String strXMLPath );
  public SpyDoc OpenAuthenticFile(String strSPSPath,String strXMLPath );
}
```

3.5.8 SpyDTDSchemaGeneratorDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyDTDSchemaGeneratorDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public long GetDTDSchemaFormat();
  public void SetDTDSchemaFormat( long newVal );
  public short GetValueList();
  public void SetValueList( short newVal );
  public long GetTypeDetection();
  public void SetTypeDetection( long newVal );
  public long GetFrequentElements();
  public void SetFrequentElements( long newVal );
  public boolean GetMergeAllEqualNamed();
  public void SetMergeAllEqualNamed( boolean newVal );
  public boolean GetResolveEntities();
  public void SetResolveEntities( boolean newVal );
  public long GetAttributeTypeDefinition();
  public void SetAttributeTypeDefinition( long newVal );
  public boolean GetGlobalAttributes();
  public void SetGlobalAttributes( boolean newVal );
  public boolean GetOnlyStringEnums();
  public void SetOnlyStringEnums( boolean newVal );
  public long GetMaxEnumLength();
  public void SetMaxEnumLength( long newVal );
  public String GetOutputPath();
  public void SetOutputPath( String newVal );
  public long GetOutputPathDialogAction();
  public void SetOutputPathDialogAction( long newVal );
}
```

3.5.9 SpyElementList (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyElementList
{
  public void ReleaseInstance();
  public long GetCount();
  public SpyElementListItem GetItem(long nIndex);
  public void RemoveElement(long nIndex);
}
```

3.5.10 SpyElementListItem (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyElementListItem
{
  public void ReleaseInstance();
  public long GetElementKind();
  public void SetElementKind(long nKind);
  public long GetFieldCount();
  public String GetName();
  public long GetRecordCount();
}
```

3.5.11 SpyExportSettings (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyExportSettings
{
  public void ReleaseInstance();
  public boolean GetCreateKeys();
  public void SetCreateKeys(boolean bValue);
  public SpyElementList GetElementList();
  public void SetElementList(SpyElementList obj);
  public boolean GetEntitiesToText();
  public void SetEntitiesToText(boolean bValue);
  public boolean GetExportAllElements();
  public void SetExportAllElements(boolean bValue);
  public boolean GetFromAttributes();
  public void SetFromAttributes(boolean bValue);
  public boolean GetFromSingleSubElements();
  public void SetFromSingleSubElements(boolean bValue);
  public boolean GetFromTextValues();
  public void SetFromTextValues(boolean bValue);
  public boolean GetIndependentPrimaryKey();
  public void SetIndependentPrimaryKey(boolean bValue);
  public long GetNamespace();
  public void SetNamespace(long nValue);
  public int GetSubLevelLimit();
  public void SetSubLevelLimit(int nValue);
```

```
}
```

3.5.12 SpyFileSelectionDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2004R4
public class SpyFileSelectionDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public SpyDialogs GetParent();
  public String GetFullName();
  public void SetFullName(String lstrName);
  public long GetDialogAction();
  public void SetDialogAction(long leAction);
}
```

3.5.13 SpyFindInFilesDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyFindInFilesDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public String GetFind();
  public void SetFind( String sNewVal );
  public boolean GetRegularExpression();
  public void SetRegularExpression( boolean bNewVal );
  public String GetReplace();
  public void SetReplace( String sNewVal );
  public boolean GetReplaceOnDisk();
  public void SetReplaceOnDisk( boolean bNewVal );
  public boolean GetDoReplace();
  public void SetDoReplace( boolean bNewVal );
  public boolean GetMatchWholeWord();
  public void SetMatchWholeWord( boolean bNewVal );
  public boolean GetMatchCase();
  public void SetMatchCase( boolean bNewVal );
  public long GetSearchLocation();
  public void SetSearchLocation( long nPosition );
  public String GetStartFolder();
  public void SetStartFolder( String sNewVal );
  public boolean GetIncludeSubfolders();
```

```
  public void SetIncludeSubfolders( boolean bNewVal );
  public boolean GetSearchInProjectFilesDoExternal();
  public void SetSearchInProjectFilesDoExternal( boolean bNewVal );
  public String GetFileExtension();
  public void SetFileExtension( String sNewVal );
  public boolean GetAdvancedXMLSearch();
  public void SetAdvancedXMLSearch( boolean bNewVal );
  public boolean GetXMLElementNames();
  public void SetXMLElementNames( boolean bNewVal );
  public boolean GetXMLElementContents();
  public void SetXMLElementContents( boolean bNewVal );
  public boolean GetXMLAttributeNames();
  public void SetXMLAttributeNames( boolean bNewVal );
  public boolean GetXMLAttributeContents();
  public void SetXMLAttributeContents( boolean bNewVal );
  public boolean GetXMLComments();
  public void SetXMLComments( boolean bNewVal );
  public boolean GetXMLCData();
  public void SetXMLCData( boolean bNewVal );
  public boolean GetXMLPI();
  public void SetXMLPI( boolean bNewVal );
  public boolean GetXMLRest();
  public void SetXMLRest( boolean bNewVal );
  public boolean GetShowResult();
  public void SetShowResult( boolean bNewVal );
}
```

3.5.14 SpyFindInFilesMatch (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyFindInFilesMatch
{
  public void ReleaseInstance();
  public long Line();
  public long Position();
  public long Length();
  public String LineText();
  public boolean Replaced();
}
```

3.5.15 SpyFindInFilesResult (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyFindInFilesResult
{
  public void ReleaseInstance();
  public long Count();
  public SpyFindInFilesMatch GetItem( long nNo );
  public String GetPath();
  public SpyDoc GetDocument();
}
```

3.5.16 SpyFindInFilesResults (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyFindInFilesResults
{
  public void ReleaseInstance();
  public long Count();
  public SpyFindInFilesResult GetItem( long nNo );
}
```

3.5.17 SpyGenerateSampleXMLDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyGenerateSampleXMLDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public boolean GetNonMandatoryAttributes();
  public void SetNonMandatoryAttributes( boolean newVal );
  public boolean GetNonMandatoryElements();
  public void SetNonMandatoryElements( boolean newVal );
  public boolean GetTakeFirstChoice();
  public void SetTakeFirstChoice( boolean newVal );
  public long GetRepeatCount();
  public void SetRepeatCount( long newVal );
  public boolean GetFillWithSampleData();
  public void SetFillWithSampleData( boolean newVal );
  public boolean GetFillElementsWithSampleData();
  public void SetFillElementsWithSampleData( boolean newVal );
  public boolean GetFillAttributesWithSampleData();
```

```
  public void SetFillAttributesWithSampleData( boolean newVal );
  public boolean GetContentOfNillableElementsIsNonMandatory();
  public void SetContentOfNillableElementsIsNonMandatory( boolean newVal );
  public boolean GetTryToUseNonAbstractTypes();
  public void SetTryToUseNonAbstractTypes( boolean newVal );
  public long GetOptimization();
  public void SetOptimization( long newVal );
  public long GetSchemaOrDTDAssignment();
  public void SetSchemaOrDTDAssignment( long newVal );
  public String GetLocalNameOfRootElement();
  public void SetLocalNameOfRootElement( String newVal );
  public String GetNamespaceURIOfRootElement();
  public void SetNamespaceURIOfRootElement( String newVal );
  public long GetOptionsDialogAction();
  public void SetOptionsDialogAction( long newVal );
}
```

3.5.18 SpyGridView (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyGridView
{
  public void ReleaseInstance();
  public SpyXMLData GetCurrentFocus();
  public void Deselect(SpyXMLData oData);
  public boolean GetIsVisible();
  public void Select(SpyXMLData oData);
  public void SetFocus(SpyXMLData oData);
}
```

3.5.19 SpyProject (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyProject
{
  public void ReleaseInstance();
  public void CloseProject(boolean bDiscardChanges,boolean bCloseFiles,boolean bDialog);
  public String GetProjectFile();
  public void SetProjectFile(String sFile);
  public SpyProjectItems GetRootItems();
  public void SaveProject();
  public void SaveProjectAs(String sPath,boolean bDialog);
}
```

3.5.20 SpyProjectItem (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyProjectItem
{
  public void ReleaseInstance();
  public SpyProjectItems GetChildItems();
  public String GetFileExtensions();
  public void SetFileExtensions(String sExtensions);
  public long GetItemType();
  public String GetName();
  public SpyDoc Open();
  public SpyProjectItem GetParentItem();
  public String GetPath();
  public String GetValidateWith();
  public void SetValidateWith(String sVal);
  public String GetXMLForXSLTransformation();
  public void SetXMLForXSLTransformation(String sVal);
  public String GetXSLForXMLTransformation();
  public void SetXSLForXMLTransformation(String sVal);
  public String GetXSLTransformationFileExtension();
  public void SetXSLTransformationFileExtension(String sVal);
  public String GetXSLTransformationFolder();
  public void SetXSLTransformationFolder(String sVal);
}
```

3.5.21 SpyProjectItems (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyProjectItems
{
  public void ReleaseInstance();
  public void AddFile(String sPath);
  public void AddFolder(String sName);
  public void AddURL(String sURL,long nURLType,String sUser,String sPassword,boolean bSave);
  public long Count();
  public SpyProjectItem GetItem(long nNumber);
  public void RemoveItem(SpyProjectItem oItemToRemove);
}
```

3.5.22 SpySchemaDocumentationDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete

from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2004R4
public class SpySchemaDocumentationDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public SpyDialogs GetParent();

  public String GetOutputFile();
  public void SetOutputFile(String lstrVal);
  public long GetOutputFormat();
  public void SetOutputFormat(long leVal);

  public boolean GetShowResult();
  public void SetShowResult(boolean lbVal);
  public long GetOptionsDialogAction();
  public void SetOptionsDialogAction(long leVal);
  public long GetOutputFileDialogAction();
  public void SetOutputFileDialogAction(long leVal);
  public boolean GetShowProgressBar();
  public void SetShowProgressBar(boolean lbVal);

  public void IncludeAll(boolean lbInclude );
  public boolean GetIncludeIndex();
  public void SetIncludeIndex(boolean lbVal);
  public boolean GetIncludeGlobalElements();
  public void SetIncludeGlobalElements(boolean lbVal);
  public boolean GetIncludeLocalElements();
  public void SetIncludeLocalElements(boolean lbVal);
  public boolean GetIncludeGroups();
  public void SetIncludeGroups(boolean lbVal);
  public boolean GetIncludeComplexTypes();
  public void SetIncludeComplexTypes(boolean lbVal);
  public boolean GetIncludeSimpleTypes();
  public void SetIncludeSimpleTypes(boolean lbVal);
  public boolean GetIncludeAttributeGroups();
  public void SetIncludeAttributeGroups(boolean lbVal);
  public boolean GetIncludeRedefines();
  public void SetIncludeRedefines(boolean lbVal);

  public void AllDetails(boolean lbDetailsOn );
  public boolean GetShowDiagram();
  public void SetShowDiagram(boolean lbVal);
  public boolean GetShowNamespace();
  public void SetShowNamespace(boolean lbVal);
  public boolean GetShowType();
  public void SetShowType(boolean lbVal);
  public boolean GetShowChildren();
  public void SetShowChildren(boolean lbVal);
  public boolean GetShowUsedBy();
  public void SetShowUsedBy(boolean lbVal);
  public boolean GetShowProperties();
  public void SetShowProperties(boolean lbVal);
  public boolean GetShowSingleFacets();
  public void SetShowSingleFacets(boolean lbVal);
```

```
public boolean GetShowPatterns();
public void SetShowPatterns(boolean ibVal);
public boolean GetShowEnumerations();
public void SetShowEnumerations(boolean ibVal);
public boolean GetShowAttributes();
public void SetShowAttributes(boolean ibVal);
public boolean GetShowIdentityConstraints();
public void SetShowIdentityConstraints(boolean ibVal);
public boolean GetShowAnnotations();
public void SetShowAnnotations(boolean ibVal);
public boolean GetShowSourceCode();
public void SetShowSourceCode(boolean ibVal);

// Since version 2009
public boolean GetEmbedDiagrams();
public void SetEmbedDiagrams(boolean ibVal);
public long GetDiagramFormat();
public void SetDiagramFormat(long inVal);
public boolean GetIncludeGlobalAttributes();
public void SetIncludeGlobalAttributes(boolean ibVal);
public boolean GetIncludeLocalAttributes();
public void SetIncludeLocalAttributes(boolean ibVal);
public boolean GetIncludeReferencedSchemas();
public void SetIncludeReferencedSchemas(boolean ibVal);
public boolean GetMultipleOutputFiles();
public void SetMultipleOutputFiles(boolean ibVal);

// Since version 2010
public boolean GetEmbedCSSInHTML();
public void SetEmbedCSSInHTML(boolean ibVal);
public boolean GetCreateDiagramsFolder();
public void SetCreateDiagramsFolder(boolean ibVal);

// Since version 2010r3
public boolean GetGenerateRelativeLinks();
public void SetGenerateRelativeLinks(boolean ibVal);

// Since version 2011r2
public boolean GetUseFixedDesign();
public void SetUseFixedDesign(boolean ibVal);
public String GetSPSFile();
public void SetSPSFile(String istrVal);
}
```

3.5.23 SpyTextImportExportSettings (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyTextImportExportSettings
{
  public void ReleaseInstance();
  public String GetDestinationFolder();
  public void SetDestinationFolder(String sVal);
```

```
  public long GetEnclosingCharacter();
  public void SetEnclosingCharacter(long nEnclosing);
  public String GetEncoding();
  public void SetEncoding(String sVal);
  public long GetEncodingByteOrder();
  public void SetEncodingByteOrder(long nByteOrder);
  public long GetFieldDelimiter();
  public void SetFieldDelimiter(long nDelimiter);
  public String GetFileExtension();
  public void SetFileExtension(String sVal);
  public boolean GetHeaderRow();
  public void SetHeaderRow(boolean bVal);
  public String GetImportFile();
  public void SetImportFile(String sVal);
}
```

3.5.24 SpyTextView (obsolete)

```
public class SpyTextView
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public SpyDoc GetParent();
  public long LineFromPosition( long nCharPos );
  public long PositionFromLine( long nLine );
  public long LineLength( long nLine );
  public String GetSelText();
  public void SetSelText( String sText );
  public String GetRangeText( long nPosFrom, long nPosTill );
  public void ReplaceText( long nPosFrom, long nPosTill, String sText );
  public void MoveCaret( long nDiff );
  public void GoToLineChar( long nLine, long nChar );
  public void SelectText( long nPosFrom, long nPosTill );
  public long GetSelectionStart();
  public void SetSelectionStart( long nNewVal );
  public long GetSelectionEnd();
  public void SetSelectionEnd( long nNewVal );
  public String GetText();
  public void SetText( String sText );
  public long LineCount();
  public long Length();
}
```

3.5.25 SpyWSDL20DocumentationDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2010
public class SpyWSDL20DocumentationDlg
{
      public void ReleaseInstance();
      public SpyApplication GetApplication();
```

```
public long GetOptionsDialogAction();
public void SetOptionsDialogAction( long nNewVal );

public long GetOutputFileDialogAction();
public void SetOutputFileDialogAction( long nNewVal );

public boolean GetShowProgressBar();
public void SetShowProgressBar( boolean bNewVal );

public String GetOutputFile();
public void SetOutputFile( String sNewVal );

public long GetOutputFormat();
public void SetOutputFormat( long nNewVal );

public boolean GetMultipleOutputFiles();
public void SetMultipleOutputFiles( boolean bNewVal );

  public boolean GetEmbedCSSInHTML();
  public void SetEmbedCSSInHTML(boolean bNewVal);

public long GetDiagramFormat();
public void SetDiagramFormat( long nNewVal );

public boolean GetEmbedDiagrams();
public void SetEmbedDiagrams( boolean bNewVal );

  public boolean GetCreateDiagramsFolder();
  public void SetCreateDiagramsFolder(boolean bNewVal);

public boolean GetShowResult();
public void SetShowResult( boolean bNewVal );

public void IncludeAll( boolean bNewVal );
public void AllDetails( boolean bNewVal );

public boolean GetIncludeOverview();
public void SetIncludeOverview( boolean bNewVal );

public boolean GetIncludeService();
public void SetIncludeService( boolean bNewVal );

public boolean GetIncludeBinding();
public void SetIncludeBinding( boolean bNewVal );

public boolean GetIncludeInterface();
public void SetIncludeInterface( boolean bNewVal );

public boolean GetIncludeTypes();
public void SetIncludeTypes( boolean bNewVal );

public boolean GetIncludeImportedWSDLFiles();
public void SetIncludeImportedWSDLFiles( boolean bNewVal );
```

```
        public boolean GetShowServiceDiagram();
        public void SetShowServiceDiagram( boolean bNewVal );

        public boolean GetShowBindingDiagram();
        public void SetShowBindingDiagram( boolean bNewVal );

        public boolean GetShowInterfaceDiagram();
        public void SetShowInterfaceDiagram( boolean bNewVal );

        public boolean GetShowTypesDiagram();
        public void SetShowTypesDiagram( boolean bNewVal );

        public boolean GetShowEndpoint();
        public void SetShowEndpoint( boolean bNewVal );

        public boolean GetShowSourceCode();
        public void SetShowSourceCode( boolean bNewVal );

        public boolean GetShowExtensibility();
        public void SetShowExtensibility( boolean bNewVal );

        public boolean GetShowUsedBy();
        public void SetShowUsedBy( boolean bNewVal );

        public boolean GetShowOperation();
        public void SetShowOperation( boolean bNewVal );

        public boolean GetShowFault();
        public void SetShowFault( boolean bNewVal );

        // Since version 2011r2
        public boolean GetUseFixedDesign();
        public void SetUseFixedDesign(boolean ibVal);

        public String GetSPSFile();
        public void SetSPSFile(String istrVal);

}
```

3.5.26 SpyWSDLDocumentationDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2008r2sp1
public class SpyWSDLDocumentationDlg
{
        public void ReleaseInstance();
        public SpyApplication GetApplication();
```

```
public String GetOutputFile();
public void SetOutputFile( String sNewVal );

public long GetOutputFileDialogAction();
public void SetOutputFileDialogAction( long nNewVal );

public long GetOptionsDialogAction();
public void SetOptionsDialogAction( long nNewVal );

public boolean GetShowProgressBar();
public void SetShowProgressBar( boolean bNewVal );

public boolean GetShowResult();
public void SetShowResult( boolean bNewVal );

public long GetOutputFormat();
public void SetOutputFormat( long nNewVal );

public boolean GetEmbedDiagrams();
public void SetEmbedDiagrams( boolean bNewVal );

public long GetDiagramFormat();
public void SetDiagramFormat( long nNewVal );

public boolean GetMultipleOutputFiles();
public void SetMultipleOutputFiles( boolean bNewVal );

public void IncludeAll( boolean bNewVal );

public boolean GetIncludeBinding();
public void SetIncludeBinding( boolean bNewVal );

public boolean GetIncludeImportedWSDLFiles();
public void SetIncludeImportedWSDLFiles( boolean bNewVal );

public boolean GetIncludeMessages();
public void SetIncludeMessages( boolean bNewVal );

public boolean GetIncludeOverview();
public void SetIncludeOverview( boolean bNewVal );

public boolean GetIncludePortType();
public void SetIncludePortType( boolean bNewVal );

public boolean GetIncludeService();
public void SetIncludeService( boolean bNewVal );

public boolean GetIncludeTypes();
public void SetIncludeTypes( boolean bNewVal );

public void AllDetails( boolean bNewVal );

public boolean GetShowBindingDiagram();
```

```
    public void SetShowBindingDiagram( boolean bNewVal );

    public boolean GetShowExtensibility();
    public void SetShowExtensibility( boolean bNewVal );

    public boolean GetShowMessageParts();
    public void SetShowMessageParts( boolean bNewVal );

    public boolean GetShowPort();
    public void SetShowPort( boolean bNewVal );

    public boolean GetShowPortTypeDiagram();
    public void SetShowPortTypeDiagram( boolean bNewVal );

    public boolean GetShowPortTypeOperations();
    public void SetShowPortTypeOperations( boolean bNewVal );

    public boolean GetShowServiceDiagram();
    public void SetShowServiceDiagram( boolean bNewVal );

    public boolean GetShowSourceCode();
    public void SetShowSourceCode( boolean bNewVal );

    public boolean GetShowTypesDiagram();
    public void SetShowTypesDiagram( boolean bNewVal );

    public boolean GetShowUsedBy();
    public void SetShowUsedBy( boolean bNewVal );

      // Since version 2010
      public boolean GetEmbedCSSInHTML();
      public void SetEmbedCSSInHTML(boolean ibVal);

      public boolean GetCreateDiagramsFolder();
      public void SetCreateDiagramsFolder(boolean ibVal);

    // Since version 2011r2
    public boolean GetUseFixedDesign();
    public void SetUseFixedDesign(boolean ibVal);

    public String GetSPSFile();
    public void SetSPSFile(String istrVal);

}
```

3.5.27 SpyXBRLDocumentationDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2010
public class SpyXBRLDocumentationDlg
{
      public void ReleaseInstance();
      public SpyApplication GetApplication();

      public long GetOptionsDialogAction();
      public void SetOptionsDialogAction( long nNewVal );

      public long GetOutputDialogAction();
      public void SetOutputDialogAction( long nNewVal );

      public boolean GetShowProgressBar();
      public void SetShowProgressBar( boolean bNewVal );

      public String GetOutputFile();
      public void SetOutputFile( String sNewVal );

      public long GetOutputFormat();
      public void SetOutputFormat( long nNewVal );

        public boolean GetEmbedCSSInHTML();
        public void SetEmbedCSSInHTML(boolean bNewVal);

      public long GetDiagramFormat();
      public void SetDiagramFormat( long nNewVal );

      public boolean GetEmbedDiagrams();
      public void SetEmbedDiagrams( boolean bNewVal );

        public boolean GetCreateDiagramsFolder();
        public void SetCreateDiagramsFolder(boolean bNewVal);

      public boolean GetShowResult();
      public void SetShowResult( boolean bNewVal );

      public void IncludeAll( boolean bNewVal );
      public void AllDetails( boolean bNewVal );

      public boolean GetIncludeOverview();
      public void SetIncludeOverview( boolean bNewVal );

      public boolean GetIncludeNamespacePrefixes();
      public void SetIncludeNamespacePrefixes( boolean bNewVal );

      public boolean GetIncludeGlobalElements();
      public void SetIncludeGlobalElements( boolean bNewVal );

      public boolean GetIncludeDefinitionLinkroles();
      public void SetIncludeDefinitionLinkroles( boolean bNewVal );

      public boolean GetIncludePresentationLinkroles();
      public void SetIncludePresentationLinkroles( boolean bNewVal );
```

```
        public boolean GetIncludeCalculationLinkroles();
        public void SetIncludeCalculationLinkroles( boolean bNewVal );

        public boolean GetShowDiagram();
        public void SetShowDiagram( boolean bNewVal );

        public boolean GetShowSubstitutiongroup();
        public void SetShowSubstitutiongroup( boolean bNewVal );

        public boolean GetShowItemtype();
        public void SetShowItemtype( boolean bNewVal );

        public boolean GetShowBalance();
        public void SetShowBalance( boolean bNewVal );

        public boolean GetShowPeriod();
        public void SetShowPeriod( boolean bNewVal );

        public boolean GetShowAbstract();
        public void SetShowAbstract( boolean bNewVal );

        public boolean GetShowNillable();
        public void SetShowNillable( boolean bNewVal );

        public boolean GetShowLabels();
        public void SetShowLabels( boolean bNewVal );

        public boolean GetShowReferences();
        public void SetShowReferences( boolean bNewVal );

        public boolean GetShowLinkbaseReferences();
        public void SetShowLinkbaseReferences( boolean bNewVal );

        public boolean GetShortQualifiedName();
        public void SetShortQualifiedName( boolean bNewVal );

        public boolean GetShowImportedElements();
        public void SetShowImportedElements( boolean bNewVal );

        // Since version 2011r2
        public boolean GetUseFixedDesign();
        public void SetUseFixedDesign(boolean ibVal);

        public String GetSPSFile();
        public void SetSPSFile(String istrVal);
};
```

3.5.28 SpyXMLData (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see

the section: Programming Languages | Java.

```
public class SpyXMLData
{
  public void ReleaseInstance();
  public void AppendChild(SpyXMLData oNewData);
  public void EraseAllChildren();
  public void EraseCurrentChild();
  public SpyXMLData GetCurrentChild();
  public SpyXMLData GetFirstChild(long nKind);
  public SpyXMLData GetNextChild();
  public boolean GetHasChildren();
  public void InsertChild(SpyXMLData oNewData);
  public boolean IsSameNode(SpyXMLData oToComp);
  public long GetKind();
  public boolean GetMayHaveChildren();
  public String GetName();
  public void SetName(String sValue);
  public SpyXMLData GetParent();
  public String GetTextValue();
  public void SetTextValue(String sValue);
}
```

3.5.29 Authentic (obsolete)

SpyAuthenticRange (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2004R3
public class SpyAuthenticRange
{
   public void ReleaseInstance();
   public SpyApplication GetApplication();
   public SpyAuthenticView GetParent();
   public SpyAuthenticRange GotoNext(long eKind);
   public SpyAuthenticRange GotoPrevious(long eKind);
   public void Select();
   public long GetFirstTextPosition();
   public void SetFirstTextPosition(long nTextPosition);
   public long GetLastTextPosition();
   public void SetLastTextPosition(long nTextPosition);
   public String GetText();
   public void SetText(String strText);
   public boolean PerformAction(long eAction, String strElementName);
   public boolean CanPerformAction(long eAction, String strElementName);
   public String[] CanPerformActionWith(long eAction);
   public SpyAuthenticRange GoTo(long eKind, long nCount, long nFrom);
   public SpyAuthenticRange SelectNext(long eKind);
   public SpyAuthenticRange SelectPrevious(long eKind);
   public SpyAuthenticRange MoveBegin(long eKind, long nCount);
   public SpyAuthenticRange MoveEnd(long eKind, long nCount);
   public SpyAuthenticRange ExpandTo(long eKind);
   public SpyAuthenticRange CollapsToBegin();
   public SpyAuthenticRange CollapsToEnd();
   public SpyAuthenticRange GotoNextCursorPosition();
```

```
   public SpyAuthenticRange GotoPreviousCursorPosition();
   public boolean IsEmpty();
   public boolean IsEqual(SpyAuthenticRange ipCmp);
   public SpyAuthenticRange Clone();
   public SpyAuthenticRange SetFromRange(SpyAuthenticRange ipSrc);
   public boolean Delete();
   public boolean Cut();
   public boolean Copy();
   public boolean Paste();
   public SpyXMLData GetFirstXMLData();
   public void SetFirstXMLData(SpyXMLData objXMLDataPtr);
   public long GetFirstXMLDataOffset();
   public void SetFirstXMLDataOffset(long nOffset);
   public SpyXMLData GetLastXMLData();
   public void SetLastXMLData(SpyXMLData objXMLDataPtr);
   public long GetLastXMLDataOffset();
   public void SetLastXMLDataOffset(long nOffset);
   public String[] GetElementHierarchy();
   public String[] GetElementAttributeNames(String strElementName);
   public boolean HasElementAttribute(String strElementName,String strAttributeName);
   public String GetElementAttributeValue(String strElementName,String strAttributeName);
   public void SetElementAttributeValue(String strElementName,String strAttributeName,String
strNewValue);
   public String[] GetEntityNames();
   public void InsertEntity(String strEntityName);
   public boolean IsInDynamicTable();
   public boolean AppendRow();
   public boolean InsertRow();
   public boolean DuplicateRow();
   public boolean DeleteRow();
   public boolean MoveRowUp();
   public boolean MoveRowDown();

   // Since version 2004R4
   public boolean IsCopyEnabled();
   public boolean IsCutEnabled();
   public boolean IsPasteEnabled();
   public boolean IsDeleteEnabled();
   public boolean IsTextStateApplied(String istrElementName);
   public boolean IsFirstRow();
   public boolean IsLastRow();
}
```

SpyAuthenticView (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2004R3
public class SpyAuthenticView
{
   public void ReleaseInstance();
   public SpyApplication GetApplication();
   public SpyDoc GetParent();
   public SpyAuthenticRange GetSelection();
   public void SetSelection(SpyAuthenticRange obj);
   public SpyAuthenticRange GetDocumentBegin();
   public SpyAuthenticRange GetDocumentEnd();
   public SpyAuthenticRange GetWholeDocument();
   public long GetMarkupVisibility();
   public void SetMarkupVisibility(long eSpyAuthenticMarkupVisibility);
```

```
    public SpyAuthenticRange GoTo(long eKind,long nCount,long nFrom );
    public void Print(boolean bWithPreview,boolean bPromptUser);
    public boolean Undo();
    public boolean Redo();
    public void UpdateXMLInstanceEntities();

     // Since version 2004R4
     public String GetAsXMLString();
     public void SetAsXMLString(String istrXML );
     public SpyXMLData GetXMLDataRoot();
     public boolean IsUndoEnabled();
     public boolean IsRedoEnabled();
}
```

SpyDocEditSelection (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyDocEditSelection
{
  public void ReleaseInstance();
  public SpyXMLData GetEnd();
  public long GetEndTextPosition();
  public SpyXMLData GetStart();
  public long GetStartTextPosition();
}
```

SpyDocEditView (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyDocEditView
{
  public void ReleaseInstance();
  public void ApplyTextState(String sElementName );
  public SpyDocEditSelection GetCurrentSelection();
  public void EditClear();
  public void EditCopy();
  public void EditCut();
  public void EditPaste();
  public void EditRedo();
  public void EditSelectAll();
  public void EditUndo();
  public SpyXMLData GetNextVisible(SpyXMLData oElement);
  public SpyXMLData GetPreviousVisible(SpyXMLData oElement);
  public boolean GetIsEditClearEnabled();
  public boolean GetIsEditCopyEnabled();
  public boolean GetIsEditCutEnabled();
  public boolean GetIsEditPasteEnabled();
```

```
  public boolean GetIsEditRedoEnabled();
  public boolean GetIsEditUndoEnabled();
  public boolean GetIsRowAppendEnabled();
  public boolean GetIsRowDeleteEnabled();
  public boolean GetIsRowDuplicateEnabled();
  public boolean GetIsRowInsertEnabled();
  public boolean GetIsRowMoveDownEnabled();
  public boolean GetIsRowMoveUpEnabled();
  public boolean IsTextStateApplied(String sElementName );
  public boolean IsTextStateEnabled(String sElementName );
  public void LoadXML(String sXML );
  public void MarkUpView (long nKind );
  public void RowAppend();
  public void RowDelete();
  public void RowDuplicate();
  public void RowInsert();
  public void RowMoveDown();
  public void RowMoveUp();
  public String SaveXML();
  public void SelectionMoveTabOrder(boolean bForward,boolean bTag );
  public boolean SelectionSet(SpyXMLData oStart,long nStartPos,SpyXMLData oEndElement,long
  nEndPos );
  public SpyXMLData GetXMLRoot();
  public String[] GetAllowedElements(long nAction,SpyXMLData oStartPtr,SpyXMLData oEndPtr);
}
```

3.5.30 Predefined constants (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

This section lists all classes that define the predefined constants used by the Java interface.

SPYApplicationStatus (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYApplicationStatus
  {
     public final static long spyApplicationStatus_Running                    = 0;
     public final static long spyApplicationStatus_AfterLicenseCheck          = 1;
     public final static long spyApplicationStatus_BeforeLicenseCheck         = 2;
     public final static long spyApplicationStatus_ConcurrentLicenseCheckFailed = 3;
     public final static long spyApplicationStatus_ProcessingCommandLine      = 4;
  }
```

SPYAttributeTypeDefinition (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYAttributeTypeDefinition
{
   public final static long spyMergedGlobal    = 0;
   public final static long spyDistinctGlobal  = 1;
   public final static long spyLocal           = 2;
}
```

SPYAuthenticActions (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYAuthenticActions
{
    public final static long spyAuthenticInsertAt       = 0;
    public final static long spyAuthenticApply          = 1;
    public final static long spyAuthenticClearSurr      = 2;
    public final static long spyAuthenticAppend         = 3;
    public final static long spyAuthenticInsertBefore   = 4;
    public final static long spyAuthenticRemove         = 5;
}
```

SPYAuthenticDocumentPosition (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYAuthenticDocumentPosition
{
    public final static long spyAuthenticDocumentBegin= 0;
```

```
    public final static long spyAuthenticDocumentEnd  = 1;
    public final static long spyAuthenticRangeBegin   = 2;
    public final static long spyAuthenticRangeEnd     = 3;
}
```

SPYAuthenticElementKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYAuthenticElementKind
{
    public final static long spyAuthenticChar     = 0;
    public final static long spyAuthenticWord     = 1;
    public final static long spyAuthenticLine     = 3;
    public final static long                      = 4;
    spyAuthenticParagraph
    public final static long spyAuthenticTag      = 6;
    public final static long                      = 8;
    spyAuthenticDocument
    public final static long spyAuthenticTable    = 9;
    public final static long                      = 10;
    spyAuthenticTableRow
    public final static long                      = 11;
    spyAuthenticTableColumn
}
```

SPYAuthenticMarkupVisibility (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYAuthenticMarkupVisibility
{
    public final static long spyAuthenticMarkupHidden   = 0;
    public final static long spyAuthenticMarkupSmall    = 1;
    public final static long spyAuthenticMarkupLarge    = 2;
    public final static long spyAuthenticMarkupMixed    = 3;
}
```

SPYDatabaseKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYLoading
  {
     public final static long spyDB_Access        = 0;
     public final static long spyDB_SQLServer     = 1;
     public final static long spyDB_Oracle        = 2;
     public final static long spyDB_Sybase        = 3;
     public final static long spyDB_MySQL         = 4;
     public final static long spyDB_DB2           = 5;
     public final static long spyDB_Other         = 6;
     public final static long spyDB_Unspecified   = 7;
  }
```

SPYDialogAction (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYDialogAction
  {
     public final static long spyDialogOK         = 0;
     public final static long spyDialogCancel     = 1;
     public final static long spyDialogUserInput  = 2;
  }
```

SPYDOMType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYDOMType
  {
     public final static long spyDOMType_msxml4  = 0;
     public final static long spyDOMType_xerces  = 1;
```

```
}
```

SPYDTDSchemaFormat (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYDTDSchemaFormat
{
    public final static long spyDTD       = 0;
    public final static long spyDCD       = 1;
    public final static long spyXMLData   = 2;
    public final static long spyBizTalk   = 3;
    public final static long spyW3C       = 4;
}
```

SPYEncodingByteOrder (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYEncodingByteOrder
{
    public final static long spyNONE          = 0;
    public final static long spyLITTLE_ENDIAN = 1;
    public final static long spyBIG_ENDIAN    = 2;
}
```

SPYExportNamespace (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYExportNamespace
{
    public final static long spyNoNamespace                = 0;
```

```
    public final static long spyReplaceColonWithUnderscore  = 1;
}
```

SPYFindInFilesSearchLocation (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYFindInFilesSearchLocation
{
   public final static long spyFindInFiles_Documents = 0;
   public final static long spyFindInFiles_Project   = 1;
   public final static long spyFindInFiles_Folder    = 2;
}
```

SPYFrequentElements (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYFrequentElements
{
    public final static long spyGlobalElements  = 0;
    public final static long                    = 1;
    spyGlobalComplexType
}
```

SPYImageKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYImageKind
{
    public final static long spyImageType_PNG   = 0;
    public final static long spyImageType_EMF   = 1;
}
```

SPYImportColumnsType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

SPYLibType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYLibType
  {
     public final static long spyLibType_static     = 0;
     public final static long spyLibType_dll        = 1;
  }
```

SPYLoading (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYLoading
{
    public final static long spyUseCacheProxy  = 0;
    public final static long spyReload         = 1;
}
```

SPYNumberDateTimeFormat (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see

the section: Programming Languages | Java.

```
public class SPYNumberDateTimeFormat
{
    public final static long spySystemLocale      = 0;
    public final static long spySchemaCompatible  = 1;
}
```

SPYProgrammingLanguage (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYLoading
 {
    public final static long spyUndefinedLanguage   = -1;
    public final static long spyJava                = 0;
    public final static long spyCpp                 = 1;
    public final static long spyCSharp              = 2;
 }
```

SPYProjectItemTypes (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYProjectItemTypes
{
    public final static long spyUnknownItem  = 0;
    public final static long spyFileItem     = 1;
    public final static long spyFolderItem   = 2;
    public final static long spyURLItem      = 3;
}
```

SPYProjectType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYProjectType
  {
     public final static long spyVisualStudioProject    = 0;
     public final static long                           = 1;
     spyVisualStudio2003Project
     public final static long spyBorlandProject         = 2;
     public final static long spyMonoMakefile           = 3;
  }
```

SPYSampleXMLGenerationOptimization (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYSampleXMLGenerationOptimization
{
   public final static long spySampleXMLGen_Optimized        = 0;
   public final static long                                  = 1;
   spySampleXMLGen NonMandatoryElements
   public final static long spySampleXMLGen Everything       = 2;
}
```

SPYSampleXMLGenerationSchemaOrDTDAssignment (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYSampleXMLGenerationOptimization
{
   public final static long spySampleXMLGen_AssignRelatively  = 0;
   public final static long spySampleXMLGen_AssignAbsolutely  = 1;
   public final static long spySampleXMLGen_DoNotAssign       = 2;
}
```

SPYSchemaDefKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYSchemaDefKind
{
    public final static long spyKindElement          = 0;
    public final static long spyKindComplexType      = 1;
    public final static long spyKindSimpleType       = 2;
    public final static long spyKindGroup            = 3;
    public final static long spyKindModel            = 4;
    public final static long spyKindAny              = 5;
    public final static long spyKindAttr             = 6;
    public final static long spyKindAttrGroup        = 7;
    public final static long spyKindAttrAny          = 8;
    public final static long spyKindIdentityUnique   = 9;
    public final static long spyKindIdentityKey      = 10;
    public final static long spyKindIdentityKeyRef   = 11;
    public final static long spyKindIdentitySelector = 12;
    public final static long spyKindIdentityField    = 13;
    public final static long spyKindNotation         = 14;
    public final static long spyKindInclude          = 15;
    public final static long spyKindImport           = 16;
    public final static long spyKindRedefine         = 17;
    public final static long spyKindFacet            = 18;
    public final static long spyKindSchema           = 19;
    public final static long spyKindCount            = 20;
}
```

SPYSchemaDocumentationFormat (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYSchemaDocumentationFormat
 {
    public final static long spySchemaDoc_HTML  = 0;
    public final static long spySchemaDoc_MSWord= 1;
    public final static long spySchemaDoc_RTF   = 2;
    public final static long spySchemaDoc_PDF   = 3;
 }
```

SPYSchemaExtensionType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYSchemaExtensionType
  {
     public final static long spySchemaExtension_None          = 0;
     public final static long spySchemaExtension_SQL_XML       = 1;
     public final static long spySchemaExtension_MS_SQL_Server = 2;
     public final static long spySchemaExtension_Oracle        = 3;
  }
```

SPYSchemaFormat (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYSchemaFormat
  {
     public final static long spySchemaFormat_Hierarchical  = 0;
     public final static long spySchemaFormat_Flat          = 1;
  }
```

SPYTextDelimiters (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYTextDelimiters
{
     public final static long spyTabulator    = 0;
     public final static long spySemicolon    = 1;
     public final static long spyComma        = 2;
     public final static long spySpace        = 3;
}
```

SPYTextEnclosing (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYTextEnclosing
{
    public final static long spyNoEnclosing   = 0;
    public final static long spySingleQuote   = 1;
    public final static long spyDoubleQuote   = 2;
}
```

SPYTypeDetection (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYTypeDetection
{
    public final static long spyBestPossible    = 0;
    public final static long spyNumbersOnly     = 1;
    public final static long spyNoDetection     = 2;
}
```

SPYURLTypes (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYURLTypes
{
    public final static long spyURLTypeAuto  = (-1);
    public final static long spyURLTypeXML   = 0;
    public final static long spyURLTypeDTD   = 1;
}
```

SpyViewModes (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYViewModes
{
    public final static long spyViewGrid          = 0;
    public final static long spyViewText          = 1;
    public final static long spyViewBrowser       = 2;
    public final static long spyViewSchema        = 3;
    public final static long spyViewContent       = 4;
    public final static long spyViewAuthentic     = 4;
    public final static long spyViewWSDL          = 5;
    public final static long spyViewZIP           = 6;
    public final static long spyViewEditionInfo   = 7;
}
```

SPYWhitespaceComparison (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYWhitespaceComparison
 {
    public final static long spyCompareAsIs       = 0;
    public final static long spyCompareNormalized = 1;
    public final static long spyStripAll          = 2;
 }
```

SPYXMLDataKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYXMLDataKind
{
```

```
    public final static long                        = 0;
    spyXMLDataXMLDocStruct
    public final static long                        = 1;
    spyXMLDataXMLEntityDocStruct
    public final static long                        = 2;
    spyXMLDataDTDDocStruct
    public final static long spyXMLDataXML          = 3;
    public final static long spyXMLDataElement      = 4;
    public final static long spyXMLDataAttr         = 5;
    public final static long spyXMLDataText         = 6;
    public final static long spyXMLDataCData        = 7;
    public final static long spyXMLDataComment      = 8;
    public final static long spyXMLDataPI           = 9;
    public final static long spyXMLDataDefDoctype   = 10;
    public final static long                        = 11;
    spyXMLDataDefExternalID
    public final static long spyXMLDataDefElement   = 12;
    public final static long spyXMLDataDefAttlist   = 13;
    public final static long spyXMLDataDefEntity    = 14;
    public final static long spyXMLDataDefNotation  = 15;
    public final static long spyXMLDataKindsCount   = 16;
}
```

4 ActiveX Integration

The XMLSpy user interface and the functionality described in this section can be integrated into custom applications that can consume ActiveX controls. ActiveX technology enables a wide variety of languages to be used for integration, such as C++, C#, VB.NET, HTML. (Note that ActiveX components integrated in HTML must be run with Microsoft Internet Explorer versions and platforms that support ActiveX). All components are full OLE Controls. Integration into Java is provided through wrapper classes.

To integrate the ActiveX controls into your custom code, the XMLSpy Integration Package must be installed (see http://www.altova.com/ide_integration.html). Ensure that you install XMLSpy first, and then the XMLSpy Integration Package. Other prerequisites apply, depending on language and platform (see Prerequisites).

You can flexibly choose between two different levels of integration: application level and document level.

Integration at application level means embedding the complete interface of XMLSpy (including its menus, toolbars, panes, etc) as an ActiveX control into your custom application. For example, in the most simple scenario, your custom application could consist of only one form that embeds the XMLSpy graphical user interface. This approach is easier to implement than integration at document level but may not be suitable if you need flexibility to configure the XMLSpy graphical user interface according to your custom requirements.

Integration at document level means embedding XMLSpy into your own application piece-by-piece. This includes implementing not only the main XMLSpy control but also the main document editor window, and, optionally, any additional windows. This approach provides greater flexibility to configure the GUI, but requires advanced interaction with ActiveX controls in your language of choice.

The sections Integration at the Application Level and Integration at Document Level describe the key steps at these respective levels. The ActiveX Integration Examples section provides examples in C#, HTML, and Java. Looking through these examples will help you to make the right decisions quickly. The Object Reference section describes all COM objects that can be used for integration, together with their properties and methods.

For information about using XMLSpy as a Visual Studio plug-in, see XMLSpy in Visual Studio.

4.1 Prerequisites

To integrate the XMLSpy ActiveX control into a custom application, the following must be installed on your computer:

- XMLSpy
- The XMLSpy Integration Package, available for download at http://www.altova.com/ide_integration.html

To integrate the 64-bit ActiveX control, install the 64-bit versions of XMLSpy and XMLSpy Integration Package. For applications developed under Microsoft .NET platform with Visual Studio, both the 32-bit and 64-bit versions of XMLSpy and XMLSpy Integration Package must be installed, as explained below.

Note: For the integration package to function correctly on Windows XP and Windows Server 2003, you will need to manually install the Windows redistributable available at https://www.microsoft.com/en-US/download/details.aspx?id=48145.

Microsoft .NET (C#, VB.NET) with Visual Studio

To integrate the XMLSpy ActiveX control into a 32-bit application developed under Microsoft .NET, the following must be installed on your computer:

- Microsoft .NET Framework 4.0 or later
- Visual Studio 2005/2008/2010/2012/2013/2015/2017
- XMLSpy 32-bit and XMLSpy Integration Package 32-bit
- The ActiveX controls must be added to the Visual Studio toolbox (see Adding the ActiveX Controls to the Toolbox).

If you want to integrate the 64-bit ActiveX control, the following prerequisites apply in addition to the ones above:

- XMLSpy 32-bit and XMLSpy Integration Package 32-bit must still be installed (this is required to provide the 32-bit ActiveX control to the Visual Studio designer, since Visual Studio runs on 32-bit)
- XMLSpy 64-bit and XMLSpy Integration Package 64-bit must be installed (provides the actual 64-bit ActiveX control to your custom application at runtime)
- In Visual Studio, create a 64-bit build configuration and build your application using this configuration. For an example, see Running the Sample C# Solution.

Java

To integrate the XMLSpy ActiveX control into Java application using the Eclipse development environment, the following must be installed on your computer:

- Java Runtime Environment (JRE) or Java Development Kit (JDK) 7 or later
- Eclipse
- XMLSpy and XMLSpy Integration Package

Note: To run the 64-bit version of the XMLSpy ActiveX control, use a 64-bit version of Eclipse, as well as the 64-bit version of XMLSpy and the XMLSpy Integration Package.

XMLSpy integration and deployment on client computers

If you create a .NET application and intend to distribute it to other clients, you will need to install the following on the client computer(s):

- XMLSpy
- The XMLSpy Integration Package
- The custom integration code or application.

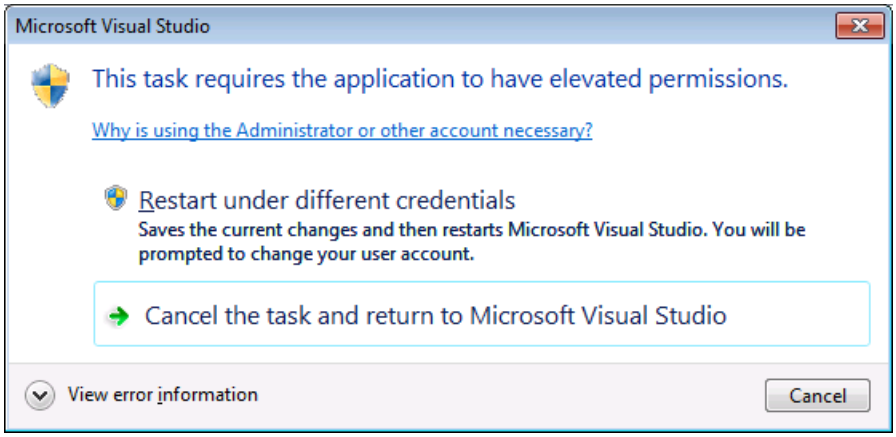
4.2 Adding the ActiveX Controls to the Toolbox

To use the XMLSpy ActiveX controls in an application developed with Visual Studio, the controls must first be added to the Visual Studio Toolbox, as follows:

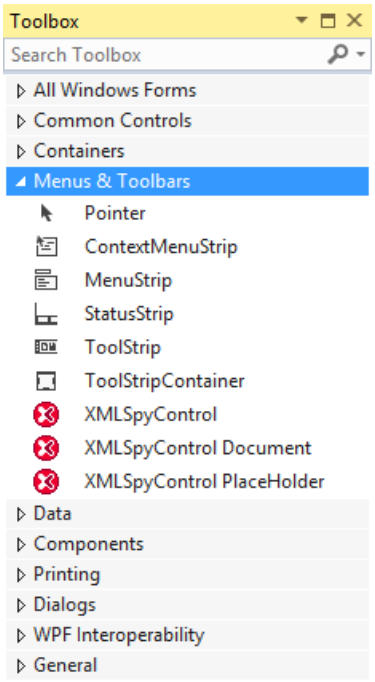
1. On the **Tools** menu of Visual Studio, click **Choose Toolbox Items**.
2. On the **COM Components** tab, select the check boxes next to the XMLSpyControl, XMLSpyControl Document, and XMLSpyControl Placeholder.

In case the controls above are not available, follow the steps below:

1. On the **COM Components** tab, click **Browse**, and select the **XMLSpyControl.ocx** file from the XMLSpy installation folder. Remember that the XMLSpy Integration Package must be installed; otherwise, this file is not available, see Prerequisites.
2. If prompted to restart Visual Studio with elevated permissions, click **Restart under different credentials**.



If the steps above were successful, the XMLSpy ActiveX controls become available in the Visual Studio Toolbox.



Note: For an application-level integration, only the **XMLSpyControl** ActiveX control is used (see Integration at Application Level). The **XMLSpyControl Document** and **XMLSpyControl Placeholder** controls are used for document-level integration (see Integration at Document Level).

4.3 Integration at Application Level

Integration at application level allows you to embed the complete interface of XMLSpy into a window of your application. With this type of integration, you get the whole user interface of XMLSpy, including all menus, toolbars, the status bar, document windows, and helper windows. Customization of the application's user interface is restricted to what XMLSpy provides. This includes rearrangement and resizing of helper windows and customization of menus and toolbars.

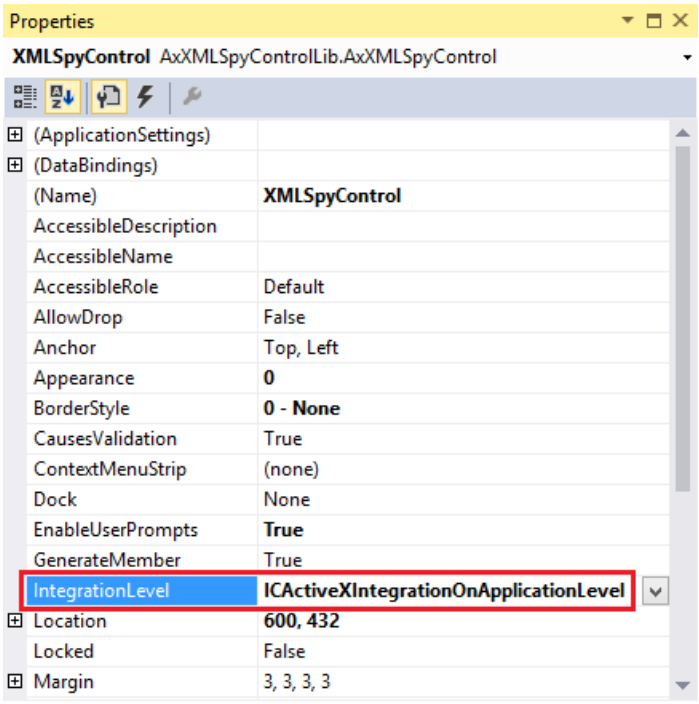
The only ActiveX control you need to integrate is `XMLSpyControl`. Do not instantiate or access `XMLSpyControlDocument` or `XMLSpyControlPlaceHolder` ActiveX controls when integrating at application-level.

If you have any initialization to do or if you want to automate some behaviour of XMLSpy, use the properties, methods, and events described for `XMLSpyControl`. Consider using `XMLSpyControl.Application` for more complex access to XMLSpy functionality.

For an example that shows how the XMLSpy application can be embedded in an HTML page, see HTML Integration at Application Level.

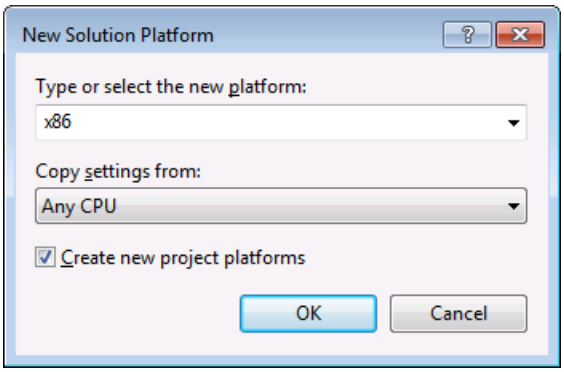
In C# or VB.NET with Visual Studio, the steps to create a basic, one-form application which integrates the XMLSpy ActiveX controls at application level are as follows:

1. Check that all prerequisites are met (see Prerequisites).
2. Create a new Visual Studio Windows Forms project with a new empty form.
3. If you have not done that already, add the ActiveX controls to the toolbox (see Adding the ActiveX Controls to the Toolbox).
4. Drag the **XMLSpyControl** from the toolbox onto your new form.
5. Select the **XMLSpyControl** on the form, and, in the Properties window, set the **IntegrationLevel** property to **ICActiveXIntegrationOnApplicationLevel**.



6. Create a build platform configuration that matches the platform under which you want to build (x86, x64). Here is how you can create the build configuration:

   a. Right-click the solution in Visual Studio, and select **Configuration Manager**.
   b. Under **Active solution platform**, select **New...** and then select the x86 or x64 configuration (in this example, **x86**).



You are now ready to build and run the solution in Visual Studio. Remember to build using the configuration that matches your target platform (x86, x64).

4.4 Integration at Document Level

Compared to integration at application level, integration at document level is a more complex, yet more flexible way to embed XMLSpy functionality into your application by means of ActiveX controls. With this approach, your code can access selectively the following parts of the XMLSpy user interface:

- Document editing window
- Project window
- Entry helper windows
- Validator output window
- XPath profiler window
- XPath dialog window
- XSLT/XQuery debugger windows
- SOAP debugger window

As mentioned in Integration at Application Level, for an ActiveX integration at application level, only one control is required, namely the **XMLSpyControl**. However, for an ActiveX integration at document level, functionality XMLSpy is provided by the following ActiveX controls:

1. **XMLSpyControl**
2. **XMLSpyControl Document**
3. **XMLSpyControl Placeholder**

These controls are supplied by the **XMLSpyControl.ocx** file available in the application installation folder of XMLSpy. When you develop the ActiveX integration with Visual Studio, you will need to add these controls to the Visual Studio toolbox (see Adding the ActiveX Controls to the Toolbox).

The basic steps to integrate the ActiveX controls at document level into your application are as follows:

1. First, instantiate **XMLSpyControl** in your application. Instantiating this control is mandatory; it enables support for the **XMLSpyControl Document** and **XMLSpyControl Placeholder** controls mentioned above. It is important to set the `IntegrationLevel` property to **ICActiveXIntegrationOnDocumentLevel** (or "1"). To hide the control from the user, set its **Visible** property to **False**.

Note: When integrating at document level, do not use the **Open** method of the **XMLSpyControl**; this might lead to unexpected results. Use the corresponding open methods of **XMLSpyControl Document** and **XMLSpyControl PlaceHolder** instead.

2. Create at least one instance of **XMLSpyControl Document** in your application. This control supplies the document editing window of XMLSpy to your application and can be instantiated multiple times if necessary.

   Use the method **Open** to load any existing file. To access document-related functionality, use the **Path** and **Save** or methods and properties accessible via the property **Document**.

Note: The control does not support a read-only mode. The value of the property **ReadOnly** is ignored.

3. Optionally, add to your application the **XMLSpyControl Placeholder** control for each additional window (other than the document window) that must be available to your application.

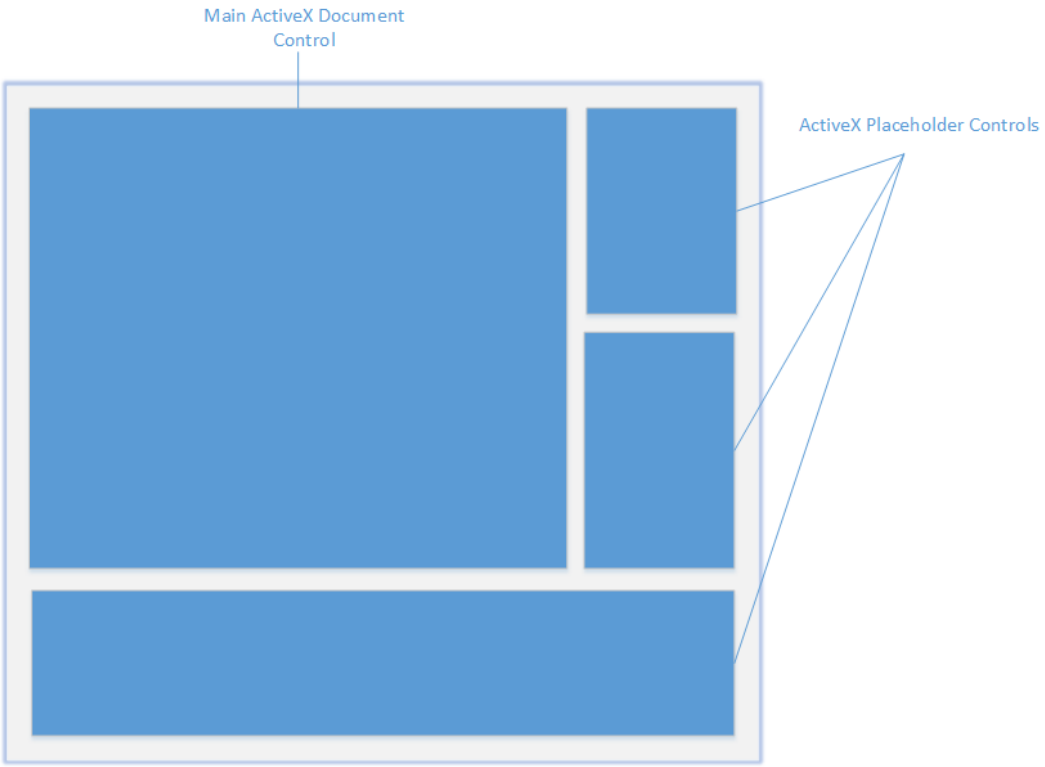
Instances of **XMLSpyControl PlaceHolder** allow you to selectively embed additional windows of XMLSpy into your application. The window kind (for example, Project window) is defined by the property **PlaceholderWindowID**. Therefore, to set the window kind, set the property **PlaceholderWindowID**. For valid window identifiers, see XMLSpyControlPlaceholderWindow.

Note: Use only one **XMLSpyControl PlaceHolder** for each window identifier.

For placeholder controls that select the XMLSpy project window, additional methods are available. Use **OpenProject** to load a XMLSpy project. Use the property Project and the methods and properties from the XMLSpy automation interface to perform any other project related operations.

For example, in C# or VB.NET with Visual Studio, the steps to create a basic, one-form application which integrates the XMLSpy ActiveX controls at document level could be similar to those listed below. Note that your application may be more complex if necessary; however, the instructions below are important to understand the minimum requirements for an ActiveX integration at document level.

1. Create a new Visual Studio Windows Forms project with a new empty form.
2. If you have not done that already, add the ActiveX controls to the toolbox (see Adding the ActiveX Controls to the Toolbox).
3. Drag the **XMLSpyControl** from the toolbox onto your new form.
4. Set the **IntegrationLevel** property of the **XMLSpyControl** to **ICActiveXIntegrationOnDocumentLevel**, and the **Visible** property to **False**. You can do this either from code or from the **Properties** window.
5. Drag the **XMLSpyControl Document** from the toolbox onto the form. This control provides the main document window of XMLSpy to your application, so you may need to resize it to a reasonable size for a document.
6. Optionally, add one or more **XMLSpyControl Placeholder** controls to the form (one for each additional window type that your application needs, for example, the **Project** window). You will typically want to place such additional placeholder controls either below or to the right or left of the main document control, for example:



7. Set the **PlaceholderWindowID** property of each **XMLSpyControl Placeholder** control to a valid window identifier. For the list of valid values, see XMLSpyControlPlaceholderWindow.
8. Add commands to your application (at minimum, you will need to open, save and close documents), as shown below.

Querying XMLSpy Commands

When you integrate at document level, no XMLSpy menu or toolbar is available to your application. Instead, you can retrieve the required commands, view their status, and execute them programmatically, as follows:

- To retrieve all available commands, use the `CommandsList` property of the **XMLSpyControl**.
- To retrieve commands organized according to their menu structure, use the `MainMenu` property.
- To retrieve commands organized by the toolbar in which they appear, use the `Toolbars` property.
- To send commands to XMLSpy, use the `Exec` method.
- To query if a command is currently enabled or disabled, use the `QueryStatus` method.

This enables you to flexibly integrate XMLSpy commands into your application's menus and toolbars.

Your installation of XMLSpy also provides you with command label images used within XMLSpy. See the folder **<ApplicationFolder>\Examples\ActiveX\Images** of your XMLSpy installation for icons in GIF format. The file names correspond to the command names as they are listed in the Command Reference section.

General considerations

To automate the behaviour of XMLSpy, use the properties, methods, and events described for the **XMLSpyControl**, **XMLSpyControl Document**, and **XMLSpyControl Placeholder**.

For more complex access to XMLSpy functionality, consider using the following properties:

- `XMLSpyControl.Application`
- `XMLSpyControlDocument.Document`
- `XMLSpyControlPlaceHolder.Project`

These properties give you access to the XMLSpy automation interface (XMLSpyAPI)

Note: To open a document, always use `XMLSpyControlDocument.Open` or `XMLSpyControlDocument.New` on the appropriate document control. To open a project, always use `XMLSpyControlPlaceHolder.OpenProject` on a placeholder control embedding a XMLSpy project window.

For examples that show how to instantiate and access the necessary controls in different programming environments, see ActiveX Integration Examples.

4.5 ActiveX Integration Examples

This section contains examples of XMLSpy document-level integration using different container environments and programming languages. (The HTML section additionally contains examples of integration at application level.) Source code for all examples is available in the folder `<ApplicationFolder>\Examples\ActiveX` of your XMLSpy installation.

4.5.1 C#

A basic ActiveX integration example solution for C# and Visual Studio is available in the folder `<ApplicationFolder>\Examples\ActiveX\C#`. Before you compile the source code and run the sample, make sure that all prerequisites are met (see Running the Sample C# Solution).

Running the Sample C# Solution

The sample Visual Studio solution available in the folder **<ApplicationFolder>\Examples\ActiveX\C#** illustrates how to consume the XMLSpy ActiveX controls. Before attempting to build and run this solution, note the following steps:

Step 1: Check the prerequisites

Visual Studio 2010 or later is required to open the sample solution. For the complete list of prerequisites, see Prerequisites.

Step 2: Copy the sample to a directory where you have write permissions

To avoid running Visual Studio as an Administrator, copy the source code to a directory where you have write permissions, instead of running it from the default location.

Step 3: Check and set all required control properties

The sample application contains one instance of XMLSpyControlDocument and one instance of XMLSpyControlPlaceHolder controls. Double-check that the following properties of these controls are set as shown in the table below:

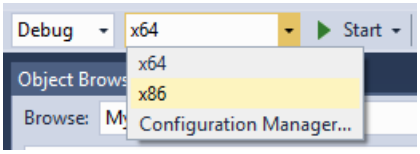
Control name	Property	Property value
XMLSpyControl	IntegrationLevel	ICActiveXIntegrationOnDocumentLevel
XPathDialog	PlaceholderWindowID	16

Here is how you can view or set the properties of an ActiveX control:

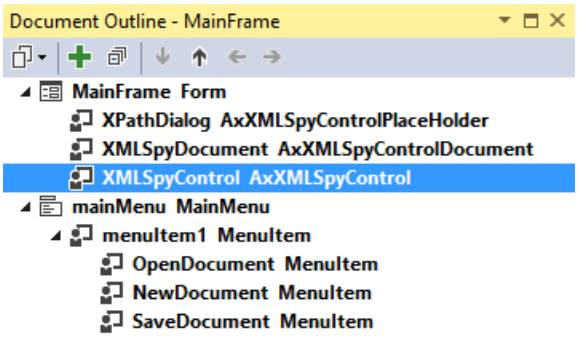
1. Open the **MDIMain.cs** form in the designer window.

Note: On 64-bit Windows, it may be necessary to change the build configuration of the Visual Studio solution to "x86" **before** opening the designer window. If you need to build the

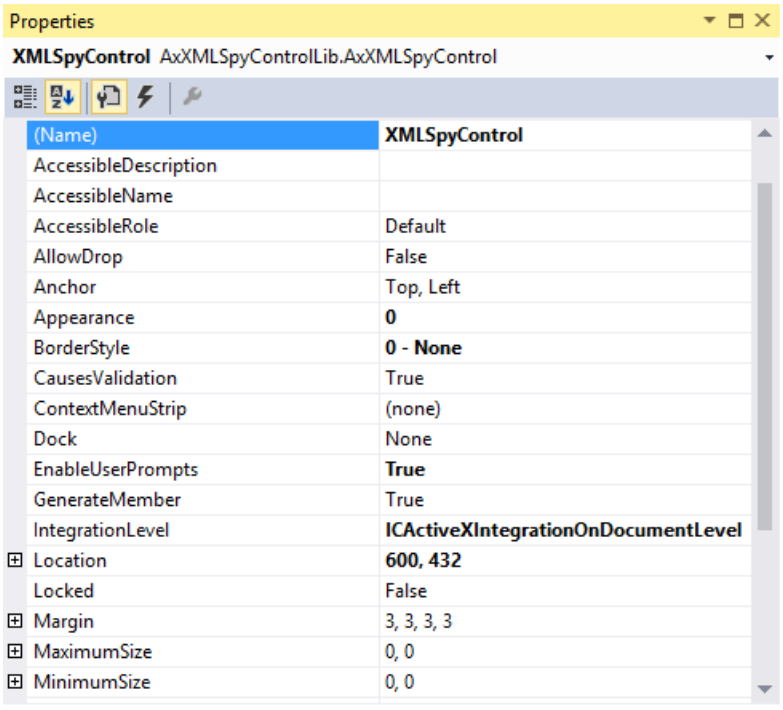
sample as a 64-bit application, see Prerequisites.



2. Open the **Document Outline** window of Visual Studio (On the **View** menu, click **Other Windows | Document Outline**).



3. Click an ActiveX control in the **Document Outline** window, and edit its required property in the **Properties** window, for example:

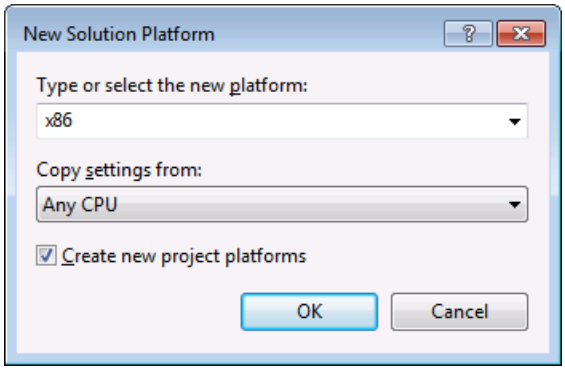


Step 4: Set the build platform

- Create a build platform configuration that matches the platform under which you want to

build (x86, x64). Here is how you can create the build configuration:

a. Right-click the solution in Visual Studio, and select **Configuration Manager**.
b. Under **Active solution platform**, select **New...** and then select the x86 or x64 configuration (in this example, **x86**).



You are now ready to build and run the solution in Visual Studio. Remember to build using the configuration that matches your target platform (x86, x64); otherwise, runtime errors might occur.

On running the sample, the main MDI Frame window is created and contains an editing window with an empty XML document and a XPath Dialog window of XMLSpy at the bottom. Use **File | Open** to open any XML file from the XMLSpy examples folder. The file is loaded and displayed. After you load the document, you can start using the XPath dialog. Note that you may need to slightly drag the lower-right corner of the form to cause the dialog to redraw itself and display its contents.

4.5.2 HTML

The code listings in this section show how to integrate the XMLSpyControl at application level and document level. Source code for all examples is available in the folder `<ApplicationFolder>\Examples\ActiveX\HTML` of your XMLSpy installation. The examples work only in Internet Explorer when it runs as a 32-bit application.

Note: When it runs in 64-bit mode, Internet Explorer 10 or later does not load ActiveX controls.

HTML Integration at Application Level

This example shows a simple integration of the XMLSpy control at application-level into a HTML page. The integration is described in the following sections:

- Instantiate a XMLSpyControl in HTML code.
- Implement buttons to load documents and automate code-generation tasks.
- Define actions for some application events.

The code for this example is available at the following location in your XMLSpy installation: `<ApplicationFolder>\Examples\ActiveX\HTML\XMLSpyActiveX_ApplicationLevel.htm`.

Instantiate the Control

The HTML `Object` tag is used to create an instance of the XMLSpyControl. The `Classid` is that of XMLSpyControl. Width and height specify the window size. No additional parameters are necessary, since application-level is the default.

```
<OBJECT id="objXMLSpyControl"
        Classid="clsid:a258bba2-3835-4c16-8590-72b44f52c471"
        width="1000"
        height="700"
        VIEWASTEXT>
</OBJECT>
```

Add Button to Open Default Document

As a simple example of how to automate some tasks, we add a button to the page:

```
<input type="button" value="Open Marketing Expenses"
onclick="BtnOpenMEFile()">
```

When clicked, a predefined document will be opened in the XMLSpyControl. We use a method to locate the file relative to the XMLSpyControl so the example can run on different installations.

```
<SCRIPT ID=Javahandlers LANGUAGE=javascript>
// --------------------------------
// open a pre-defined document
function BtnOpenMEFile()
{

             objXMLSpyControl.Open("C:\Documents and Settings\username\My
Documents\Altova\XMLSpy2017\Examples/OrgChart.xml");

}
</SCRIPT>
```

Add Buttons for Code Generation

For direct access, we want to have a button that will validate the current document. The method is similar to that used in the previous section.

First comes the button:

```
<input type="button" value="Validate" onclick="BtnValidate()">
```

Then we provide the script that will validate the current document.

```
<SCRIPT ID=Javahandlers LANGUAGE=javascript>
// ------------------------------------------------------------------------
// check validity of current document.
// if validation fails, show validation result in alert box .
function BtnValidate()
{
       // get top-level object of automation interface
       var objApp = objXMLSpyControl.Application;

       // get the active document
       var objDocument = objApp.ActiveDocument;

       if ( objDocument == null )
              alert( "no active document found" );
       else
       {
              // define as arrays to support their usage as return parameters
              var errorText = new Array(1);
```

```
                var errorPos = new Array(1);
                var badData = new Array(1);

                var valid = objDocument.IsValid(errorText, errorPos, badData);

                if (! valid)
                {
                          // compose the error description
                          var text = errorText;

                          // access that XMLData object only if filled in
                          if (badData[0] != null)
                                text += "(" + badData[0].Name + "/" +
badData[0].TextValue + ")";


                          alert("Validation error[" + errorPos + "]: " + text);
                }
                else
                       alert("Docuent is valid"); }
}
</SCRIPT>
```

Connect to Custom Events

The example implements two event callbacks for XMLSpyControl custom events to show the principle:

```
<!-- ------------------------------------------------------------ -->
<!--  custom event 'OnDocumentOpened" of XMLSpyControl object  -->
<SCRIPT LANGUAGE="javascript">
       function objXMLSpyControl::OnDocumentOpened( objDocument )
       {
       // alert("Document '" + objDocument.Name + "' opened!");
       }
</SCRIPT>

<!-- ------------------------------------------------------------ -->
<!--  custom event 'OnDocumentClosed" of XMLSpyControl object  -->
<SCRIPT LANGUAGE="javascript">
       function objXMLSpyControl::OnDocumentClosed( objDocument )
       {
       // alert("Document '" + objDocument.Name + "' closed!");
       }
</SCRIPT>
```

HTML Integration at Document Level

This example shows an integration of the XMLSpy control at document-level into a HTML page. The following topics are covered:

- Instantiate a XMLSpyControl ActiveX control object in HTML code
- Instantiate a XMLSpyControlDocument ActiveX control to allow editing a XMLSpy file
- Instantiate one XMLSpyControlPlaceHolder for a XMLSpyControl project window
- Instantiate one XMLSpyControlPlaceHolder to alternatively host one of the XMLSpy helper windows
- Instantiate one XMLSpyControlPlaceHolder ActiveX control to show the XMLSpy validation output window

- Create a simple customer toolbar for some heavy-used XMLSpy commands
- Add some more buttons that use the COM automation interface of XMLSpy
- Use event handlers to update command buttons

This example is available in its entirety in the file `XMLSpyActiveX_ApplicationLevel.htm` within the `<ApplicationFolder>\Examples\ActiveX\HTML\` folder of your XMLSpy installation.

Instantiate the XMLSpyControl

XMLSpyControlThe HTML `OBJECT` tag is used to create an instance of the XMLSpyControl. The Classid is that of XMLSpyControl. Width and height are set to 0 since we use this control as manager control without use for its user interface. The integration level is specified as a parameter within the `OBJECT` tag.

```
<OBJECT id="objXMLSpyXXMLSpyControl"
        Classid="clsid:a258bba2-3835-4c16-8590-72b44f52c471"
        width="0"
        height="0"
        VIEWASTEXT>
   <PARAM NAME="IntegrationLevel" VALUE="1">
</OBJECT>
```

Create Editor Window

The HTML `OBJECT` tag is used to embed an editing window.

```
<OBJECT id="objDoc1"
        Classid="clsid:DFBB0871-DAFE-4502-BB66-08CEB7DF5255"
        width="600"
        height="500"
        VIEWASTEXT>
</OBJECT>
```

Create Project Window

The HTML `OBJECT` tag is used to create a XMLSpyControlPlaceHolder window. The first additional custom parameter defines the placeholder to show the XMLSpy project window. The second parameter loads one of the example projects delivered with your XMLSpy installation (located in the `<yourusername>/MyDocuments` folder).

```
<OBJECT id="objProjectWindow"
        Classid="clsid:FDEC3B04-05F2-427d-988C-F03A85DE53C2"
        width="200"
        height="200"
        VIEWASTEXT>
    <PARAM name="PlaceholderWindowID" value="3">
          <PARAM name="FileName" value="Examples/Examples.spp">
</OBJECT>
```

Create Placeholder for Helper Windows

The HTML `OBJECT` tag is used to instantiate a XMLSpyControlPlaceHolder ActiveX control that can host the different XMLSpy helper windows. Initially, no helper window is shown. See the example file.

```
<OBJECT id="objEHWindow"
        Classid="clsid:135DEEF4-6DF0-47c2-8F8C-F145F5F3F672"
        width="200"
        height="200"
        VIEWASTEXT>
       <PARAM name="PlaceholderWindowID" value="0">
</OBJECT>
```

Three buttons allow us to switch the actual window that will be shown. The JavaScript execute on-button-click sets the property `PlaceHolderWindowID` to the corresponding value defined in XMLSpyControlPlaceholderWindow.

```
<input type="button" value="EH - Elements" onclick="BtnHelperWindow(0)">
<input type="button" value="EH - Attributes" onclick="BtnHelperWindow(1)">
<input type="button" value="EH - Entities" onclick="BtnHelperWindow(2)">

<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
//
---------------------------------------------------------------------------
// specify which of the windows shall be shown in the placeholder control.
function BtnHelperWindow(i_ePlaceholderWindowID)
{
       objEHWindow.PlaceholderWindowID = i_ePlaceholderWindowID;
}
</SCRIPT>
```

Create a Custom Toolbar

The custom toolbar consists of buttons with images of XMLSpy commands. The command ID numbers can be found in the button elements shown below Command Table.

```
<button id="btnWellFormed" title="Check Well-formedness"
onclick="BtnDoCommand(34049)">
       <img src="..\Images\IDC_CHECK_WELL_FORM.gif" width="16" height="16" />
</button>
<button id="btnValidate" title="Validate" onclick="BtnDoCommand(34174)">
       <img src="..\Images\IDC_VALIDATE.gif" width="16" height="16" />
</button>
```

On clicking one of these buttons the corresponding command ID is sent to the manager control.

```
<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
// ---------------------------------------------------------------
// perform any command specified by cmdID.
// command routing includes application, active document and view.
function BtnDoCommand(cmdID)
{
      objXMLSpyX.Exec(cmdID);
      msgtext.innerText = "Command " + cmdID + " performed.";
}
```

```
</SCRIPT>
```

Create More Buttons

In the example, we add some more buttons to show some automation code.

```
<p>
	<input type="button" value="New File" onclick="BtnNewFile(objDoc1)">
	<input type="button" value="Save File" onclick="BtnSaveFile(objDoc1)">
	<input type="text" title="Path" id="strPath" width="150">
	<input type="button" value="Open OrgChart.xml"
onclick="BtnOpenFile(objDoc1, 'OrgChart.xml')">
	<input type="button" value="Open OrgChart.xsd"
onclick="BtnOpenFile(objDoc1, 'OrgChart.xsd')">
<p>
```

The corresponding JavaScript looks like this:

```
<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
// ---------------------------------------------------------
// open a document in the specified document controlwindow.
function BtnOpenFile(objDocCtrl, strFileName)
{
	// do not use XMLSpyXApplication.OpenDocument(...) to open a document,
	// since then XMLSpyControlwouldn't know a controlwindow to show
	// the document in. Instead:

	objDocCtrl.Open("C:\Documents and Settings\username\My Documents\Altova\XMLSpy2017
\Examples/" + strFileName);
	objDocCtrl.setActive();

}

// ---------------------------------------------------------------------
// open a new empty document in the specified document controlwindow.
function BtnNewFile(objDocCtrl)
{
	objDocCtrl.Open("");
	objDocCtrl.setActive();
}

// ---------------------------------------------------------------------
// Saves the current file in the specified document controlwindow.
function BtnSaveFile(objDocCtrl)
{
	if(objDocCtrl.Path.length > 0)
		objDocCtrl.SaveDocument();
	else
	{
		if(strPath.value.length > 0)
		{
			objDocCtrl.Path = strPath.value;
			objDocCtrl.Save();
		}
		else
		{
			alert("Please set path for the document first!");
			strPath.focus();
		}
```

```
        }

        objDocCtrl.setActive();
}

// ----------------------------------------------------------------------
// check validity of current document.
// if validation fails, show validation result in alert box.
function BtnValidate()
{
        // get top-level object of automation interface
        var objApp = objXMLSpyControl.Application;

        // get the active document
        var objDocument = objApp.ActiveDocument;

        if (objDocument == null)
                alert("no active document found");
        else
        {
                // define as arrays to support their usage as return parameters
                var errorText = new Array(1);
                var errorPos = new Array(1);
                var badData = new Array(1);

                var valid = objDocument.IsValid(errorText, errorPos, badData);

                if (!valid)
                {
                        // compose the error description
                        var text = errorText;

                        // access that XMLData object only if filled in
                        if (badData[0] != null)
                                text += " (" + badData[0].Name + "/" + badData[0].TextValue + ")";


                        alert("Validation error[" + errorPos + "]:" + text);
                }
                else
                        alert("Docuent is valid");
        }
}
</SCRIPT>
```

Create Event Handler to Update Button Status

Availability of a command may vary with every mouse click or keystroke. The custom event `OnUpdateCmdUI` of XMLSpyControl gives us an opportunity to update the enabled/disabled state of buttons associated with XMLSpy commands. The method `XMLSpyControl.QueryStatus` is used to query whether a command is enabled or not.

```
<SCRIPT LANGUAGE="javascript">

        function objXMLSpyX::OnUpdateCmdUI()
        {
        if (document.readyState == "complete" )                 // 'complete'
```

```
        {
            // update status of buttons
            btnWellFormed.disabled = !(objDoc1.QueryStatus(34049) & 0x02);   // not enabled
            btnValidate.disabled = !(objDoc1.QueryStatus(34174) & 0x02);     // not enabled
        }
        }

        // set activity status of simulated toolbar
</SCRIPT>
```

4.5.3 Java

XMLSpy ActiveX components can be accessed from Java code. Java integration is provided by the libraries listed below. These libraries are available in the folder `<ApplicationFolder>\Examples\JavaAPI` of your XMLSpy installation, after you have installed both XMLSpy and the XMLSpy Integration Package (see also Prerequisites).

- `AltovaAutomation.dll`: a JNI wrapper for Altova automation servers (in case of the 32-bit installation of XMLSpy)
- `AltovaAutomation_x64.dll`: a JNI wrapper for Altova automation servers (in case of the 64-bit installation of XMLSpy)
- `AltovaAutomation.jar`: Java classes to access Altova automation servers
- `XMLSpyActiveX.jar`: Java classes that wrap the XMLSpy ActiveX interface
- `XMLSpyActiveX_JavaDoc.zip`: a Javadoc file containing help documentation for the Java interface

Note: In order to use the Java ActiveX integration, the .dll and .jar files must be included in the Java class search path.

Example Java project

An example Java project is supplied with your product installation. You can test the Java project and modify and use it as you like. For more details, see Example Java Project.

Rules for mapping the ActiveX Control names to Java

For the documentation of ActiveX controls, see Object Reference. Note that the object naming conventions are slightly different in Java compared to other languages. Namely, the rules for mapping between the ActiveX controls and the Java wrapper are as follows:

- ***Classes and class names***
  For every component of the XMLSpy ActiveX interface a Java class exists with the name of the component.

- ***Method names***
  Method names on the Java interface are the same as used on the COM interfaces but start with a small letter to conform to Java naming conventions. To access COM properties, Java methods that prefix the property name with get and set can be used. If a property does not support write-access, no setter method is available. Example: For the `IntegrationLevel` property of the XMLSpyControl, the Java methods `getIntegrationLevel` and `setIntegrationLevel` are available.

- ***Enumerations***
  For every enumeration defined in the ActiveX interface, a Java enumeration is defined

with the same name and values.

- ***Events and event handlers***
  For every interface in the automation interface that supports events, a Java interface with the same name plus 'Event' is available. To simplify the overloading of single events, a Java class with default implementations for all events is provided. The name of this Java class is the name of the event interface plus '`DefaultHandler`'. For example:
  `XMLSpyControl`: Java class to access the application
  `XMLSpyControlEvents`: Events interface for the XMLSpyControl
  `XMLSpyControlEventsDefaultHandler`: Default handler for `XMLSpyControlEvents`

Exceptions to mapping rules

There are some exceptions to the rules listed above. These are listed below:

Interface	Java name
`XMLSpyControlDocument`, method `New`	`newDocument`
`Document`, method `SetEncoding`	`setFileEncoding`
`AuthenticView`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Clone`	`cloneRange`

This section

This section shows how some basic XMLSpy ActiveX functionality can be accessed from Java code. It is organized into the following sub-sections:

- Example Java Project
- Creating the ActiveX Controls
- Loading Data in the Controls
- Basic Event Handling
- Menus
- UI Update Event Handling
- Creating an XML Tree

Example Java Project

The XMLSpy installation package contains an example Java project, located in the ActiveX Examples folder of the application folder: `<ApplicationFolder>\Examples\ActiveX\Java\`.

The Java example shows how to integrate the XMLSpyControl in a common desktop application created with Java. You can test it directly from the command line using the batch file `BuildAndRun.bat`, or you can compile and run the example project from within Eclipse. See below for instructions on how to use these procedures.

File list

The Java examples folder contains all the files required to run the example project. These files are listed below:

.classpath	Eclipse project helper file
.project	Eclipse project file
AltovaAutomation.dll	Java-COM bridge: DLL part (for the 32-bit installation)
AltovaAutomation_x64.dll	Java-COM bridge: DLL part (for the 64-bit installation)
AltovaAutomation.jar	Java-COM bridge: Java library part
BuildAndRun.bat	Batch file to compile and run example code from the command line prompt. Expects folder where Java Virtual Machine resides as parameter.
XMLSpyActiveX.jar	Java classes of the XMLSpy ActiveX control
XMLSpyActiveX_JavaDoc.zip	Javadoc file containing help documentation for the Java API
XMLSpyContainer.java	Java example source code
XMLSpyContainerEventHandler.java	Java example source code
XMLTreeDialog.java	Java example source code

What the example does

The example places one XMLSpy document editor window, the XMLSpy project window, the XMLSpy XPath window and an XMLSpy entry helper in an AWT frame window. It reads out the File menu defined for XMLSpy and creates an AWT menu with the same structure. You can use this menu or the project window to open and work with files in the document editor.

You can modify the example in any way you like.

The following specific features are described in code listings:

- Creating the ActiveX Controls: Starts XMLSpy, which is registered as an automation server, or activates XMLSpy if it is already running.
- Loading Data in the Controls: Locates one of the example documents installed with XMLSpy and opens it.
- Basic Event Handling: Changes the view of all open documents to Text View. The code also shows how to iterate through open documents.
- Menus: Validates the active document and shows the result in a message box. The code shows how to use output parameters.
- UI Update Event Handling: Shows how to handle XMLSpy events.
- Creating an XML Tree: Shows how to create an XML tree and prepare it for modal activation.

Updating the path to the Examples folder

Before running the provided sample, you may need to edit the **XMLSpyContainer.java** file. Namely, check that the following path refers to the actual folder where the XMLSpy example files are stored on your operating system:

```
// Locate samples installed with the product.
final String strExamplesFolder = System.getenv( "USERPROFILE" ) + "\\My
Documents\\Altova\\XMLSpy2017\\XMLSpyExamples\\";
```

Running the example from the command line

To run the example from the command line:

1. Check that all prerequisites are met (see Prerequisites).
2. Open a command prompt window, change the current directory to the sample Java project folder, and type:

   ```
   buildAndRun.bat "<Path-to-the-Java-bin-folder>"
   ```

3. Press **Enter**.

The Java source in `XMLSpyContainer.java` will be compiled and then executed.

Compiling and running the example in Eclipse

To import the sample Java project into Eclipse:

1. Check that all prerequisites are met (see Prerequisites).
2. On the **File** menu, click **Import**.
3. Select **Existing Projects into Workspace**, and browse for the Eclipse project file located at `<ApplicationFolder>\Examples\ActiveX\Java\`. Since you may not have write-access in this folder, it is recommended to select the **Copy projects into workspace** check box on the Import dialog box.

To run the example application, right-click the project in Package Explorer and select the command **Run as | Java Application**.

Help for Java API classes is available through comments in code as well as the Javadoc view of Eclipse. To enable the Javadoc view in Eclipse, select the menu command **Window | Show View | JavaDoc**.

Creating the ActiveX Controls

The code listing below show how ActiveX controls can be created. The constructors will create the Java wrapper objects. Adding these Canvas-derived objects to a panel or to a frame will trigger the creation of the wrapped ActiveX object.

```
01 /**
02 * XMLSpy manager control - always needed
03 */
04  public static XMLSpyControl      xmlSpyControl= null;
05
06 /**
07 * XMLSpy document editing control
08 */
09  public static XMLSpyControlDocument   xmlSpyDocument= null;
10
11 /**
12 * Toolwindows - XMLSpy place-holder controls
13 */
14  private static XMLSpyControlPlaceHolder  xmlSpyProjectToolWindow = null;
15  private static XMLSpyControlPlaceHolder  xmlSpyXpathToolWindow = null;
16  private static XMLSpyControlPlaceHolder  xmlSpyEHAttributeToolWindow = null;
17
```

```
// Create the XMLSpy ActiveX control; the parameter determines that we want
// to place document controls and place-holder controls individually.
// It gives us full control over the menu, as well.
  xmlSpyControl = new XMLSpyControl(
IActiveXIntegrationLevel.IActiveXIntegrationOnDocumentLevel.getValue());
  xmlSpyDocument = new XMLSpyControlDocument();
  xmlSpyDocument.setPreferredSize(new Dimension(640, 480));

// Create a project window and open the sample project in it
  xmlSpyProjectToolWindow = new XMLSpyControlPlaceHolder(
  XMLSpyControlPlaceholderWindow.XMLSpyControlProjectWindowToolWnd.getValue());
  xmlSpyProjectToolWindow.setPreferredSize(new Dimension(200, 200));
  xmlSpyXpathToolWindow = new XMLSpyControlPlaceHolder(
XMLSpyControlPlaceholderWindow.XMLSpyControlXPathDialogToolWnd.getValue());
  xmlSpyEHAttributeToolWindow = new XMLSpyControlPlaceHolder(
XMLSpyControlPlaceholderWindow.XMLSpyControlEntryHelperTopToolWnd.getValue());

  frame.add(xmlSpyControl, BorderLayout.NORTH);
  frame.add(xmlSpyDocument, BorderLayout.CENTER);
  southPane.add(xmlSpyProjectToolWindow);
  southPane.add(xmlSpyXpathToolWindow);
  southPane.add(xmlSpyEHAttributeToolWindow);
```

Loading Data in the Controls

The code listing below show how data can be loaded in the ActiveX controls.

```
// Locate samples installed with the product.
  final String strExamplesFolder = System.getenv("USERPROFILE") +
"\\My Documents\\Altova\\XMLSpy2017\\Examples\\";
  xmlSpyProjectToolWindow.openProject(strExamplesFolder + "Examples.spp");
```

Basic Event Handling

The code listing below shows how basic events can be handled. When calling the XMLSpyControl's `open` method, or when trying to open a file via the menu or Project tree, the `onOpenedOrFocused` event is sent to the attached event handler. The basic handling for this event is opening the file by calling the XMLSpyDocumentControl's `open` method.

```
  // Open the PXF file when button is pressed
  btnOpenPxf.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      try {
        xmlSpyControl.open(strExamplesFolder + "OrgChart.pxf");
      } catch (AutomationException e1) {
        e1.printStackTrace();
      }
    }
  });
  public void onOpenedOrFocused(String istrFileName, boolean ibOpenWithThisControl, boolean
ibFileAlreadyOpened) throws AutomationException
{
  // Handle the New/Open events coming from the Project tree or from the menus
  if (!ibFileAlreadyOpened)
  {
    // This is basically an SDI interface, so open the file in the already existing document control
    try {
      XMLSpyContainer.xmlSpyDocument.open(istrFileName);
      XMLSpyContainer.xmlSpyDocument.requestFocusInWindow();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Menus

The code listing below shows how menu items can be created. Each `XMLSpyCommand` object gets a corresponding `MenuItem` object, with the `ActionCommand` set to the ID of the command. The actions generated by all menu items are handled by the same function, which can perform specific handlings (like reinterpreting the closing mechanism) or can delegate the execution to the `XMLSpyControl` object by calling its `exec` method. The `menuMap` object that is filled during menu creation is used later (see section UI Update Event Handling).

```
01 // Load the file menu when the button is pressed
02    btnMenu.addActionListener(new ActionListener() {
03     public void actionPerformed(ActionEvent e) {
04      try {
05       // Create the menubar that will be attached to the frame
06       MenuBar mb = new MenuBar();
07       // Load the main menu's first item - the File menu
08       XMLSpyCommand xmlSpyMenu = xmlSpyControl.getMainMenu().getSubCommands().getItem(0);
09       // Create Java menu items from the Commands objects
10       Menu fileMenu = new Menu();
11       handlerObject.fillMenu(fileMenu, xmlSpyMenu.getSubCommands());
12       fileMenu.setLabel(xmlSpyMenu.getLabel().replace("&", ""));
13       mb.add(fileMenu);
14       frame.setMenuBar(mb);
15       frame.validate();
16      }catch (AutomationException e1) {
17       e1.printStackTrace();
18      }
19      // Disable the button when the action has been performed
20      ((AbstractButton)e.getSource()).setEnabled(false);
21     }
22    });
23 /** * Populates a menu with the commands and submenus contained in an XMLSpyCommands object */
24    public void fillMenu(Menu newMenu, XMLSpyCommands xmlSpyMenu) throws AutomationException
25  {
26   // For each command/submenu in the xmlSpyMenu
27   for (int i=0;i<xmlSpyMenu.getCount();++i)
28   {
29    XMLSpyCommand xmlSpyCommand = xmlSpyMenu.getItem(i);
30    if (xmlSpyCommand.getIsSeparator())
31     newMenu.addSeparator();
32    else
33    {
34     XMLSpyCommands subCommands = xmlSpyCommand.getSubCommands();
35     // Is it a command (leaf), or a submenu?
36     if (subCommands.isNull() || subCommands.getCount()==0)
37     {
38      // Command -> add it to the menu, set its ActionCommand to its ID and store it in the menuMap
39      MenuItem mi = new MenuItem(xmlSpyCommand.getLabel().replace("&", ""));
40      mi.setActionCommand("" + xmlSpyCommand.getID());
41      mi.addActionListener(this);
42      newMenu.add(mi);
43      menuMap.put(xmlSpyCommand.getID(), mi);
44     }
45     else
46     {
47      // Submenu -> create submenu and repeat recursively
48      Menu newSubMenu = new Menu();
49      fillMenu(newSubMenu, subCommands);
50      newSubMenu.setLabel(xmlSpyCommand.getLabel().replace("&", ""));
51      newMenu.add(newSubMenu);
52     }
53    }
54   }
55  }
56
57   /**
58   * Action handler for the menu items
59   * Called when the user selects a menu item; the item's action command corresponds to the command table for
XMLSpy
60   */
61  public void actionPerformed(ActionEvent e)
62  {
63   try
64   {
65    int iCmd = Integer.parseInt(e.getActionCommand());
66    // Handle explicitly the Close commands
```

```
67     switch (iCmd)
68     {
69      case 57602:    // Close
70      case 34050:    // Close All
71       XMLSpyContainer.initXmlSpyDocument();
72       break;
73      default:
74       XMLSpyContainer.xmlSpyControl.exec(iCmd);
75       break;
76     }
77    }
78    catch (Exception ex)
79    {
80     ex.printStackTrace();
81    }
82
83   }
```

UI Update Event Handling

The code listing below shows how a UI-Update event handler can be created.

```
01 /**
02   * Call-back from the XMLSpyControl.
03   * Called to enable/disable commands
04   */
05  @Override
06  public void onUpdateCmdUI() throws AutomationException
07  {
08   // A command should be enabled if the result of queryStatus contains the Supported (1) and Enabled (2) flags
09   for (java.util.Map.Entry<Integer, MenuItem> pair : menuMap.entrySet())
10    pair.getValue().setEnabled(XMLSpyContainer.xmlSpyControl.queryStatus(pair.getKey()) > 2);
11  }
12   /**
13   * Call-back from the XMLSpyControl.
14   * Usually called while enabling/disabling commands due to UI updates
15   */
16  @Override
17  public boolean onIsActiveEditor(String istrFilePath) throws AutomationException
18  {
19   try {
20    return XMLSpyContainer.xmlSpyDocument.getDocument().getFullName().equalsIgnoreCase(istrFilePath);
21   }catch (Exception e) {
22    return false;
23   }
24  }
```

Creating an XML Tree

The listing below loads an XML data object as nodes in a tree.

```
01 // access required XMLSpy Java-COM classes
02 import com.altova.automation.XMLSpy.XMLData;
03
04 // access AWT and Swing components
05 import java.awt.*;
06 import javax.swing.*;
07 import javax.swing.tree.*;
08
09 /**
10 * A simple example of a tree control loading the structure from an XMLData object.
11 * The class receives an XMLData object, loads its nodes in a JTree, and prepares
12 * for modal activation.
13 *
14 * Feel free to modify and extend this sample.
15 *
16 * @author Altova GmbH
17 */
18 class XMLTreeDialog extends JDialog
19 {
20  /**
21   * The tree control
22   */
```

```
23  private JTree myTree;
24
25  /**
26   * Root node of the tree control
27   */
28  private DefaultMutableTreeNode top;
29
30  /**
31   * Constructor that prepares the modal dialog containing the filled tree control
32   * @param xml The data to be displayed in the tree
33   * @param parent Parent frame
34   */
35  public XMLTreeDialog(XMLData xml, Frame parent)
36  {
37    // Construct the modal dialog
38    super(parent, "XML tree", true);
39    // Arrange controls in the dialog
40    top = new DefaultMutableTreeNode("root");
41    myTree = new JTree(top);
42    setContentPane(new JScrollPane(myTree));
43    // Build up the tree
44    fillTree(top, xml);
45    myTree.expandRow(0);
46  }
47
48  /**
49   * Loads the nodes of an XML element under a given tree node
50   * @param node Target tree node
51   * @param elem Source XML element
52   */
53  private void fillTree(DefaultMutableTreeNode node, XMLData elem)
54  {
55    try
56    {
57      // There are several ways to iterate through child elements: either using the getFirstChild/getNextChild,
58      // or by incrementing an index up to countChildren and calling getChild [as shown below].
59      // If you only want to get children of one kind, you should use countChildrenKind/getChildKind,
60      // or provide a kind to the getFirstChild before iterating with the getNextChild.
61      int nSize = elem.countChildren();
62      for (int i=0; i<nSize; ++i)
63      {
64        // Create a new tree node for each child element, and continue recursively
65        XMLData newElem = elem.getChild(i);
66        DefaultMutableTreeNode newNode = new DefaultMutableTreeNode(newElem.getName());
67        node.add(newNode);
68        fillTree(newNode, newElem);
69      }
70    }
71    catch (Exception e)
72    {
73      e.printStackTrace();
74    }
75  }
76
77 }
```

4.6 Command Reference

This section lists the names and identifiers of all menu commands that are available within XMLSpy. Every sub-section lists the commands from the corresponding top-level menu of XMLSpy. The command tables are organized as follows:

- The "Menu Item" column shows the command's menu text as it appears in XMLSpy, to make it easier for you to identify the functionality behind the command.
- The "Command Name" column specifies the string that can be used to get an icon with the same name from **ActiveX\Images** folder of the XMLSpy installation directory.
- The "ID" column shows the numeric identifier of the column that must be supplied as argument to methods which execute or query this command.

To execute a command, use the `XMLSpyControl.Exec` or the `XMLSpyControlDocument.Exec` methods. To query the status of a command, use the `XMLSpyControl.QueryStatus` or `XMLSpyControlDocument.QueryStatus` methods.

Depending on the edition of XMLSpy you have installed, some of these commands might not be supported.

4.6.1 "File" Menu

The "File" menu has the following commands:

Menu item	Command name	ID
New...	ID_FILE_NEW	57600
Open...	ID_FILE_OPEN	57601
Reload	IDC_FILE_RELOAD	34065
Encoding...	IDC_ENCODING	34061
Close	ID_FILE_CLOSE	57602
Close All	IDC_CLOSE_ALL	34050
Close All But Active	IDC_CLOSE_OTHERS	34271
Save	ID_FILE_SAVE	57603
Save As...	ID_FILE_SAVE_AS	57604
Save All	ID_FILE_SAVE_ALL	34208
Send by Mail...	ID_FILE_SEND_MAIL	57612
Print...	ID_FILE_PRINT	57607
Print Preview	IDC_PRINT_PREVIEW	34104
Print Setup...	ID_FILE_PRINT_SETUP	57606
Recent File	ID_FILE_MRU_FILE1	57616
Exit	ID_APP_EXIT	57665

4.6.2 "Edit" Menu

The "Edit" menu has the following commands:

Menu item	Command name	ID
Undo	ID_EDIT_UNDO	57643
Redo	ID_EDIT_REDO	57644
Cut	ID_EDIT_CUT	57635
Copy	ID_EDIT_COPY	57634
Paste	ID_EDIT_PASTE	57637
Delete	ID_EDIT_CLEAR	57632
Copy as XML Text	IDC_COPY_AS_XML_TEXT	33443
Copy as Structured Text	IDC_COPY_AS_STRUCTURED_TEXT	33442
Copy XPath	IDC_COPY_XPATH	33444
Copy XPointer	IDC_COPY_XPOINTER	33445
File Path...	IDC_EDIT_INSERT_PATH_STRING	34013
XInclude...	IDC_EDIT_INSERT_XINCLUDE_STRING	34017
Encoded External File...	IDC_EDIT_INSERT_ENCODED_BINARY_STRING	34273
Pretty-Print	IDC_PRETTY_PRINT	34101
Strip Whitespaces	IDC_STRIP_WHITESPACES	34296
Select All	ID_EDIT_SELECT_ALL	57642
Find...	ID_EDIT_FIND	57636
Find Next	ID_EDIT_REPEAT	57640
Replace...	ID_EDIT_REPLACE	57641
Find in Files...	IDC_FIND_IN_FILES	34000
Insert/Remove Bookmark	IDC_TOGGLE_BOOKMARK	34162
Remove All Bookmarks	IDC_REMOVEALLBOOKMARKS	34132
Go to Next Bookmark	IDC_GOTONEXTBOOKMARK	34070
Go to Previous Bookmark	IDC_GOTOPREVBOOKMARK	34071
Comment In/Out	IDC_TOGGLE_XML_COMMENT	34029

4.6.3 "Project" Menu

The "Project" menu has the following commands:

Menu item	Command name	ID
New Project	IDC_ICPROJECTGUI_NEW	37200
Open Project...	IDC_ICPROJECTGUI_OPEN	37201
Reload Project	IDC_ICPROJECTGUI_RELOAD	37202
Close Project	IDC_ICPROJECTGUI_CLOSE	37203
Save Project	IDC_ICPROJECTGUI_SAVE	37204
Save Project As...	IDC_ICPROJECTGUI_SAVE_AS	37207
Enable Source Control	ID_SCC_ENABLE	38602
Add Files to Project...	IDC_ICPROJECTGUI_ADD_FILES_TO_PROJECT	37205
Add Global Resource to Project...	IDC_ICPROJECTGUI_ADD_GLOBAL_RESOURCE_TO_PROJECT	37239
Add URL to Project...	IDC_ICPROJECTGUI_ADD_URL_TO_PROJECT	37206
Add Active File to Project	IDC_ICPROJECTGUI_ADD_ACTIVE_FILE_TO_PROJECT	37208
Add Active and Related Files to Project	IDC_ICPROJECTGUI_ADD_ACTIVE_AND_RELATED_FILES_TO_PROJECT	37209
Add Project Folder to Project...	IDC_ICPROJECTGUI_ADD_FOLDER_TO_PROJECT	37210
Add External Folder to Project...	IDC_ICPROJECTGUI_ADD_EXT_FOLDER_TO_PROJECT	37211
Add External Web Folder to Project...	IDC_ICPROJECTGUI_ADD_EXT_URL_FOLDER_TO_PROJECT	37212
Script settings...	IDC_PROJECT_SCRIPT_SETTINGS	34136
Properties...	IDC_ICPROJECTGUI_PROJECT_PROPERTIES	37223
Recent Project	IDC_ICPROJECTGUI_RECENT	37224

4.6.4 "XML" Menu

The "XML" menu has the following commands:

Menu item	Command name	ID
Attribute	IDC_INSERT_ATTRIBUTE	33449
Element	IDC_INSERT_STRUCT	33459
Text	IDC_INSERT_TEXT	33460
CDATA	IDC_INSERT_CDATA	33450
Comment	IDC_INSERT_COMMENT	33451

Menu item	Command name	ID
XML	IDC_INSERT_XML	33461
Processing Instruction	IDC_INSERT_PI	33458
XInclude...	IDC_INSERT_XINCLUDE	34019
DOCTYPE	IDC_INSERT_DEF_DOCTYPE	33453
ExternalID	IDC_INSERT_DEF_EXTERNAL_ID	33456
ELEMENT	IDC_INSERT_DEF_ELEMENT	33454
ATTLIST	IDC_INSERT_DEF_ATTLIST	33452
ENTITY	IDC_INSERT_DEF_ENTITY	33455
NOTATION	IDC_INSERT_DEF_NOTATION	33457
Encoded External File...	IDC_INSERT_ENCODED_BINARY	34274
Attribute	IDC_APPEND_ATTRIBUTE	33415
Element	IDC_APPEND_STRUCT	33425
Text	IDC_APPEND_TEXT	33426
CDATA	IDC_APPEND_CDATA	33416
Comment	IDC_APPEND_COMMENT	33417
XML	IDC_APPEND_XML	33427
Processing Instruction	IDC_APPEND_PI	33424
XInclude...	IDC_APPEND_XINCLUDE	34026
DOCTYPE	IDC_APPEND_DEF_DOCTYPE	33419
ExternalID	IDC_APPEND_DEF_EXTERNAL_ID	33422
ELEMENT	IDC_APPEND_DEF_ELEMENT	33420
ATTLIST	IDC_APPEND_DEF_ATTLIST	33418
ENTITY	IDC_APPEND_DEF_ENTITY	33421
NOTATION	IDC_APPEND_DEF_NOTATION	33423
Encoded External File...	IDC_APPEND_ENCODED_BINARY	34276
Attribute	IDC_ADD_CHILD_ATTRIBUTE	33402
Element	IDC_ADD_CHILD_STRUCT	33412
Text	IDC_ADD_CHILD_TEXT	33413
CDATA	IDC_ADD_CHILD_CDATA	33403
Comment	IDC_ADD_CHILD_COMMENT	33404
XML	IDC_ADD_CHILD_XML	33414
Processing Instruction	IDC_ADD_CHILD_PI	33411
XInclude...	IDC_ADD_CHILD_XINCLUDE	34027

Menu item	Command name	ID
DOCTYPE	IDC_ADD_CHILD_DEF_DOCTYPE	33406
ExternalID	IDC_ADD_CHILD_DEF_EXTERNAL_ID	33409
ELEMENT	IDC_ADD_CHILD_DEF_ELEMENT	33407
ATTLIST	IDC_ADD_CHILD_DEF_ATTLIST	33405
ENTITY	IDC_ADD_CHILD_DEF_ENTITY	33408
NOTATION	IDC_ADD_CHILD_DEF_NOTATION	33410
Encoded External File...	IDC_ADD_CHILD_ENCODED_BINARY	34277
Attribute	IDC_CONVERT_TO_ATTRIBUTE	33429
Element	IDC_CONVERT_TO_STRUCT	33439
Text	IDC_CONVERT_TO_TEXT	33440
CDATA	IDC_CONVERT_TO_CDATA	33430
Comment	IDC_CONVERT_TO_COMMENT	33431
XML	IDC_CONVERT_TO_XML	33441
Processing Instruction	IDC_CONVERT_TO_PI	33438
DOCTYPE	IDC_CONVERT_TO_DEF_DOCTYPE	33433
ExternalID	IDC_CONVERT_TO_DEF_EXTERNAL_ID	33436
ELEMENT	IDC_CONVERT_TO_DEF_ELEMENT	33434
ATTLIST	IDC_CONVERT_TO_DEF_ATTLIST	33432
ENTITY	IDC_CONVERT_TO_DEF_ENTITY	33435
NOTATION	IDC_CONVERT_TO_DEF_NOTATION	33437
Display as Table	IDC_GRID_VIEW_AS_TABLE	34075
Insert Row	IDC_TABLE_INSERT_ROW	34158
Append Row	IDC_TABLE_APPEND_ROW	34157
Ascending Sort	IDC_TABLE_SORT_ASC	33464
Descending Sort	IDC_TABLE_SORT_DESC	33465
Move Left	IDC_MOVE_LEFT	34091
Move Right	IDC_MOVE_RIGHT	34092
Enclose in Element	IDC_ENCLOSE_IN_ELEMENT	33446
Evaluate XPath...	IDC_EVALUATE_XPATH	34007
Check Well-Formedness	IDC_CHECK_WELL_FORM	34049
Validate XML	IDC_VALIDATE	32954
Validate XML on Server (high-performance)	IDC_VALIDATE_RAPTOR	34309

Menu item	Command name	ID
Update Entry Helpers	IDC_UPDATE_ELEMENT_CHOICE	34173
Namespace Prefix...	IDC_NAMESPACE	33462
Create XML Signature...	IDC_XML_SIGNATURE_CREATE	34280
Verify XML Signature...	IDC_XML_SIGNATURE_VERIFY	34281

4.6.5 "DTD/ Schema" Menu

The "DTD/Schema" menu has the following commands:

Menu item	Command name	ID
Assign DTD...	IDC_ASSIGN_DTD	34032
Assign Schema...	IDC_ASSIGN_SCHEMA	34033
Include Another DTD...	IDC_INCLUDE_DTD	34084
Go to DTD	IDC_GOTO_DTD	34072
Go to Schema	IDC_GOTO_SCHEMA	34074
Go to Definition	IDC_GOTO_DEFINITION	33447
Generate DTD/Schema...	IDC_GENERATE_DTD_SCHEMA	34068
Flatten DTD...	IDC_FLATTEN_DTD	34301
Convert DTD To Schema...	IDC_CONVERT_DTD_TO_SCHEMA	34299
Flatten Schema...	IDC_FLATTEN_SCHEMA	34302
Convert Schema To DTD...	IDC_CONVERT_SCHEMA_TO_DTD	34300
Convert to UML...	IDC_CONVERT_SCHEMA_TO_UML	34008
Generate XML from DB, Excel, EDI with MapForce...	IDC_DTD_OPENIN_MAPFORCE	34056
Design HTML/PDF/Word Output with StyleVision...	IDC_DTD_OPENIN_STYLEVISION	34057
Generate Sample XML/JSON File...	IDC_GENERATE_XML_FROM_SCHEMA	34069
Generate Program Code...	IDC_GENERATE_CODE_FROM_SCHEMA	34067
Flush Memory Cache	IDC_FLUSH_CACHED_FILES	34066

4.6.6 "Schema design" Menu

The "Schema design" menu has the following commands:

Menu item	Command name	ID
Schema Settings...	IDC_SCHEMA_NAMESPACES	33571

Menu item	Command name	ID
Save Diagram...	IDC_SCHEMA_SAVE_DIAGRAM	33581
Generate Documentation...	IDC_SCHEMA_DOCUMENTATION	34146
Configure View...	IDC_SCHEMA_VIEW_CONFIG	33593
Zoom...	IDC_SCHEMA_ZOOM	34150
Display All Globals	IDC_SCHEMA_MODE_GLOBALS	34147
Display Diagram	IDC_SCHEMA_MODE_DIAGRAM	33570
Enable Oracle Schema Extensions	IDC_SCHEMA_ORACLE_EXTENSIONS	33577
Oracle Schema Settings...	IDC_SCHEMA_ORACLE_SCHEMA_SETTINGS	33578
Enable Microsoft SQL Server Schema Extensions	IDC_SCHEMA_SQLSERVER_EXTENSIONS	33588
Named Schema Relationships...	IDC_SCHEMA_SQLSERVER_GLOBAL_RELATIONSHIPS	33589
Unnamed Element Relationships...	IDC_SCHEMA_SQLSERVER_LOCAL_RELATIONSHIPS	33590
Connect to SchemaAgent Server...	IDC_SCHEMA_SCHEMAAGENT_SERVER_CONNECT	33582
Disconnect from SchemaAgent Server	IDC_SCHEMA_SCHEMAAGENT_SERVER_DISCONNECT	33583
File Only	IDC_SCHEMAAGENT_SHOW_FILE_ONLY	33504
File and All Directly Referenced Schema Files	IDC_SCHEMAAGENT_SHOW_WITH_DIRECTLY_REFERENCED_SCHEMAS	33608
File and All Directly Referencing Schema Files	IDC_SCHEMAAGENT_SHOW_WITH_DIRECTLY_REFERENCING_SCHEMAS	33602
File and All Directly Related Schema Files	IDC_SCHEMAAGENT_SHOW_WITH_DIRECTLY_RELATED_SCHEMAS	33613
SchemaAgent Validation...	IDC_SCHEMA_EXTVALID_MENU	33539
Create Schema Subset...	IDC_SCHEMA_CREATE_SUBSET	33650
Flatten Schema...	IDC_SCHEMA_FLATTEN	33651

4.6.7 "XSL/ XQuery" Menu

The "XSL/XQuery" menu has the following commands:

Menu item	Command name	ID
XSL Transformation	IDC_TRANSFORM_XSL	33006
XSL Speed Optimizer	IDC_TRANSFORM_XSLPBO	34306
XSL-FO Transformation	IDC_TRANSFORM_XSLFO	33007

Menu item	Command name	ID
XSL Parameters / XQuery Variables...	IDC_TRANSFORM_XSL_PARAMS	33008
XQuery/Update Execution	IDC_TRANSFORM_XQUERY	34170
Enable Back Mapping	IDC_ENABLE_BACKMAPPING	34364
Enable XSLT/ XQuery Profiling....	IDC_PROFILING_OPTIONS	34105
Assign XSL...	IDC_ASSIGN_XSL	33001
Assign XSL-FO...	IDC_ASSIGN_XSLFO	33002
Assign Sample XML File...	IDC_ASSIGN_SAMPLE_XML	33000
Go to XSL	IDC_GOTO_XSL	33004
Start Debugger / Go	ID_PROCESS_XSL	34212
Stop Debugger	ID_XSLT_DEBUGGER_STOP	33017
Restart Debugger	ID_XSLT_DEBUGGER_RESTART	33013
End Debugger Session	ID_XSLT_DEBUGGER_END_SESSION	33011
Step Into	ID_XSLT_DEBUGGER_STEP	33014
Step Out	ID_XSLT_DEBUGGER_STEP_OUT	33015
Step Over	ID_XSLT_DEBUGGER_STEP_OVER	33016
Show Current Execution Node	ID_XSLT_DEBUGGER_GO_TO_CURRENT_EXECUTION_NODES	33012
Insert/Remove Breakpoint	IDC_TOGGLE_BREAKPOINT	34246
Insert/Remove Tracepoint	IDC_TOGGLE_TRACEPOINT	34248
Enable/Disable Breakpoint	IDC_ENABLE_BREAKPOINT	34245
Enable/Disable Tracepoint	IDC_ENABLE_TRACEPOINT	34247
Breakpoints/Tracepoints...	ID_XSLTDEBUGGER_BREAKPOINTS	33009
Call Stack	ID_XSL_DEBUGWINDOWS_CALLSTACK	34238
XPath-Watch	ID_XSL_DEBUGWINDOWS_WATCH	34244
Context	ID_XSL_DEBUGWINDOWS_CONTEXT	34239
Variables	ID_XSL_DEBUGWINDOWS_VARIABLE	34243
Messages	ID_XSL_DEBUGWINDOWS_MESSAGES	34240
Templates	ID_XSL_DEBUGWINDOWS_TEMPLATES	34241
Info	ID_XSLXQUERY_DEBUGWINDOWS_INFO	34237
Trace	ID_XSL_DEBUGWINDOWS_TRACES	34242
Debug Settings...	ID_XSLTDEBUGGER_SETTINGS	33010

4.6.8 "Authentic" Menu

The "Authentic" menu has the following commands:

Menu item	Command name	ID
New Document...	IDC_AUTHENTIC_NEW_FILE	34036
Edit Database Data...	IDC_AUTHENTIC_EDIT_DB	34035
Assign a StyleVision Stylesheet...	IDC_ASSIGN_SPS	34034
Edit StyleVision Stylesheet	IDC_EDIT_SPS	34060
Select New Row with XML Data for Editing...	IDC_CHANGE_WORKING_DB_XML_CELL	32861
XML Signature...	IDC_AUTHENTICGUI_XMLSIGNATURE	32862
Define XML Entities...	IDC_DEFINE_ENTITIES	32805
Hide Markup	IDC_MARKUP_HIDE	32855
Show Small Markup	IDC_MARKUP_SMALL	32858
Show Large Markup	IDC_MARKUP_LARGE	32856
Show Mixed Markup	IDC_MARKUP_MIXED	32857
Toggle Bold	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEBOLD	32813
Toggle Italic	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEITALIC	32814
Toggle Underline	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEUNDERLINE	32815
Toggle Strikethrough	IDC_AUTHENTICGUI_RICHEDIT_TOGGLESTRIKETHROUGH	32816
Foreground Color	IDC_AUTHENTICGUI_RICHEDIT_COLOR_FOREGROUND	32824
Background Color	IDC_AUTHENTICGUI_RICHEDIT_COLOR_BACKGROUND	32830
Align Left	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_LEFT	32818
Center	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_CENTER	32819
Align Right	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_RIGHT	32820
Append Row	IDC_ROW_APPEND	32806
Insert Row	IDC_ROW_INSERT	32809
Duplicate Row	IDC_ROW_DUPLICATE	32808
Move Row Up	IDC_ROW_MOVE_UP	32811

Menu item	Command name	ID
Move Row Down	IDC_ROW_MOVE_DOWN	32810
Delete Row	IDC_ROW_DELETE	32807
Generate an HTML document	IDC_PXF_GENERATE_HTML	34283
Generate an RTF document	IDC_PXF_GENERATE_RTF	34284
Generate a PDF document	IDC_PXF_GENERATE_PDF	34285
Generate a Word 2007+ document	IDC_PXF_GENERATE_DOCX	34286
Trusted Locations...	IDC_TRUSTED_LOCATIONS	34288

4.6.9 "DB" Menu

The "DB" menu has the following commands:

Menu item	Command name	ID
Query Database	IDC_QUERYDATABASE	34012
Manage XML Schemas...	IDC_DB_MANAGESCHEMAS	34014
Assign XML Schema...	IDC_DB_CHOOSEVALIDATIONSCHEMA	34016
Manage XML Schemas...	IDC_DB_MANAGESCHEMAS	34014
Manage XML Schemas...	IDC_DB_MANAGESCHEMAS	34014
Browse Oracle XML Documents...	ID_CONVERT_ORACLEXMLDB_BROWSE	34205

4.6.10 "Convert" Menu

The "Convert" menu has the following commands:

Menu item	Command name	ID
Import Text File...	IDC_IMPORT_TEXT	34082
Import Database Data...	IDC_IMPORT_DATABASE	34080
Import Microsoft Word Document...	IDC_IMPORT_WORD	34083
Create XML Schema from DB Structure	IDC_CREATE_DB_SCHEMA	34054
DB Import Based on XML Schema	IDC_IMPORT_DB_SCHEMA	34081
Create DB Structure from XML Schema	IDC_CREATE_DB_BASED_ON_SCHEMA	34053
Export to Text Files...	IDC_EXPORT_TEXTFILE	34064
Export to a Database...	IDC_EXPORT_DB	34003
Convert XML Instance to/from JSON...	IDC_JSON_CONVERT_TOFROM_XML	34135
Convert XML Schema to/from JSON Schema...	IDC_JSON_CONVERT_TOFROM_XSD	34350

4.6.11 "View" Menu

The "View" menu has the following commands:

Menu item	Command name	ID
Text View	IDC_VIEW_TEXT	34180
Enhanced Grid View	IDC_VIEW_GRID	34178
Schema Design View	IDC_VIEW_SCHEMA	34179
WSDL Design View	IDC_VIEW_WSDL	34117
XBRL Taxonomy View	IDC_VIEW_XBRL	34118
Authentic View	IDC_VIEW_CONTENT	34177
Browser View	IDC_VIEW_BROWSER	34176
Expand +	IDC_SEL_EXPAND	34152
Collapse -	IDC_SEL_COLLAPSE	34151
Expand Fully	IDC_SEL_EXPAND_ALL	33463
Collapse Unselected	IDC_COLLAPSE_UNSELECTED	33428
Optimal Widths	IDC_OPTIMAL_WIDTHS	34099
Word Wrap	IDC_WORD_WRAP	34181
Go to Line/Character	IDC_GOTO_LINE	34073
Go to File	IDC_GOTO_FILE	33448
Text View Settings	IDC_TEXTVIEW_SETTINGS	34119

4.6.12 "Browser" Menu

The "Browser" menu has the following commands:

Menu item	Command name	ID
Back	IDC_STEP_BACK	32958
Forward	IDC_STEP_FORWARD	32957
Stop	IDC_BROWSER_STOP	34047
Refresh	IDC_BROWSER_REFRESH	34046
Largest	IDC_BROWSER_FONT_LARGEST	34041
Larger	IDC_BROWSER_FONT_LARGE	34040
Medium	IDC_BROWSER_FONT_MEDIUM	34042
Smaller	IDC_BROWSER_FONT_SMALL	34043
Smallest	IDC_BROWSER_FONT_SMALLEST	34044

4.6.13 "WSDL" Menu

The "WSDL" menu has the following commands:

Menu item	Command name	ID
Insert Message	ID_WSDL_MESSAGES_ADDNEWMESSAGE	33715
Delete Message	ID_WSDL_MESSAGES_DELETESELECTEDMESSAGE	33717
Add Message Part (Parameter)	ID_WSDL_MESSAGES_ADDMESSAGEPART	33714
Delete Message Part (Parameter)	ID_WSDL_MESSAGES_DELETEMESSAGEPART	33716
request-response	IDC_WSDL_OPERATION_APPENDREQUESTRESPONSE	33734
solicit-response	IDC_WSDL_OPERATION_APPENDSOLICITRESPONSE	33737
one-way	IDC_WSDL_OPERATION_APPENDONEWAY	33735
notification	IDC_WSDL_OPERATION_APPENDNOTIFICATION	33736
Empty Operation	ID_WSDL_OPERATIONS_APPENDAOPERATIONTOTHISPORTTYPE	33722
Delete Operation	ID_WSDL_OPERATIONS_DELETEOPERATION	33724
Add Input Element	ID_WSDL_OPERATIONS_ADDINPUTFUNCTION	33719
Add Output Element	ID_WSDL_OPERATIONS_ADDOUTPUTFUNCTION	33721
Add Fault Element	ID_WSDL_OPERATIONS_ADDFAULTFUNCTION	33718
Delete Input/Output/Fault Element	ID_WSDL_OPERATIONS_DELETEINPUTOUTPUTFUNCTION	33723
Add New Message to Input/Output/Fault Element	ID_WSDL_OPERATIONS_ADDNEWMESSAGETOTHISELEMENT	33720
Insert Port Type	ID_WSDL_PORTTYPE_INSERTAPORTTYPE	33727
Delete Port Type	ID_WSDL_PORTTYPE_DELETETHISPORTTYPE	33726
Insert Binding	ID_WSDL_BINDING_NEWBINDING	33713
Delete Binding	ID_WSDL_BINDING_DELETEBINDING	33711
soap:body	ID_WSDL_BINDING_APPENDEXTENSI	33706

Menu item	Command name	ID
	BILITY_SOAPBODY	
soap:header	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPHEADER	33708
soap:headerfault	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPHEADERFAULT	33709
soap:fault	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPFAULT	33707
mime:content	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMECONTENT	33702
mime:multipartrelated	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMEMULTIPARTRELATED	33704
mime:part	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMEPART	33705
mime:mimeXml	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMEMIMEXML	33703
http:urlencoded	ID_WSDL_BINDING_APPENDEXTENSIBILITY_HTTPURLENCODED	33700
http:urlreplacement	ID_WSDL_BINDING_APPENDEXTENSIBILITY_HTTPURLREPLACEMENT	33701
Delete Extensibility Element	ID_WSDL_BINDING_DELETEEXTESIBILITY	33712
Insert Service	ID_WSDL_SERVICE_INSERTSERVICE	33731
Delete Service	ID_WSDL_SERVICE_DELETETHISSERVICE	33729
Insert Port	ID_WSDL_SERVICE_INSERTNEWPORT	33730
Delete Port	ID_WSDL_SERVICE_DELETETHISPORT	33728
Add New Interface	IDC_WSDL20_ADDINTERFACE	33794
Delete Interface	IDC_WSDL20_DELETEINTERFACE	33795
Add New Fault	IDC_WSDL20_ADDINTERFACEFAULT	33796
Delete Fault	IDC_WSDL20_DELETEINTERFACEFAULT	33808
In-only	IDC_WSDL20_ADDINTERFACEOPERATION_INONLY	33797
Robust-in-only	IDC_WSDL20_ADDINTERFACEOPERATION_ROBUSTINONLY	33798
In-out	IDC_WSDL20_ADDINTERFACEOPERATION_INOUT	33801

Menu item	Command name	ID
In-opt-out	IDC_WSDL20_ADDINTERFACEOPERATION_INOPTOUT	33802
Out-in	IDC_WSDL20_ADDINTERFACEOPERATION_OUTIN	33803
Out-opt-in	IDC_WSDL20_ADDINTERFACEOPERATION_OUTOPTIN	33804
Out-only	IDC_WSDL20_ADDINTERFACEOPERATION_OUTONLY	33800
Robust-out-only	IDC_WSDL20_ADDINTERFACEOPERATION_ROBUSTOUTONLY	33799
Empty Operation	IDC_WSDL20_ADDINTERFACEOPERATION_EMPTY	33805
Delete Operation	IDC_WSDL20_DELETEINTERFACEOPERATION	33809
Add New Binding	IDC_WSDL20_ADDBINDING	33820
Delete Binding	IDC_WSDL20_DELETEBINDING	33821
Add New Fault	IDC_WSDL20_ADDBINDINGFAULT	33822
Delete Fault	IDC_WSDL20_DELETEBINDINGFAULT	33826
Add New Operation	IDC_WSDL20_ADDBINDINGOPERATION	33823
Delete Operation	IDC_WSDL20_DELETEBINDINGOPERATION	33827
Add New Service	IDC_WSDL20_ADDSERVICE	33839
Delete Service	IDC_WSDL20_DELETESERVICE	33840
Add New Endpoint	IDC_WSDL20_ADDENDPOINT	33841
Delete Endpoint	IDC_WSDL20_DELETEENDPOINT	33842
New Schema	ID_WSDL_TYPES_NEWSCHEMA	33733
Embed Schema	ID_WSDL_TYPES_EMBEDSCHEMA	39456
Extract Schema(s)	ID_WSDL_TYPES_EXTRACTSCHEMAS	39459
Edit Schema(s) in Schema View	ID_WSDL_TYPES_EDITTHISSCHEMA	33732
Save Diagram...	IDC_WSDL_SAVE_DIAGRAM	39451
Generate Documentation...	ID_WSDL_GENERATEDOCUMENTATION	39452
Reparse WSDL Document	IDC_WSDL_REPARSE	33774
Convert to WSDL 2.0	IDC_WSDL_CONVERT_TO_WSDL20	39453
Generate WSDL Program Code with MapForce...	IDC_WSDL_GENERATE_CODE_MAPFORCE	34122

4.6.14 "SOAP" Menu

The "SOAP" menu has the following commands:

Menu item	Command name	ID
Create New SOAP Request...	ID_SOAP_GENERATESOAPMESSAGE	34224
Send Request to Server...	ID_SOAP_SENDREQUESTTOSERVER	34225
SOAP Request Settings...	ID_SOAP_SOAPREQUESTSETTINGS	34227
Soap Debugger Session	ID_SOAP_SOAPDEBUGGER	34226
Go	ID_SOAPDEBUGGER_BUTTONPLAY	34221
Single Step	ID_SOAPDEBUGGER_SINGLESTEP	34222
Break on Next Request	ID_SOAPDEBUGGER_BREAKONNEXTREQUEST	34219
Break on Next Response	ID_SOAPDEBUGGER_BREAKONNEXTRESPONSE	34220
Stop the Proxy Server	ID_SOAPDEBUGGER_STOPSERVER	34223
Soap Debugger Options	ID_SOAPDEBUGGEROPTIONS	34218

4.6.15 "XBRL" Menu

The "XBRL" menu has the following commands:

Menu item	Command name	ID
Arcroles...	IDC_XMLSPYXBRLEDITOR_ARCROLES	34114
Linkroles...	IDC_XMLSPYXBRLEDITOR_LINKROLES	34115
Namespace Prefixes...	IDC_XMLSPYXBRLEDITOR_NAMESPACES	34116
Set Target Namespace...	IDC_XMLSPYXBRLEDITOR_SET_TARGETNAMESPACE	34039
Parameter Values...	IDC_ICXBRLEDITOR_PARAMETER_VALUES	38913
Import/Reference...	IDC_XMLSPYXBRLEDITOR_IMPORT_REFERENCE	34137
Find Component By Id...	IDC_ICXBRLEDITOR_FIND_COMPONENT_BY_ID	38893
Generate Documentation...	IDC_XMLSPYXBRLEDITOR_GENERATEDOCUMENTATION	34125
View Settings...	IDC_XMLSPYXBRLEDITOR_VIEWSETTINGS	34113
Generate XBRL from DB, Excel, CSV	IDC_XBRL_GENERATE_WITH_MAPFOR	34045

Menu item	Command name	ID
with MapForce...	CE	
Present XBRL as HTML/PDF/Word with StyleVision...	IDC_XBRL_PRESENT_WITH_STYLEVISION	34121
Execute Formula...	IDC_XBRL_EXECUTE_FORMULA	34305
Execute Formula on Server (high-performance)...	IDC_XBRL_EXECUTE_FORMULA_RAPTOR	34352
Generate Table...	IDC_XBRL_GENERATE_TABLE	34304
Generate Table on Server (high-performance)...	IDC_XBRL_GENERATE_TABLE_RAPTOR	34353
Transform Inline XBRL	IDC_IXBRL_TRANSFORM	34354

4.6.16 "Tools" Menu

The "Tools" menu has the following commands:

Menu item	Command name	ID
Spelling...	IDC_SPELL_CHECK	34154
Spelling Options...	IDC_SPELL_OPTIONS	34155
Scripting Editor...	ID_SCRIPTFORMEDITOR_EDIT_PROJECT	39666
none	ID_SCRIPTFORMEDITOR_EXECUTE_MACRO_MENU_UPPDATE	39600
Compare Open File With...	ID_XMLDIFF_CHOOSE_FILES	34235
Compare Directories...	ID_XMLDIFF_CHOOSE_DIRECTORIES	34234
Compare Options...	ID_XMLDIFF_SETTINGS	34236
	IDC_TOOLS_ENTRY	34292
Global Resources	IDC_GLOBALRESOURCES	37401
	IDC_GLOBALRESOURCES_SUBMENUENTRY1	37408
Manage Raptor Servers ...	IDC_VALIDATE_RAPTOR_MANAGER	34311
none	IDC_VALIDATE_RAPTOR_NOCFG	34326
Customize...	IDC_APP_TOOLS_CUSTOMIZE	32959
Options...	IDC_SETTINGS	34133
	ID_SCRIPTING_MACROITEMS	34249

4.6.17 "Window" Menu

The "Window" menu has the following commands:

Menu item	Command name	ID
Cascade	ID_WINDOW_CASCADE	57650
Tile horizontally	ID_WINDOW_TILE_HORZ	57651
Tile vertically	ID_WINDOW_TILE_VERT	57652
Project window	IDC_PROJECT_WINDOW	34128
Info window	IDC_INFO_WINDOW	34085
Entry Helpers	IDC_ENTRY_HELPERS	34062
Output windows	IDC_OUTPUT_DIALOGBARS	34004
Project and Entry Helpers	IDC_PROJECT_ENTRYHELPERS	34006
All on/off	IDC_ALL_BARS	34031

4.6.18 "Help" Menu

The "Help" menu has the following commands:

Menu item	Command name	ID
Table of Contents...	IDC_HELP_CONTENTS	34076
Index...	IDC_HELP_INDEX	34077
Search...	IDC_HELP_SEARCH	34079
Keyboard Map...	IDC_HELP_KEYMAPDLG	34078
Software Activation...	IDC_ACTIVATION	34005
Order Form...	IDC_OPEN_ORDER_PAGE	34094
Registration...	IDC_REGISTRATION	34131
Check for Updates...	IDC_CHECK_FOR_UPDATES	34275
Support Center...	IDC_OPEN_SUPPORT_PAGE	34096
FAQ on the Web...	IDC_SHOW_FAQ	34153
Download Components and Free Tools...	IDC_OPEN_COMPONENTS_PAGE	34093
XMLSpy on the Internet..	IDC_OPEN_XML_SPY_HOME	34098
XMLSpy Training...	ID_HELP_XMLSPYTRAINING	34210
About XMLSpy...	ID_APP_ABOUT	57664

4.7 Object Reference

Objects:
XMLSpyCommand
XMLSpyCommands
XMLSpyControl
XMLSpyControlDocument
XMLSpyControlPlaceHolder

To give access to standard XMLSpy functionality, objects of the **XMLSpy automation interface** can be accessed as well. See XMLSpyControl.Application, XMLSpyControlDocument.Document and XMLSpyControlPlaceHolder.Project for more information.

4.7.1 XMLSpyCommand

Properties:
ID
Label
Name
IsSeparator
ToolTip
StatusText
Accelerator
SubCommands

Description:
A command object can be one of the following: an executable command, a command container (for example, a menu, submenu, or toolbar), or a menu separator. To determine what kind of information is stored in the current `Command` object, query its `ID`, `IsSeparator`, and `SubCommands` properties, as follows.

The Command object is...	When...
An executable command	• `ID` is greater than zero • `IsSeparator` is false • `SubCommands` is empty
A command container	• `ID` is zero • `IsSeparator` is true • `SubCommands` contains a collection of `Command` objects.
Separator	• `ID` is zero • `IsSeparator` is true

Accelerator

Property: `Accelerator` as string

Description:
Returns the accelerator key defined for the command. If the command has no accelerator key assigned, this property returns the empty string. The string representation of the accelerator key has the following format:

```
[ALT+][CTRL+][SHIFT+]key
```

Where `key` is converted using the Windows Platform SDK function `GetKeyNameText`.

ID

Property: `ID` as long

Description:
This property gets the unique identifier of the command. A command's ID is required to execute the command (using `Exec`) or query its status (using `QueryStatus`). If the command is a container for other commands (for example, a top-level menu), or a separator, the ID is 0.

IsSeparator

Property: `IsSeparator` as boolean

Description:
The property returns `true` if the command object is a menu separator; `false` otherwise. See also Command.

Label

Property: `Label` as string

Description:
This property gets the text of the command as it is displayed in the graphical user interface of XMLSpy. If the command is a separator, "Label" is an empty string. This property may also return an empty string for some toolbar commands that do not have any GUI text associated with them.

Name

Property: `Name` as string

Description:
This property gets the unique name of the command. This value can be used to get the icon file of the command, where it is available. The available icon files can be found in the folder **<ApplicationFolder>\Examples\ActiveX\Images** of your XMLSpy installation.

StatusText

Property: `Label` as string

Description:
The status text is the text shown in the status bar of XMLSpy when the command is selected. It applies only to command objects that are not separators or containers of other commands; otherwise, the property is an empty string.

SubCommands

Property: `SubCommands` as `Commands`

Description:
The `SubCommands` property gets the collection of `Command` objects that are sub-commands of the current command. The property is applicable only to commands that are containers for other commands (menus, submenus, or toolbars). Such container commands have the `ID` set to 0, and the `IsSeparator` property set to `false`.

ToolTip

Property: ToolTip as string

Description:
This property gets the text that is shown as a tool-tip for each command. If the command does not have a tooltip text, the property returns an empty string.

4.7.2 XMLSpyCommands

Properties:
Count
Item

Description:
Collection of Command objects to get access to command labels and IDs of the XMLSpyControl. Those commands can be executed with the Exec method and their status can be queried with QueryStatus.

Count

Property: Count as long

Description:
Number of Command objects on this level of the collection.

Item

Property: Item (n as long) as Command

Description:
Gets the command with the index n in this collection. Index is 1-based.

4.7.3 XMLSpyControl

Properties:
IntegrationLevel
Appearance
Application
BorderStyle
CommandsList
CommandsStructure (deprecated)
EnableUserPrompts
MainMenu
Toolbars

Methods:
Open
Exec
QueryStatus

Events:
OnUpdateCmdUI
OnOpenedOrFocused
OnCloseEditingWindow

OnFileChangedAlert
OnContextChanged
OnDocumentOpened
OnValidationWindowUpdated

This object is a complete ActiveX control and should only be visible if the XMLSpy library is used in the Application Level mode.

CLSID:a258bba2-3835-4c16-8590-72b44f52c471
ProgID:AltovaXMLSpyControl

Properties

The following properties are defined:

IntegrationLevel
EnableUserPrompts
Appearance
BorderStyle

Command related properties:
CommandsList
MainMenu
Toolbars
CommandsStructure (deprecated)

Access to XMLSpyAPI:
Application

Appearance

Property: Appearance as short

Dispatch Id: *-520*

Description:
A value not equal to 0 displays a client edge around the control. Default value is 0.

Application

Property: Application as Application

Dispatch Id: *1*

Description:
The Application property gives access to the Application object of the complete XMLSpy automation server API. The property is read-only.

BorderStyle

Property: BorderStyle as short

Dispatch Id: *-504*

Description:

A value of `1` displays the control with a thin border. Default value is `0`.

CommandsList

Property: `CommandList` as `Commands` (read-only)

Dispatch Id: *1004*

Description:
This property returns a flat list of all commands defined available with XMLSpyControl. To get commands organized according to their menu structure, use `MainMenu`. To get toolbar commands, use `Toolbars`.

```
public void GetAllXmlSpyCommands()
{
    // Get all commands from the XMLSpy ActiveX control assigned to the
current form
    XMLSpyControlLib.XMLSpyCommands commands =
this.axXMLSpyControl1.CommandList;
    // Iterate through all commands
    for (int i = 0; i < commands.Count; i++)
    {
      // Get each command by index and output it to the console
      XMLSpyControlLib.XMLSpyCommand cmd = axXMLSpyControl1.CommandList[i];
      Console.WriteLine("{0} {1} {2}", cmd.ID, cmd.Name,
cmd.Label.Replace("&", ""));
    }
}
```

C# example

EnableUserPrompts

Property: `EnableUserPrompts` as boolean

Dispatch Id: *1006*

Description:
Setting this property to *false*, disables user prompts in the control. The default value is *true*.

IntegrationLevel

Property: `IntegrationLevel` as `ICActiveXIntegrationLevel`

Dispatch Id: *1000*

Description:
The `IntegrationLevel` property determines the operation mode of the control. See also Integration at Application Level and Integration at Document Level for more information.

Note: It is important to set this property immediately after the creation of the `XMLSpyControl` object.

MainMenu

Property: `MainMenu` as Command (read-only)

Dispatch Id: *1003*

Description:
This property provides information about the structure and commands available in the XMLSpyControl main menu, as a `Command` object. The `Command` object contains all available submenus of XMLSpy (for example "File", "Edit", "View" etc.). To access the submenu objects, use the `SubCommands` property of the `MainMenu` property. Each submenu is also a `Command` object. For each submenu, you can then further iterate through their `SubCommands` property in order to get their corresponding child commands and separators (this technique may be used, for example, to create the application menu programmatically). Note that some menu commands act as containers ("parents") for other menu commands, in which case they also have a `SubCommands` property. To get the structure of all menu commands programmatically, you will likely need to create a recursive function.

```
public void GetXmlSpyMenus()
{
    // Get the main menu from the XMLSpy ActiveX control assigned to the
current form
    XMLSpyControlLib.XMLSpyCommand mainMenu =
this.axXMLSpyControl1.MainMenu;

    // Loop through entries of the main menu (e.g. File, Edit, etc.)
    for (int i = 0; i < mainMenu.SubCommands.Count; i++)
    {
      XMLSpyControlLib.XMLSpyCommand menu = mainMenu.SubCommands[i];
      Console.WriteLine("{0} menu has {1} children items (including
separators)", menu.Label.Replace("&", ""), menu.SubCommands.Count);
    }
}
```

C# example

Toolbars

Property: `Toolbars` as Commands (read-only)

Dispatch Id: *1005*

Description:
This property provides information about the structure of XMLSpyControl toolbars, as a `Command` object. The `Command` object contains all available toolbars of XMLSpy. To access the toolbars, use the `SubCommands` property of the `Toolbars` property. Each toolbar is also a `Command` object. For each toolbar, you can then further iterate through their `SubCommands` property in order to get their commands (this technique may be used, for example, to create the application's toolbars programmatically).

```
public void GetXmlSpyToolbars()
{
    // Get the application toolbars from the StyleVision ActiveX control
assigned to the current form
    XMLSpyControlLib.XMLSpyCommands toolbars =
```

```
this.axXMLSpyControl1.Toolbars;

    // Iterate through all toolbars
    for (int i = 0; i < toolbars.Count; i++)
    {
      XMLSpyControlLib.XMLSpyCommand toolbar = toolbars[i];
      Console.WriteLine();
      Console.WriteLine("The toolbar \"{0}\" has the following commands:",
toolbar.Label);

      // Iterate through all commands of this toolbar
      for (int j = 0; j < toolbar.SubCommands.Count; j++)
      {
         XMLSpyControlLib.XMLSpyCommand cmd = toolbar.SubCommands[j];
         // Output only command objects that are not separators
         if (!cmd.IsSeparator)
         {
            Console.WriteLine("{0}, {1}, {2}", cmd.ID, cmd.Name,
cmd.Label.Replace("&", ""));
         }
      }
    }
}
```

C# example

Methods

The following methods are defined:

Open
Exec
QueryStatus

Exec

Method: Exec (nCmdID as long) as boolean

Dispatch Id: *6*

Description:
This method calls the XMLSpy command with the ID nCmdID. If the command can be executed, the method returns true. To get a list of all available commands, use CommandsList. To retrieve the status of any command, use QueryStatus.

Open

Method: Open (strFilePath as string) as boolean

Dispatch Id: *5*

Description:
The result of the method depends on the extension passed in the argument strFilePath. If the file extension is .sps, a new document is opened. If the file extension is .svp, the corresponding project is opened. If a different file extension is passed into the method, the control tries to load the file as a new component into the active document.

Do not use this method to load documents or projects when using the control in document-level

integration mode. Instead, use XMLSpyControlDocumentOpen and XMLSpyControlPlaceHolderOpenProject.

QueryStatus

Method: QueryStatus(nCmdID as long) as long

Dispatch Id: *7*

Description:
QueryStatus returns the enabled/disabled and checked/unchecked status of the command specified by nCmdID. The status is returned as a bit mask.

Bit	Value	Name	Meaning
0	1	Supported	Set if the command is supported.
1	2	Enabled	Set if the command is enabled (can be executed).
2	4	Checked	Set if the command is checked.

This means that if QueryStatus returns 0 the command ID is not recognized as a valid XMLSpy command. If QueryStatus returns a value of 1 or 5, the command is disabled.

Events

The XMLSpyControl ActiveX control provides the following connection point events:

OnUpdateCmdUI
OnOpenedOrFocused
OnCloseEditingWindow
OnFileChangedAlert
OnContextChanged

OnDocumentOpened
OnValidationWindowUpdated

OnCloseEditingWindow

Event: OnCloseEditingWindow (i_strFilePath as String) as boolean

Dispatch Id: *1002*

Description:
This event is triggered when XMLSpy needs to close an already open document. As an answer to this event, clients should close the editor window associated with *i_strFilePath*. Returning *true* from this event indicates that the client has closed the document. Clients can return *false* if no specific handling is required and XMLSpyControl should try to close the editor and destroy the associated document control.

OnContextChanged

Event: OnContextChanged(i_strContextName as String, ibActive as bool) as bool

Dispatch Id: *1004*

Description:
This event is triggered when XMLSpy activates or de-actives one of the following operational contexts:

- XSLT Profiling - `"XSLTProfiling"` is passed as the context name
- XSLT / XQuery debugging - `"DebuggingXSLT"` is passed as the context name
- SOAP debugging - `"DebuggingSOAP"` is passed as the context name (Enterprise edition only)

OnDocumentOpened

Event: `OnDocumentOpened(objDocument as Document)`

Dispatch Id: *1*

Description:
This event is triggered whenever a document is opened. The argument `objDocument` is a `Document` object from the XMLSpy automation interface and can be used to query for more details about the document, or perform additional operations. When integrating on document-level, it is often better to use the event `XMLSpyControlDocumentOnDocumentOpened` instead.

OnFileChangedAlert

Event: `OnFileChangedAlert(i_strFilePath as String) as bool`

Dispatch Id: *1001*

Description:
This event is triggered when a file loaded with XMLSpyControl is changed on the hard disk by another application. Clients should return true, if they handled the event, or false, if XMLSpy should handle it in its customary way, i.e. prompting the user for reload.

OnLicenseProblem

Event: `OnLicenseProblem (i_strLicenseProblemText as String)`

Dispatch Id: *1005*

Description:
This event is triggered when XMLSpyControl detects that no valid license is available for this control. In case of restricted user licenses this can happen some time after the control has been initialized. Integrators should use this event to disable access to this control's functionality. After returning from this event, the control will block access to its functionality (e.g. show empty windows in its controls and return errors on requests).

OnOpenedOrFocused

Event: `OnOpenedOrFocused(i_strFilePath as String, i_bOpenWithThisControl as bool)`

Dispatch Id: *1000*

Description:
When integrating at application level, this event informs clients that a document has been opened, or made active by XMLSpy.

When integrating at document level, this event instructs the client to open the file `i_strFilePath` in a document window. If the file is already open, the corresponding document window should be made the active window.

if `i_bOpenWithThisControl` is `true`, the document must be opened with XMLSpyControl, since internal access is required. Otherwise, the file can be opened with different editors.

OnToolWindowUpdated

Event: `OnToolWindowUpdated(pToolWnd as long )`

Dispatch Id: *1006*

Description:
This event is triggered when the tool window is updated.

OnUpdateCmdUI

Event: `OnUpdateCmdUI()`

Dispatch Id: *1003*

Description:
Called frequently to give integrators a good opportunity to check status of XMLSpy commands using `XMLSpyControlQueryStatus`. Do not perform long operations in this callback.

OnValidationWindowUpdated

Event: `OnValidationWindowUpdated()`

Dispatch Id: *3*

Description:
This event is triggered whenever the validation output window is updated with new information.

4.7.4 XMLSpyControlDocument

Properties:
Appearance
BorderStyle
Document
IsModified
Path
ReadOnly

Methods:
Exec
New
Open
QueryStatus
Reload
Save
SaveAs

Events:
OnDocumentOpened
OnDocumentClosed
OnModifiedFlagChanged
OnContextChanged
OnFileChangedAlert
OnActivate

If the XMLSpyControl is integrated in the Document Level mode each document is displayed in an own object of type `XMLSpyControlDocument`. The `XMLSpyControlDocument` contains only one document at the time but can be reused to display different files one after another.

This object is a complete ActiveX control.

```
CLSID:52A552E6-2AB8-4e3e-B545-BE998233DDA0
ProgID:AltovaXMLSpyControlDocument
```

Properties

The following properties are defined:

ReadOnly
IsModified
Path
Appearance
BorderStyle

Access to XMLSpyAPI:
Document

Appearance

Property: `Appearance` as short

Dispatch Id: *-520*

Description:
A value not equal to `0` displays a client edge around the document control. Default value is `0`.

BorderStyle

Property: `BorderStyle` as short

Dispatch Id: *-504*

Description:
A value of `1` displays the control with a thin border. Default value is `0`.

Document

Property: `Document` as Document

Dispatch Id: *1*

Description:

The Document property gives access to the Document object of the XMLSpy automation server API. This interface provides additional functionality which can be used with the document loaded in the control. The property is read-only.

IsModified

Property: IsModified as boolean (read-only)

Dispatch Id: *1006*

Description:
IsModified is *true* if the document content has changed since the last open, reload or save operation. It is *false*, otherwise.

Path

Property: Path as string

Dispatch Id: *1005*

Description:
Sets or gets the full path name of the document loaded into the control.

ReadOnly

Property: ReadOnly as boolean

Dispatch Id: *1007*

Description:
Using this property you can turn on and off the read-only mode of the document. If ReadOnly is true it is not possible to do any modifications.

Methods

The following methods are defined:

Document handling:
New
Open
Reload
Save
SaveAs

Command Handling:
Exec
QueryStatus

Exec

Method: `Exec` (`nCmdID` as long) as boolean

Dispatch Id: *8*

Description:
`Exec` calls the XMLSpy command with the ID `nCmdID`. If the command can be executed, the method returns `true`. This method should be called only if there is currently an active document available in the application.

To get commands organized according to their menu structure, use the `MainMenu` property of XMLSpyControl. To get toolbar commands, use the `Toolbars` property of the XMLSpyControl.

New

Method: `New` () as boolean

Dispatch Id: *1000*

Description:
This method initializes a new document inside the control..

Open

Method: `Open` (`strFileName` as string) as boolean

Dispatch Id: *1001*

Description:
`Open` loads the file `strFileName` as the new document into the control.

QueryStatus

Method: `QueryStatus` (`nCmdID` as long) as long

Dispatch Id: *9*

Description:
`QueryStatus` returns the enabled/disabled and checked/unchecked status of the command specified by `nCmdID`. The status is returned as a bit mask.

Bit	Value	Name	Meaning
0	1	Supported	Set if the command is supported.
1	2	Enabled	Set if the command is enabled (can be executed).
2	4	Checked	Set if the command is checked.

This means that if `QueryStatus` returns `0` the command ID is not recognized as a valid XMLSpy command. If `QueryStatus` returns a value of `1` or `5` the command is disabled. The client should call the `QueryStatus` method of the document control if there is currently an active document

available in the application.

Reload

Method: Rebad() as boolean

Dispatch Id: *1002*

Description:
Rebad updates the document content from the file system.

Save

Method: Save() as boolean

Dispatch Id: *1003*

Description:
Save saves the current document at the location Path.

SaveAs

Method: SaveAs (strFileName as string) as boolean

Dispatch Id: *1004*

Description:
SaveAs sets Path to *strFileName* and then saves the document to this location.

Events

The XMLSpyControlDocument ActiveX control provides following connection point events:

OnDocumentOpened
OnDocumentClosed
OnModifiedFlagChanged
OnContextChanged
OnFileChangedAlert
OnActivate
OnSetEditorTitle

OnActivate

Event: OnActivate()

Dispatch Id: *1005*

Description:
This event is triggered when the document control is activated, has the focus, and is ready for user input.

OnContextChanged

Event: `OnContextChanged(i_strContextName as String, ibActive as bool) as bool`

Dispatch Id: *1004*

Description:
This event is triggered when this document is shown in a different XMLSpy view. The following values are passed:

- Grid view - `"View_0"` is passed as the context name
- Text view - `"View_1"` is passed as the context name
- Browser view - `"View_2"` is passed as the context name
- Schema view - `"View_3"` is passed as the context name
- Authentic view - `"View_4"` is passed as the context name
- WSDL view - `"View_5"` is passed as the context name

OnDocumentClosed

Event: `OnDocumentClosed(objDocument as Document)`

Dispatch Id: *1001*

Description:
This event is triggered whenever the document loaded into this control is closed. The argument `objDocument` is a `Document` object from the XMLSpy automation interface and should be used with care.

OnDocumentOpened

Event: `OnDocumentOpened(objDocument as Document)`

Dispatch Id: *1000*

Description:
This event is triggered whenever a document is opened in this control. The argument `objDocument` is a `Document` object from the XMLSpy automation interface, and can be used to query for more details about the document, or perform additional operations.

OnDocumentSaveAs

Event: `OnContextDocumentSaveAs(i_strFileName as String)`

Dispatch Id: *1007*

Description:
This event is triggered when this document gets internally saved under a new name.

OnFileChangedAlert

Event: `OnFileChangedAlert() as bool`

Dispatch Id: *1003*

Description:
This event is triggered when the file loaded into this document control is changed on the hard disk by another application. Clients should return true, if they handled the event, or false, if XMLSpy should handle it in its customary way, i.e. prompting the user for reload.

OnModifiedFlagChanged

Event: `OnModifiedFlagChanged` (`ibIsModified` as boolean)

Dispatch Id: *1002*

Description:
This event gets triggered whenever the document changes between modified and unmodified state. The parameter *i_bIsModifed* is *true* if the document contents differs from the original content, and *false*, otherwise.

OnSetEditorTitle

Event: OnSetEditorTitle ()

Dispatch Id: *1006*

Description:
This event is being raised when the contained document is being internally renamed.

4.7.5 XMLSpyControlPlaceHolder

Properties available for all kinds of placeholder windows:
`PlaceholderWindowID`

Properties for project placeholder window:
`Project`

Methods for project placeholder window:
`OpenProject`
`CloseProject`

The `XMLSpyControlPlaceHolder` control is used to show the additional XMLSpy windows like Overview, Library or Project window. It is used like any other ActiveX control and can be placed anywhere in the client application.

```
CLSID:135DEEF4-6DF0-47c2-8F8C-F145F5F3F672
ProgID:Altova.XMLSpyControlPlaceHolder
```

Properties

The following properties are defined:

`PlaceholderWindowID`

Access to XMLSpyAPI:
`Project`

Label

Property: Label as String (read-only)

Dispatch Id: *1001*

Description:
This property gives access to the title of the placeholder. The property is read-only.

PlaceholderWindowID

Property: PlaceholderWindowID as XMLSpyControlPlaceholderWindow

Dispatch Id: *1*

Description:
Using this property the object knows which XMLSpy window should be displayed in the client area of the control. The PlaceholderWindowID can be set at any time to any valid value of the XMLSpyControlPlaceholderWindow enumeration. The control changes its state immediately and shows the new XMLSpy window.

Project

Property: Project as Project (read-only)

Dispatch Id: *2*

Description:
The Project property gives access to the Project object of the XMLSpy automation server API. This interface provides additional functionality which can be used with the project loaded into the control. The property will return a valid project interface only if the placeholder window has PlaceholderWindowID with a value of XMLSpyXProjectWindow (=3). The property is read-only.

Methods

The following method is defined:

OpenProject
CloseProject

OpenProject

Method: OpenProject(strFileName as string) as boolean

Dispatch Id: *3*

Description:
OpenProject loads the file strFileName as the new project into the control. The method will fail if the placeholder window has a PlaceholderWindowID different to XMLSpyXProjectWindow (=3).

CloseProject

Method: CloseProject()

Dispatch Id: *4*

Description:
CloseProject closes the project loaded the control. The method will fail if the placeholder window has a PlaceholderWindowID different to XMLSpyXProjectWindow (=3).

Events

The XMLSpyControlPlaceholder ActiveX control provides following connection point events:

OnModifiedFlagChanged

OnModifiedFlagChanged

Event: OnModifiedFlagChanged(ibIsModified as boolean)

Dispatch Id: *1*

Description:
This event gets triggered only for placeholder controls with a PlaceholderWindowID of XMLSpyXProjectWindow (=3). The event is fired whenever the project content changes between modified and unmodified state. The parameter *i_bIsModifed* is *true* if the project contents differs from the original content, and *false*, otherwise.

OnSetLabel

Event: OnSetLabel(istrNewLabel as string)

Dispatch Id: *1000*

Description:
Raised when the title of the placeholder window is changed.

4.7.6 Enumerations

The following enumerations are defined:

ICActiveXIntegrationLevel
XMLSpyControlPlaceholderWindow

ICActiveXIntegrationLevel

Possible values for the IntegrationLevel property of the XMLSpyControl.

```
ICActiveXIntegrationOnApplicationLevel  = 0
ICActiveXIntegrationOnDocumentLevel     = 1
```

XMLSpyControlPlaceholderWindow

This enumeration contains the list of the supported additional XMLSpy windows.

```
XMLSpyControlNoToolWnd                          = -1
XMLSpyControlEntryHelperTopToolWnd              = 0
XMLSpyControlEntryHelperMiddleToolWnd           = 1
XMLSpyControlEntryHelperBottomToolWnd           = 2
XMLSpyControlValidatorOutputToolWnd             = 3
XMLSpyControlProjectWindowToolWnd               = 4
XMLSpyControlXSLTDebuggerContextToolWnd         = 5
XMLSpyControlXSLTDebuggerCallstackToolWnd       = 6
XMLSpyControlXSLTDebuggerVariableToolWnd        = 7
XMLSpyControlXSLTDebuggerWatchToolWnd           = 8
XMLSpyControlXSLTDebuggerTemplateToolWnd        = 9
XMLSpyControlXSLTDebuggerInfoToolWnd            = 10
XMLSpyControlXSLTDebuggerMessageToolWnd         = 11
XMLSpyControlXSLTDebuggerTraceToolWnd           = 12
XMLSpyControlSOAPDebuggerToolWnd                = 13
XMLSpyControlXPathProfilerListToolWnd           = 14
XMLSpyControlXPathProfilerTreeToolWnd           = 15
XMLSpyControlXPathDialogToolWnd                 = 16
XMLSpyControlDBQueryManagerToolWnd              = 17
XMLSpyControlInfoToolWnd                        = 18
XMLSpyControlXSLOutlineToolWnd                  = 19
XMLSpyControlSchemaFindToolWnd                  = 20
XMLSpyControlXBRLFindToolWnd                    = 21
XMLSpyControlChartsToolWnd                      = 22
```

Altova XMLSpy 2017 Enterprise Edition

付録

付録

以下の付録には、XMLSpy に関する技術的な情報や、ライセンスに関する重要な情報が収められています。各付録には以下のようにサブセクションが収められています：

エンジン情報

- XSLT とXQuery エンジン情報
- XSLT とXQuery 拡張関数

データ型、DB とXML スキーマ間の変換

- DB からXML スキーマへの変換
- XML スキーマからDB への変換

技術情報

- OS ならびにメモリの必要条件
- Altova XML パーサー
- Altova XSLT ならびにXQuery エンジン
- Unicode のサポート
- インターネットへの接続

ライセンス情報

- 電子的なソフトウェアの配布
- ソフトウェアのアクティベーションとライセンスメータリング
- 著作権
- 使用許諾契約書

1 XSLT および XQuery エンジンに関する情報

XMLSpy のXSLT およびXQuery エンジンは、W3C 仕様に従っています、ですから、XMLSpy の以前のバージョン内のAltova エンジンよりも厳密です。この結果、以前のエンジンで無視されていた小さなエラーが、XMLSpy によりエラーとして挙げられます。

例えば:

- パス演算子の結果がノードと非 -ノードを両方含む場合、型エラー (`err:XPTY0018`) です。
- パス式 `E1/E2` 内の `E1` がノードのシーケンスを評価しない場合、型エラー (`err:XPTY0019`) です。

この種類のエラーが発生した場合、XSLT/XQuery ドキュメントまたはインスタンスドキュメントを必要に応じて修正してください。

このセクションは、エンジンの実装固有の機能を仕様別に整理して説明します。

- XSLT 1.0
- XSLT 2.0
- XSLT 3.0
- XQuery 1.0
- XQuery 3.1

1.1 XSLT 1.0

XMLSpy のXSLT 1.0 エンジンは、World Wide Web Consortium (ワールド・ワイド・ウェブ・コンソーシアム) (W3C) の1999 年 11月 16日版の XSLT 1.0 勧告 および1999 年 11月 16日版の XPath 1.0 勧告 に準拠します。実装に関しての以下の情報に注意してください。

実装についての注意点

`xsl:output` の`method` 属性がHTMLに設定された場合、または、がHTML 出力 デフォルトで選択されている場合、内の特殊文字はHTML ドキュメントにHTML 文字参照として出力内に挿入されます。例えば、文字 U+00A0 (ブレーク無しのスペースのための 16進数レファレンス)がHTML コード内に文字の参照 (` ` or ` `) 、または、エンティティ参照 ` ` として挿入されます

1.2 XSLT 2.0

このセクション:

- エンジン適合性
- 下位互換性
- 名前空間
- スキーマ認識
- 実装固有の振る舞い

適合性

XMLSpy のXSLT 2.0 エンジンは、World Wide Web Consortium (ワールド・ワイド・ウェブ・コンソーシアム) (W3C) の2007 年 1月 23日版の XSLT 2.0 勧告 および2010 年 12月 14日版の XPath 2.0 勧告に準拠します。

下位互換性

XSLT 2.0 エンジンは下位互換性を有します。XSLT 2.0 エンジンの 下位互換性が有効になるのは、XSLT 1.0 スタイルシートを処理するためにXSLT 2.0 エンジン が使用される際です。XSLT 1.0 エンジン と下位互換性を持つXSLT 2.0 エンジンにより作成される出力に違いがあるかもしれないことに注意してください。

名前空間

XSLT 2.0 スタイルシートは、XSLT 2.0. プレフィックス内で使用することのできる型コンストラクタおよび関数を使用するため、以下の名前空間を宣言する必要があります。 下のリストは通常使用されるリストです。希望する場合は、代替プレフィックスを使用することもできます。

名前空間	プレフィックス	名前空間 URI
XML スキーマ型	`xs:`	`http://www.w3.org/2001/XMLSchema`
XPath 2.0 関数	`fn:`	`http://www.w3.org/2005/xpath-functions`

通常これらの名前空間は、以下のリストで表示されるように`xsl:`スタイルシートまたは、`xsl:transform` 要素で宣言されます:

```
<xsl:スタイルシート version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    ...
</xsl:スタイルシート>
```

次の点に注意してください:

- XSLT 2.0 エンジンは、(上のテーブルでリストされている)XPath 2.0 およびXQuery 1.0 関数 名前空間 を**デフォルトの関数名前空間**として使用します。XPath 2.0 およびXSLT 2.0 関数をプレフィックス無しでスタイ

ルシート内で使用することができます。XPath 2.0 関数 名前空間 をスタイルシート内でプレフィックスと共に宣言すると、割り当てられた宣言内でプレフィックスを追加して使用することができます。t

- XML スキーマ名前空間から型コンストラクタ型を使用する場合、名前空間 宣言内で使用された、プレフィックスを使用して型コンストラクタを呼び出さなければなりません (例えば、`xs:date`)。
- XPath 2.0 関数の一部は、XML スキーマデータ型と同じ名前を保有します。例えば、XPath 関数 `fn:string` および `fn:boolean` のためには、同じロケーション名 :`xs:string` および`xs:boolean` を持つ XML スキーマデータ型 が存在します。ですから、XPath 式 `string('Hello')`を使用する場合、式は、`xs:string('Hello')`ではなくて`fn:string('Hello')` として検証します。

スキーマ認識

XSLT 2.0 エンジンは、スキーマを認識します。ですから、ユーザー定義 スキーマ型 および`xsl:validate` 命令を使用することができます。

実装固有の振る舞い

以下は、XSLT 2.0 エンジンが、特定のXSLT 2.0 関数の振る舞いの実装 特定のアスペクトをどのように扱うかの説明です。

`xsl:result-document`
追加してサポートされるエンコードは以下の通りです (Altova-固有):`x-base16tobinary` および`x-base64tobinary`.

`function-available`
インスコープ関数の使用をテストする関数 (XSLT、XPath、および拡張関数)。

`unparsed-text`
`href` 属性は、以下を受け入れます (i) ベースuri フォルダー内のファイルの相対パス、および (ii) 相対パスを持つまたは持たない`file://` プロトコール。追加してサポートされるエンコードは以下の通りです (Altova-固有):`x-binarytobase16` および`x-binarytobase64`.

`unparsed-text-available`
`href` 属性は、以下を受け入れます (i) ベースuri フォルダー内のファイルの絶対パス、および (ii) 絶対パスを持つまたは持たない`file://` プロトコール。追加してサポートされるエンコードは以下の通りです (Altova-固有):`x-binarytobase16` および`x-binarytobase64`.

メモ: RaptorXML の先行製品であるAltovaXML で実装されていた以下のエンコード値は使用しないでください: `base16tobinary`, `base64tobinary`, `binarytobase16` and `binarytobase64`.

1.3 XSLT 3.0

XMLSpy のXSLT 3.0 エンジンは、World Wide Web Consortium (ワールド・ワイド・ウェブ・コンソーシアム) (W3C) の2017 年 2月 7日版 XSLT 3.0 勧告候補 および2017 年 1月 17日版 XPath 3.1 勧告候補 に準拠します。

XSLT 3.0 エンジンは、XSLT 2.0 エンジンと同様の実装固有の機能を搭載しています。更に、以下のXSLT 3.0 機能へのサポートを含みます。更に次の一連の新規のXSLT 3.0 機能をサポートします :XPath/XQuery 3.1 関数とオペレーターとXPath 3.1 仕様。

メモ: 任意のストリーミングの機能は現在サポートされていません。`streamable` 属性の値に関係なく、ドキュメント全体がメモリにロードされ、使用することのできるメモリが十分な場合は処理されます。メモリが問題の場合は、システムに解決策を追加する必要があります。

1.4 XQuery 1.0

このセクション:

- エンジン適合性
- スキーマ認識
- エンコード
- 名前空間
- XML ソースと検証
- 静的および動的な型のチェック
- ライブラリモジュール
- 外部モジュール
- 照合順序
- 数値データの精度
- XQuery 命令サポート

適合性

XMLSpy のXQuery 1.0 エンジンは、World Wide Web Consortium (ワールド・ワイド・ウェブ・コンソーシアム) (W3C) の2010 年 12月 14日版の XQuery 1.0 勧告に準拠します。XQuery 標準は、多数の機能の実装についての裁量を提供します。下には、XQuery 1.0 エンジンがどのようにこれらの機能を実装するかについて説明するリストが下に挙げられています。

スキーマ認識

XQuery 1.0 エンジンはスキーマを認識します。

エンコード

UTF-8 およびUTF-16 文字のエンコードは、サポートされています。

名前空間

以下の名前空間 URI と関連するバインドはは定義済みです。

名前空間	プレフィックス	名前空間 URI
XML スキーマ型	`xs:`	`http://www.w3.org/2001/XMLSchema`
スキーマインスタンス	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
内蔵の 関数	`fn:`	`http://www.w3.org/2005/xpath-functions`
Local 関数	`local:`	`http://www.w3.org/2005/xquery-local-functions`

次の点に注意してください:

- XQuery 1.0 エンジンは、上にリストされたプレフィックスを名前空間に対応するバウンドとして認識します。
- Since the 上にリストされた内蔵の関数 名前空間は、XQuery内のデフォルトの関数です。内蔵の関数 が呼び出される際、名前空間、`fn:`プレフィックス を使用する必要はありません。 (例えば、`string("Hello")` が`fn:string` 関数を呼び出す場合。) しかし、プレフィックス`fn:` は、クエリプロローグ内で名前空間を宣言することなく内蔵の関数を呼び出す時に使用することができます。 (サンプル:`fn:string("Hello")`).
- クエリプロローグ内で`default function` 名前空間 式を宣言することにより、デフォルトの関数 名前空間をすることにより変更することができます。
- XML スキーマ名前空間空の型を使用する場合、プレフィックス`xs:` は、名前空間を明確に宣言することなく、また、これらのプレフィックスをクエリプロローグ内でバインドすることなく使用することができます。 (サンプル:`xs:date` および`xs:yearMonthDuration`。) XML スキーマ名前空間のために、他のプレフィックスを使用する場合は、クエリプロローグ内で明確に宣言されている必要があります。 (サンプル:`declare 名前空間 alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)
- `untypedAtomic`, `dayTimeDuration`, および`yearMonthDuration` データ型 が 23 January 2007 のCR 共に、XPath データ型 名前空間からXML スキーマ名前空間へ移動されていることに注意してください。ですから以下となります:`xs:yearMonthDuration`。

関数のための名前空間、型コンストラクタ ノード テスト、が間違って割り当てられている場合、エラーが発生します。しかし、一部の関数はスキーマデータ型と同じ名前を持つことに注意してください。 例:`fn:string` および `fn:boolean`。(`xs:string` および`xs:boolean` は宣言されています)名前空間 プレフィックス、関数または型コンストラクタが使用されるか決定します。

XML ソースドキュメントと検証

XML ドキュメントused in executing an XQuery ドキュメントwith XQuery 1.0 エンジンmust be 整形式 . しかし、they do not need to be valid according to an XML スキーマ.If the ファイルis not valid, the invalid ファイルis loaded without schema information. XML ファイルが外部スキーマと関連付けられ、また有効な場合、then post-schema 検証情報 is generated for XML データand will be used for クエリ検証 .

静的および動的な型のチェック

静的分析フェーズchecks aspects of クエリsuch as syntax, whether 外部レファレンス (例 for モジュール) exist, whether invoked 関数と変数 are defined, and so on. If an error is detected in 静的分析フェーズ, it is reported and the execution is stopped.

動的な型チェックis carried out ランタイム中 ,when クエリis actually executed. If a type is incompatible with the requirement of an operation, an error is reported. 例えば、式`xs:string("1") + 1` は、エラーを返します。because the addition operation cannot be carried out on an operand of type `xs:string`.

ライブラリモジュール

ライブラリモジュールstore 関数と変数 so they can be reused. XQuery 1.0 エンジンsupports モジュールthat are stored in **a single external XQuery ファイル**. Such モジュール ファイルmust contain モジュール 宣言 in its prolog, which associates a target 名前空間 .以下はモジュールサンプルです:

```
module 名前空間 libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns:webaddress() { "http://www.altova.com" };
```

すべての関数および変数は、モジュールに関連した名前空間に属するモジュール内で宣言されています。.モジュールis used by importing it into an XQuery ファイルwith the `import module` statement クエリプロローグ内の. `import module` ステートメントは、ライブラリモジュールファイル内で直接宣言された、関数と変数のみをインポートします。例:

```
import module namespace modlib = "urn:module-library" at
   "modulefilename.xq";
if  ($modlib:company = "Altova")
then        modlib:webaddress()
else        error("No match found.")
```

外部関数

外部関数 は、サポートされていません。i.e. in those 式 using the `external` keyword, as in:

```
declare function hoo($param as xs:integer) as xs:string external;
```

照合順序

デフォルトの照合順序 is the Unicode-コードポイント照合順序 ,which compares strings on the basis of their Unicode コードポイント.その他にサポートされる照合順序 are the ICU 照合順序 listed here. To use a specific 照合順序 ,supply its URI as given in the list of supported 照合順序. Any string comparisons, including for the `fn:max` and `fn:min` 関数 ,will be made according to the specified 照合順序 .照合順序オプションが指定されていない場合、デフォルトのUnicode-コードポイント照合順序 が使用されます。

数値データの精度

- The `xs:integer` データ型 is arbitrary-精度 ,i.e. it can represent any number of 桁 .
- The `xs:decimal` データ型 has a limit of 20 桁 after the decimal point.
- The `xs:float` and `xs:double` データ型 have limited-精度 of 15 桁 .

XQuery 命令サポート

`Pragma` 命令 は、サポートされていません。発生した場合、無視されフォールバックの式が検証されます。

1.5 XQuery 3.1

of XMLSpy のXQuery 3.1 エンジンは、World Wide Web Consortium (ワールド・ワイド・ウェブ・コンソーシアム) (W3C) の2017 年 1月 17日版の XQuery 3.1 候補勧告に準拠し、またXPath および XQuery 関数 3.1. へのサポートを含みます。XQuery 3.1 仕様は、 3.0仕様のスーパーセットです。XQuery 3.1 エンジンは、ですかXQuery 3.0 機能をサポートします。

実装固有の特性はXQuery 1.0 でも同様です。

2 XSLT と XPath/ XQuery 関数

このセクションでは、XPath および/または XQuery 式で使用することのできる、Altova 拡張関数と他の拡張関数をリストします。Altova 拡張関数は Altova の XSLT および XQuery エンジンで使用することができ、W3C 標準で定義された関数ライブラリで使用することのできる機能に追加して機能を提供します。

一般的な情報

以下の一般的な情報に注意してください:

- W3C 仕様により定義されているコア関数ライブラリの関数は、関数の呼び出しにプレフィックスは必要ありません。これは、XSLT および XQuery エンジンが、XPath/XQuery 関数仕様で指定されている `http://www.w3.org/2005/xpath-functions` プレフィックス無しの関数をデフォルト関数の名前空間に属するものとして読み込むためです。
- 関数において、各アイテムが引数となるようなシーケンスが期待されており、2つ以上のアイテムがシーケンスにより呼び出された場合、エラーが返されます。
- 全ての比較は Unicode コードポイントコレクションを使用することで行われます。
- QName の結果は `[prefix:]localname` という形式でシリアル化されます。

xs:decimal の精度

精度とは、数値内にある桁数のことで、仕様では少なくとも18桁が求められます。`xs:decimal` 型に結果が収められる除算の場合、端数処理を行うことなく精度は小数点以下の19桁になります。

黙示的なタイムゾーン

2つの `date`、`time`、または `dateTime` 値を比較する場合、比較する値のタイムゾーンを明らかにする必要があります。値の中にタイムゾーンが明示的に与えられていない場合、黙示的なタイムゾーンが使用されます。黙示的なタイムゾーンはシステムクロックから取得され `implicit-timezone()` 関数によりその値をチェックすることができます。

照合順序

デフォルトの照合順序は、Unicode コードポイントをベースに文字列を比較する Unicode コードポイント照合順序です。エンジンは Unicode 照合アルゴリズムを使用しています。他のサポートされる照合順序は下にリストされる ICU 照合順序 です。使用するには、サポートされる照合順序のリストの URI を提供してください (下のテーブル)。`max` と `min` 関数を含む、文字列の比較は、指定された照合順序に沿って行われます。照合順序オプションが指定されていない場合、デフォルトの Unicode コードポイント照合順序が使用されます。

言語	URI
`da`: デンマーク語	`da_DK`
`de`: ドイツ語	`de_AT, de_BE, de_CH, de_DE, de_LI, de_LU`
`en`: 英語	`en_AS, en_AU, en_BB, en_BE, en_BM, en_BW, en_BZ, en_CA, en_GB, en_GU, en_HK, en_IE, en_IN, en_JM, en_MH, en_MP, en_MT, en_MU, en_NA, en_NZ, en_PH, en_PK, en_SG, en_TT, en_UM, en_US, en_VI, en_ZA, en_ZW`
`es`: スペイン語	`es_419, es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_ES, es_GQ, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_US, es_UY, es_VE`

fr: フランス語	fr_BE, fr_BF, fr_BI, fr_BJ, fr_BL, fr_CA, fr_CD, fr_CF, fr_CG, fr_CH, fr_CI, fr_CM, fr_DJ, fr_FR, fr_GA, fr_GN, fr_GP, fr_GQ, fr_KM, fr_LU, fr_MC, fr_MF, fr_MG, fr_ML, fr_MQ, fr_NE, fr_RE, fr_RW, fr_SN, fr_TD, fr_TG
it: イタリア語	it_CH, it_IT
ja: 日本語	ja_JP
nb: ノルウェー語 (ブークモール)	nb_NO
nl: オランダ語	nl_AW, nl_BE, nl_NL
nn: ノルウェー語 (ニーノシュク)	nn_NO
pt: ポルトガル語	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: ロシア語	ru_MD, ru_RU, ru_UA
sv: スウェーデン語	sv_FI, sv_SE

名前空間軸
名前空間軸はXPath 2.0 にて廃止されましたが、名前空間軸の使用はサポートされています。XPath 2.0 メカニズムにより名前空間情報へアクセスするには、in-scope-prefixes()、namespace-uri()、namespace-uri-for-prefix(関数を使用してください。

2.1 Altova 拡張関数

Altova 拡張関数は XPath/XQuery 式で使用することができ、XPath、XQuery、および XSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は **Altova 拡張関数名前空間、** `http://www.altova.com/xslt-extensions` に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

W3C の XPath/XQuery 関数仕様で定義された関数は、以下で使用することができます : (i) XSLT 子アンテキスト内の XPath 式と (ii) XQuery 文書内の XQuery 式。このドキュメントでは、前者 (XSLT 内の XPath) のコンテキストで使用することのできる関数を、`XP` シンボルと共に表示し、と称します。後者 (XQuery) で使用することのできる関数は の `XQ` シンボルと共に表示され、XQuery 関数と共に作業することができます。W3C の XSLT 仕様は、XPath/XQuery 関数の仕様ではなく、XSLT 文書内の XPath 式でも使用することのできる関数を定義します。これらの関数は、`XSLT` シンボルと共に表示され、XSLT 関数と称されます。関数を使用することのできる XPath/XQuery および XSLT のバージョンは、関数の詳細に記載されています (*下のシンボルを参照してください*)。XPath/XQuery および XSLT 関数ライブラリからの関数は、*プレフィックス*無しでリストされています。Altova 拡張関数などの、他のライブラリからの関数は*プレフィックス*と共にリストされています。

XPath 関数 (XSLT 内の XPath 式で使用):	`XP1` `XP2` `XP3`
XSLT 関数 (XSLT 内の XPath 式で使用):	`XSLT1` `XSLT2` `XSLT3`
XQuery 関数 (XQuery 内の XQuery 式で使用):	`XQ1` `XQ3`

XSLT 関数

XSLT 関数は XSLT 2.0 の `current-group()` や `key()` 関数と同様に、XSLT コンテキストにて使用することができます。 (例えば、XQuery コンテキストなどの) 非 XSLT コンテキストでは使用することができません。XBRL に対する XSLT 関数は、XBRL をサポートするエディションの Altova 製品でのみ使用することができます。

XPath/XQuery 関数

XPath/XQuery 関数は、 XSLT コンテキスト、 XQuery 関数の XPath 式で使用することができます :

- 日付 時刻
- 位置情報
- イメージに関連した
- 数値
- シーケンス
- 文字列
- その他

チャート関数 (Enterprise および Server Editions のみ)

チャート関数のための Altova 拡張子は、**Enterprise** および **サーバー エディション** の Altova 製品でしかサポートされていません。XML データからチャートを生成することができます。

2.1.1 XSLT 関数

XSLT 拡張関数 はXSLT コンテキスト内のXPath 式にて使用することができます。 (例えば、 XQuery コンテキストなどの)非 XSLT コンテキストでは使用することができません。

関数の名前指定と言語の適用性

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、** `http://www.altova.com/xslt-extensions` に収められており、 `altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3

標準関数

▼ **distinct-nodes [altova:]**

altova:distinct-nodes(*node()**)** を**node()*** とする XSLT1 XSLT2 XSLT3
入力として1つ以上のノードを必要とし、同じセットから重複した値を持つノードを除いたノードを返します。XPath/ XQuery 関数 `fn:deep-equal` を使用して比較を行うことができます。

▫ *サンプル*

- **altova:altova:distinct-nodes**(country)は重複した値を持つものを除く、全ての子 country ノード返します。

▼ **evaluate [altova:]**

altova:evaluate(XPathExpression *as xs:string***[, ValueOf$p1, ... ValueOf$pN])** XSLT1 XSLT2 XSLT3
XPath 式を必要とし、必須引数として文字列をパスします。評価された式の出力を返します。例えば:
altova:evaluate('//Name[1]') は、ドキュメント内の最初のName 要素のコンテンツを返します。式 `//Name[1]` は、一重引用符を使用することにより、文字列としてパスされます。

`altova:evaluate` 関数は、オプションとして追加の引数を持つことができます。これらの引数は、`p1`, `p2`, `p3`... `pN` の名前を持つスコープ内の変数の値です。使用に関して以下の点に注意してください: (i) 変数は、`x` が整数である箇所のフォーム`pX` の名前と共に定義される必要があります。(ii) `altova:evaluate` 関数の引数は、(*上の署名参照*) 2 番目の引数からは、数値順の変数のシーケンスに対応した引数のシーケンス変数の値を与えます: `p1` to `pN`: 第 2引数は変数 `p1` の値で、第 3引数は、変数 `p2` の値です。 (ii) 変数の値は型 `item*` である必要があります。

▫ *サンプル*

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 'hi')" />
```

outputs "hi 20 10"

上のリストに関して、以下の点に注意してください:

- `altova:evaluate` 式の第 2引数は、変数 `$p1` に割り当てられた値で、第三の引数は変数

$p2 に割り当てられた値です。
- 関数の第 4番目の引数は、引用符による囲いで表示された文字列の値です。
- xs:variable 要素の select 属性は、XPath 式を提供します。この式は xs:string の型である必要があり、一重引用符で囲まれています。

変数の使用方法を更に説明するサンプル

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, //Name[1])" />
  ```
  最初の Name 要素の出力値

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, '//Name[1]')" />
  ```
  Outputs "//Name[1]"

altova:evaluate() 拡張関数は、XSLT スタイルシート内の XPath 式が動的に評価される必要のある1つ以上の部分を持つシチュエーションで役に立ちます。例えば、ユーザーが並べ替えの必要条件をリクエストする場合、このシチュエーションは属性 UserReq/@sortkey に保管されます。スタイルシートでは、以下の式が使用できます：<xsl:sort select="**altova:evaluate(../UserReq/@sortkey)**" order="ascending"/>。altova:evaluate() 関数は、コンテキストノードの親の UserReq 子要素の sortkey 属性を読み込みます。sortkey 属性の値が Price の場合、Price は、altova:evaluate() 関数により返され、select 属性 <xsl:sort select="**Price**" order="ascending"/> の値になります。この sort 命令が、Order という要素のコンテキスト内で発生する場合、Order 要素は Price の子の値に従い並べ替えられます。また、@sortkey の値が、Date の場合、Order 要素は、Date の子の値に従い並べ替えられます。ですから、Order の並べ替えの条件は、ランタイムでの sortkey 属性から選択されます。これは、以下の式などでは達成することはできません：<xsl:sort select="**../UserReq/@sortkey**" order="ascending"/>。上の場合、並べ替え条件は sortkey 属性自身であり、Price または Date（または、現在の sortkey のコンテンツ）ではありません。

メモ: 静的なコンテキストは、呼び出し環境の名前空間、型、機能、しかし変数を以外を含みます。ベース URI とデフォルトの名前空間は継承されます。

追加サンプル

- 静的な変数：<xsl:value-of select="$i3, $i2, $i1" />
  3 つの変数の値を出力します。

- 動的な変数を持つ動的 XPath 式：
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />
  ```
  "30 20 10" を出力します。

- 動的な変数を持たない XPath 式：
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath)" />
  ```
  出力エラー：$p3 に対して定義されている変数はありません。

encode-for-rtf [altova:]

altova:encode-for-rtf(input *as xs:string*, **preserveallwhitespace** *as xs:boolean*, **preservenewlines** *as xs:boolean*) を **xs:string** とする **XSLT2** **XSLT3**

RTF のためのコードに入力文字列を変換します。空白と新しい行は、それぞれの引数により指定される boolean の値に基づき保管されます。

[トップ]

XBRL 関数

Altova XBRL 関数は XBRL をサポートする Altova 製品のエディションのみで使用することができます。

▼ **xbrl-footnotes [altova:]**

altova:xbrl-footnotes(*node()***)** を **node()*** とする XSLT2 XSLT3
ノードを入力引数として必要とし、入力ノードに参照される XBRL フットノート ノードを返します。

▼ **xbrl-labels [altova:]**

altova:xbrl-labels(*xs:QName, xs:string***)** を **node()*** とする XSLT2 XSLT3
以下の 2 つの入力引数を必要とします : ノード名とノードを含むタクソノミファイルロケーション。関数は、入力ノードと関連した XBRL ラベルノードを返します。

[トップ]

2.1.2 XPath/ XQuery 関数 :日付と時刻

Altova の日付 時刻拡張関数は XPath と XQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の **XPath 3.0** および **XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性

Altova 拡張関数は XPath/XQuery 式で使用することができ、XPath、XQuery、および XSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は **Altova 拡張関数名前空間、http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意ください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内の XPath 式で使用):	XP1 XP2 XP3
XSLT 関数 (XSLT 内の XPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内の XQuery 式で使用):	XQ1 XQ3

▼ 機能によりグループ化

- xs:dateTime に期間を追加して、xs:dateTime を返す
- xs:date に期間を追加して、xs:date を返す
- xs:time に期間を追加して、return xs:time を返す
- フォーマットと期間の取得
- 現在の日付 時刻を生成する関数からタイムゾーンを削除する
- 日付から整数を週の曜日として返す
- 日付から週数を整数として返す
- 各型の構文コンポーネントから日付、時刻、期間 の型を構築する

- 文字列入力から日付、日付時刻 または時刻 を構築する
- 年齢に関連した関数

▼ アルファベット順にグループ化

altova:add-days-to-date
altova:add-days-to-dateTime
altova:add-hours-to-dateTime
altova:add-hours-to-time
altova:add-minutes-to-dateTime
altova:add-minutes-to-time
altova:add-months-to-date
altova:add-months-to-dateTime
altova:add-seconds-to-dateTime
altova:add-seconds-to-time
altova:add-years-to-date
altova:add-years-to-dateTime
altova:age
altova:age-details
altova:build-date
altova:build-duration
altova:build-time
altova:current-dateTime-no-TZ
altova:current-date-no-TZ
altova:current-time-no-TZ
altova:format-duration
altova:parse-date
altova:parse-dateTime
altova:parse-duration
altova:parse-time
altova:weekday-from-date
altova:weekday-from-dateTime
altova:weeknumber-from-date
altova:weeknumber-from-dateTime

[トップ]

---

xs:dateTime に期間を追加する XP3 XQ3

これらの関数は xs:dateTime に期間を追加し、xs:dateTime を返します。xs:dateTime 型は CCYY-MM-DDThh:mm:ss.sss のフォーマットです。これは xs:date と xs:time フォーマットの連結で、T により区切られています。タイムゾーンサフィックス +01:00 (for example) は任意です。

▼ add-years-to-dateTime [altova:]

altova:add-years-to-dateTime(DateTime *as xs:dateTime*, **Years** を *xs:integer*) as **xs:dateTime** とする XP3 XQ3

日付までの期間を年数で表示します。第 2 の引数は第 1 の引数として与えられた xs:date に追加される年数です。結果は xs:date 型です。

▼ *サンプル*

- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) returns 2024-01-15T14:00:00
- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -4) returns 2010-01-15T14:00:00

▼ add-months-to-dateTime [altova:]

altova:add-months-to-dateTime(DateTime *as xs:dateTime*, **Months** を *xs:integer*) as **xs:dateTime** とする **XP3 XQ3**

xs:dateTime に月数での期間を追加します (*下のサンプル参照*)。第 2の引数は、第 1の引数として与えられたxs:dateTime に追加される月数です。結果はxs:dateTime 型です。

サンプル

- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) 2014-11-15T14:00:00 を返します。
- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -2) 2013-11-15T14:00:00 を返します。

add-days-to-dateTime [altova:]

altova:add-days-to-dateTime(DateTime *as xs:dateTime*, **Days** *as xs:integer*) を **xs:dateTime** とする **XP3 XQ3**

xs:dateTime に日数での期間を追加します (*下のサンプル参照*)。第 2の引数は、第 1の引数として与えられたxs:dateTime に追加される日数です。結果はxs:dateTime 型です。

サンプル

- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10)は 2014-01-25T14:00:00 を返します。
- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8)は 2014-01-25T14:00:00 を返します。

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(DateTime *as xs:dateTime*, **Hours** *as xs:integer*) を**xs:dateTime** とする **XP3 XQ3**

xs:dateTime に時間数での期間を追加します (*下のサンプル参照*)。第 2の引数は、第 1の引数として与えられたxs:dateTime に追加される時間数です。結果はxs:dateTime 型です。

サンプル

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) は 2014-01-15T23:00:00 を返します。
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) は 2014-01-15T05:00:00 を返します。

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(DateTime *as xs:dateTime*, **Minutes** *as xs:integer*) を**xs:dateTime** とする **XP3 XQ3**

xs:dateTime に分数での期間を追加します (*下のサンプル参照*)。第 2の引数は、第 1の引数として与えられたxs:dateTime に追加される分数です。結果はxs:dateTime 型です。

サンプル

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) 2014-01-15T14:55:00 を返します。
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) 2014-01-15T14:05:00 を返します。

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(DateTime *as xs:dateTime*, **Seconds** *xs:integer*) as **xs:dateTime** とする **XP3** **XQ3**

xs:dateTime に秒数での期間を追加します (*下のサンプル参照*)。第 2 の引数は、第 1 の引数として与えられた xs:dateTime に追加される秒数です。結果は xs:dateTime 型です。

サンプル

- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20) 2014-01-15T14:00:30 を返します。
- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5) 2014-01-15T14:00:05 を返します。

[トップ]

---

xs:date に期間を追加する **XP3** **XQ3**

これらの関数は **xs:date** に期間を追加し、**xs:date** を返します。xs:date 型は CCYY-MM-DD フォーマットです。

add-years-to-date [altova:]

altova:add-years-to-date(Date *as xs:date*, **Years** *as xs:integer*) を **xs:date** とする **XP3** **XQ3**

日付までの期間を年数で表示します。第 2 の引数は第 1 の引数として与えられた xs:date に追加される年数です。結果は xs:date 型です。

サンプル

- **altova:add-years-to-date**(xs:date("2014-01-15"), 10) は 2024-01-15 を返します。
- **altova:add-years-to-date**(xs:date("2014-01-15"), -4) は 2010-01-15 を返します。

add-months-to-date [altova:]

altova:add-months-to-date(Date *as xs:date*, **Months** *as xs:integer*) を **xs:date** とする **XP3** **XQ3**

日付までの期間を月数で表示します。第 2 の引数は第 1 の引数として与えられた xs:date に追加される月数です。結果は xs:date 型です。

サンプル

- **altova:add-months-to-date**(xs:date("2014-01-15"), 10) 2014-11-15 を返します。

- **altova:add-months-to-date**(xs:date("2014-01-15"), -2) 2013-11-15 を返します。

add-days-to-date [altova:]

altova:add-days-to-date(Date *as xs:date*, **Days** *as xs:integer*) を**xs:date** とする **XP3 XQ3**
日付までの期間を日数で表示します。第 2の引数は第 1の引数として与えられたxs:date に追加される日数です。結果はxs:date 型です。

サンプル

- **altova:add-days-to-date**(xs:date("2014-01-15"), 10)は 2014-01-25 を返します。
- **altova:add-days-to-date**(xs:date("2014-01-15"), -8) は2014-01-07 を返します。

[トップ]

フォーマットと期間の取得 XP3 XQ3

これらの関数は**xs:date** に期間を追加し、**xs:date** を返します。xs:date 型はCCYY-MM-DD フォーマットです。

format-duration [altova:]

altova:format-duration(Duration *as xs:duration*, **Picture** *as xs:string*) as **xs:string** とする **XP3 XQ3**
第 1の引数として提出された期間を、第 2の引数として提出された文字列によりをフォーマットします。出力は、文字列によりフォーマットされたテキスト文字列です。

サンプル

- **altova:format-duration**(xs:duration("P2DT2H53M11.7S"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") は"Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7" を返します。
- **altova:format-duration**(xs:duration("P3M2DT2H53M11.7S"), "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") は"Months:03 Days:02 Hours:02 Minutes:53" を返します。

parse-duration [altova:]

altova:parse-duration(InputString *as xs:string*, **Picture** *as xs:string*) を **xs:duration** とする **XP3 XQ3**
パターン化された文字列を最初の引数として、文字を第 2の引数とします。入力文字列は文字をベースに解析され、xs:duration が返されます。

サンプル

- **altova:parse-duration**("Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") は"P2DT2H53M11.7S" を返します。
- **altova:parse-duration**("Months:03 Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") は"P3M2DT2H53M" を返します。

[Top]

xs:time に期間を追加する XP3 XQ3

これらの関数は xs:time に期間を追加し、 xs:time を返します。 xs:time 型は hh:mm:ss.sss 構文フォームです。

文字 Z は協定世界時 (UTC) を表します。他のタイムゾーンは、UTC との差異を +hh:mm または -hh:mm のフォーマットで表示しています。タイムゾーンの値が無い場合は、UTC ではない未知のタイムゾーンとして見なされます。

add-hours-to-time [altova:]

altova:add-hours-to-time(Time *as xs:time*, **Hours** *as xs:integer*) を **xs:time** とする XP3 XQ3

日付までの期間を時間数で表示します。第 2 の引数は第 1 の引数として与えられた xs:time に追加される時間数です。結果は xs:time 型です。

サンプル

- **altova:add-hours-to-time**(xs:time("11:00:00"), 10) は 21:00:00 を返します。
- **altova:add-hours-to-time**(xs:time("11:00:00"), -7) は 04:00:00 を返します。

add-minutes-to-time [altova:]

altova:add-minutes-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) を **xs:time** とする XP3 XQ3

日付までの期間を分数で表示します。第 2 の引数は第 1 の引数として与えられた xs:date に追加される分数です。結果は xs:date 型です。

サンプル

- **altova:add-minutes-to-time**(xs:time("14:10:00"), 45) 14:55:00 を返します。
- **altova:add-minutes-to-time**(xs:time("14:10:00"), -5) 14:05:00 を返します。

add-seconds-to-time [altova:]

altova:add-seconds-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) を **xs:time** とする XP3 XQ3

時間までの期間を秒数で表示します。第 2 の引数は、第 1 の引数として与えられた xs:time に追加される秒数です。結果は xs:time 型です。第 2 のコンポーネントは 0 から 59.999 の範囲であることができます。

サンプル

- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20) は 14:00:20 を返します。
- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20.895) は 14:00:20.895 を返します。

[Top]

現在の日付 /時刻を生成する関数からタイムゾーンを削除する XP3 XQ3

これらの関数は、現在の **xs:dateTime**、**xs:date**、または **xs:time** 値からそれぞれタイムゾーンを削除します。xs:dateTime と xs:dateTimeStamp の差異は、後者のタイムゾーンが必要な場合です。前者の場合は任意で

す。) xs:dateTimeStamp 値のフォーマットは CCYY-MM-DDThh:mm:ss.sss±hh:mm または CCYY-MM-DDThh:mm:ss.sssZ です。、日付と時刻が xs:dateTimeStamp としてシステムクロックから読み込まれる場合、current-dateTime-no-TZ() 関数がタイムゾーンを削除するために使用されます。

current-dateTime-no-TZ [altova:]

altova:current-dateTime-no-TZ() を **xs:dateTime** とする XP3 XQ3

この関数は引数を必要としません。current-dateTime() (システムクロックによる現在の時刻) のタイムゾーンの部分を削除し、xs:dateTime の値を返します。

サンプル

現在の dateTime が **2014-01-15T14:00:00+01:00** の場合：

- **altova:current-dateTime-no-TZ()** は 2014-01-15T14:00:00 を返します。

current-date-no-TZ [altova:]

altova:current-date-no-TZ() を **xs:date** とする XP3 XQ3

この関数は引数を必要としません。current-date() (システムクロックによる現在の時刻) のタイムゾーンの部分を削除し、xs:date の値を返します。

サンプル

現在の date が **2014-01-15+01:00** の場合：

- **altova:current-date-no-TZ()** は 2014-01-15 を返します。

current-time-no-TZ [altova:]

altova:current-time-no-TZ() を **xs:time** とする XP3 XQ3

この関数は引数を必要としません。current-time() (システムクロックによる現在の時刻) のタイムゾーンの部分を削除し、xs:time の値を返します。

サンプル

現在の time が **14:00:00+01:00** の場合：

- **altova:current-time-no-TZ()** は 14:00:00 を返します。

[Top]

---

xs:dateTime または xs:date として曜日を返す XP3 XQ3

これらの関数は、xs:dateTime または xs:date から曜日を (整数として) を返します。曜日は (米国式フォーマットを使用して) 1 から 7 と番号づけられています。 日曜 =1 と番号付けられます。欧州のフォーマットでは月曜 (=1) として番号付けられます。日曜=1 である米国フォーマットは整数 0 がフォーマットを表示するために使用できる箇所で設定することができます。

weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *as xs:dateTime*) を **xs:integer** とする XP3 XQ3

時刻付きの日付を単一の引数として、この日付の曜日を正数として返します。曜日は 日曜 =1 から開始して番号を付けます。 (月曜=1 とする) ヨーロッパのフォーマットが必要な場合、この関数の他の署名を使用します (*下の次の署名を参照*)。

サンプル

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00")) は月曜

を表示する2 を返します。

altova:weekday-from-dateTime(DateTime *as xs:dateTime*, **Format** *as xs:integer*) as **xs:integer** とする **XP3 XQ3**
時間月の日付を最初の引数として、この日付の曜日を正数として返します。曜日は月曜=1 から開始して番号を付けます。第 2の引数 (整数) が0 の場合、日曜=1 から開始して、曜日は1から7 と番号を付けます。第 2の引数が0 以外の整数の場合、月曜=1です。第 2の引数がない場合、関数はこの関数の他の署名を持つと読み込まれます (*前の次の署名を参照*)。

サンプル

- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 1)` は月曜を表示する1 を返します。
- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 4)` は月曜を表示する1 を返します。
- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 0)` は月曜を表示する2 を返します。

weekday-from-date [altova:]

altova:weekday-from-date(Date *as xs:date*) を **xs:integer** とする **XP3 XQ3**
日付を単一の引数として、この日付の曜日を正数として返します。曜日は日曜=1 から開始して番号を付けます。(月曜=1 とする) ヨーロッパのフォーマットが必要な場合、この関数の他の署名を使用します (*下の次の署名を参照*)。

サンプル

- `altova:weekday-from-date(xs:date("2014-02-03+01:00"))` は月曜を表示する2 を返します。

altova:weekday-from-date(Date *as xs:date*, **Format** *as xs:integer*) を **xs:integer** とする **XP3 XQ3**
日付を最初の引数とし、この日付の曜日を正数として返します。曜日は月曜=1 から開始して番号を付けます。第 2 (フォーマット) 引数が0,の場合、日曜=1 から開始し、曜日は1 から7で番号付けられます。第 2 引数が整数で 0 以外の場合、月曜=1です。第 2 の引数がない場合、関数はこの関数の他の署名を持つと読み込まれます (*前の署名を参照*)。

サンプル

- `altova:weekday-from-date(xs:date("2014-02-03"), 1)` は月曜を示す1 を返します。
- `altova:weekday-from-date(xs:date("2014-02-03"), 4)` は月曜を示す1 を返します。
- `altova:weekday-from-date(xs:date("2014-02-03"), 0)` は月曜を示す2 を返します。

[Top]

xs:dateTime または xs:date から週数を返す XP2 XQ1 XP3 XQ3

これらの関数は週数 (整数) としてを xs:dateTime から xs:date. から返します。週の番号付けは、米国、欧州、イスラムのカレンダーフォーマットで使用することができます。週の始まりが異なるため、週数の番号付けは、カレンダーのフォーマットにより異なります。(米国フォーマットでは、日曜、欧州フォーマットでは月曜、イスラムフォーマットでは土曜が週の開始日です)。

weeknumber-from-date [altova:]

altova:weeknumber-from-date(Date *as xs:date*, **Calendar** *as xs:integer*) を **xs:integer** とする **XP2 XQ1 XP3 XQ3**
正数として提出された **Date** 引数の週数を返します。第 2 の引数 (**カレンダー**) は続くカレンダーシステムを指定します。
サポートされる **カレンダー** の値は次のとおりです :

- **0 = US 米国のカレンダー** (週の始まりは日曜日)
- **1 = ISO 標準、 欧州のカレンダー** (週の始まりは月曜日)
- **2 = イスラムのカレンダー** (週の始まりは土曜日)

デフォルトは **0** です。

サンプル

- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 0) は 13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 1) は 13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 2) は 13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23")) は 13 を返します。

上のサンプル (2014-03-23) の date の曜日は日曜日です。米国およびイスラムのカレンダーは欧州カレンダーのこの日付よりも一週間先です。

weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(DateTime *as xs:dateTime*, **Calendar** *as xs:integer*) を **xs:integer** とする **XP2 XQ1 XP3 XQ3**
正数として提出された **DateTime** 引数の週数を返します。第 2 の引数 (**カレンダー**) は続くカレンダーシステムを指定します。
サポートされる **カレンダー** の値は次のとおりです :

- **0 = US 米国のカレンダー** (週の始まりは日曜日)
- **1 = ISO 標準、 欧州のカレンダー** (週の始まりは月曜日)
- **2 = イスラムのカレンダー** (週の始まりは土曜日)

Default is **0**.

サンプル

- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 0) は 13 を返します。
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 1) は 13 を返します。
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 2) は 13 を返します。
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00")) は 13 を返します。

上のサンプル (2014-03-23T00:00:00) の dateTime の曜日は日曜日です。米国およびイスラムのカレンダーは欧州カレンダーのこの日付よりも一週間先です。

[トップ]

各型の構文コンポーネントから日付、時刻、期間の型を構築する XP3 XQ3

関数は、xs:date, xs:time または xs:duration の構文コンポーネントを入力引数とし、引数を結合させて対応するデータ型を構築します。

build-date [altova:]

altova:build-date(Year *as xs:integer*, **Month** *as xs:integer*, **Date** *as xs:integer*) を **xs:date** とする XP3 XQ3

第 1、第 2、第 3 の引数は、それぞれ、年、月、日を表します。xs:date 型の値を構築するため結合されます。整数の値は、特定の日付の一部の適正な範囲内である必要があります。例えば第 2 の引数 (月の部分) は 12 以上であってはなりません。

サンプル

- **altova:build-date**(2014, 2, 03) は 2014-02-03 を返します。

build-time [altova:]

altova:build-time(Hours *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*) を **xs:time** とする XP3 XQ3

第 1、第 2、第 3 引数はそれぞれ時間数 (0 から 23)、分数 (0 から 59)、および秒数 (0 から 59) の値です。これらの値は、xs:time 型の値を作成するために結合されます。整数の値は、それぞれの部分の正しい範囲内である必要があります。例えば、秒 (分) 引数は、59 以上であってはなりません。値にタイムゾーンを追加するには、この関数の他の署名を使用してください (*次の署名を参照*)。

サンプル

- **altova:build-time**(23, 4, 57) は 23:04:57 を返します。

altova:build-time(Hours *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*, **TimeZone** *as xs:string*) を **xs:time** とする XP3 XQ3

第 1、第 2、第 3 引数はそれぞれ時間数 (0 から 23)、分数 (0 から 59)、および秒数 (0 から 59) の値です。第四の引数は、値の一部としてタイムゾーンを与えます。4 つの引数が結合され、xs:time 型の値を構築します。例えば、第 2 の引数 (分数) は 59 以上であってはなりません。

サンプル

- **altova:build-time**(23, 4, 57, '+1') は 23:04:57+01:00 を返します。

build-duration [altova:]

altova:build-duration(Years *as xs:integer*, **Months** *as xs:integer*) を **xs:yearMonthDuration** とする XP3 XQ3

xs:yearMonthDuration 型の値を構築するためには 2 つの引数が必要です。最初の引数は期間の年数 値の部分を与え、第 2 の引数は、月数 値の部分を与えます。第 2 の引数 (月数) が同じまたはより大きくなると、整数は 12 により分割されます。 商は、年数 の部分を表す 最初の引数に加算され、除算の残りは部分は月数の部分を表すために使用されます。期間の xs:dayTimeDuration 型を作成するには、次の署名を参照してください。

サンプル

- **altova:build-duration**(2, 10) は P2Y10M を返します。
- **altova:build-duration**(14, 27) は P16Y3M を返します。
- **altova:build-duration**(2, 24) は P4Y を返します。

altova:build-duration(Days *as xs:integer*, **Hours** *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*) を **xs:dayTimeDuration** とする XP3 XQ3

xs:dayTimeDuration 型の値を構築するためには 4 つの引数が必要です。最初の引数は期間の日数 値の部分で、第 2、第 3、第 4 引数は、それぞれ期間の時間数、分数、秒数 値のです。これらの 3 つの時間 引数

は、次の高い単位の値に加算され、結果は期間の全体の値を計算するために使用されます。例えば、72秒は、1M+12S (1 分 と12秒)に変換され、この値が期間全体の値を計算するために使用されます。
xs:yearMonthDuration 型の期間を構築するためには、次の署名を参照してください。

サンプル

- **altova:build-duration**(2, 10, 3, 56) は P2DT10H3M56S を返します。
- **altova:build-duration**(1, 0, 100, 0) は P1DT1H40M を返します。
- **altova:build-duration**(1, 0, 0, 3600) は P1DT1H を返します。

[トップ]

文字列入力から日付、日付時刻 または 時刻 を構築する XP2 XQ1 XP3 XQ3

関数は、文字列を引数として、xs:date、xs:dateTime または xs:time データ型を構築します。文字列は、データ型のコンポーネントを提出されたパターン引数をベースにして分析されます。

parse-date [altova:]

altova:parse-date(Date *as xs:string*, **DatePattern** *as xs:string*) を **xs:date** とする XP2 XQ1 XP3 XQ3

Date 入力文字列を **xs:date** の値として返します。第二の引数である **DatePattern** は、入力文字列のパターン (コンポーネントのシーケンス)を指定します。**DatePattern** は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネント セパレータと共に説明されています。下のサンプルを参照してください。

D 日付
M 月
Y 年

DatePattern のパターンは、**Date** のパターンに一致する必要があります。出力は **xs:date** 型であるため、出力は常に **YYYY-MM-DD** 構文フォーマットになります。

サンプル

- **altova:parse-date**(xs:string("06-03-2014"), "[D]-[M]-[Y]") は 2014-03-06 を返します。
- **altova:parse-date**(xs:string("06-03-2014"), "[M]-[D]-[Y]") は 2014-03-06 を返します。
- **altova:parse-date**("06/03/2014", "[M]/[D]/[Y]") は 2014-03-06 を返します。
- **altova:parse-date**("06 03 2014", "[M] [D] [Y]") は 2014-03-06 を返します。
- **altova:parse-date**("6 3 2014", "[M] [D] [Y]") は 2014-03-06 を返します。

parse-dateTime [altova:]

altova:parse-dateTime(DateTime *as xs:string*, **DateTimePattern** *as xs:string*) を **xs:dateTime** とする XP2 XQ1 XP3 XQ3

DateTime 入力文字列を **xs:dateTime** の値として返します。第二の引数である **DateTimePattern** は、入力文字列のパターン (コンポーネントのシーケンス)を指定します。**DateTimePattern** は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネント セパレータと共に説明されています。下のサンプルを参照してください。

D 日
M 月
Y 年
H 時間

m	分
s	秒

DateTimePattern のパターンは、DateTime のパターンに一致する必要があります。出力は xs:dateTime 型であるため、出力は常に YYYY-MM-DDTHH:mm:ss 構文フォーマットになります。

サンプル

- altova:parse-dateTime(xs:string("06-03-2014 13:56:24"), "[D]-[M]-[Y] [H]:[m]:[s]") は 2014-03-06T13:56:24 を返します。
- altova:parse-dateTime("time=13:56:24; date=06-03-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]") は 2014-03-06T13:56:24 を返します。

parse-time [altova:]

altova:parse-time(Time *as xs:string*, TimePattern *as xs:string*) を xs:time とする XP2 XQ1 XP3 XQ3

Time 入力文字列を xs:time の値として返します。第二の引数である TimePattern は、入力文字列のパターン (コンポーネントのシーケンス) を指定します。TimePattern は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネントセパレータと共に説明されています。下のサンプルを参照してください。

H	時間
m	分
s	秒

TimePattern のパターンは、Time のパターンに一致する必要があります。出力は xs:time 型であるため、出力は常に YYYY-HH:mm:ss 構文フォーマットになります。

サンプル

- altova:parse-time(xs:string("13:56:24"), "[H]:[m]:[s]") は 13:56:24 を返します。
- altova:parse-time("13-56-24", "[H]-[m]") は 13:56:00 を返します。
- altova:parse-time("time=13h56m24s", "time=[H]h[m]m[s]s") は 13:56:24 を返します。
- altova:parse-time("time=24s56m13h", "time=[s]s[m]m[H]h") は 13:56:24 を返します。

[トップ]

年齢に関連した関数 XP3 XQ3

関数は計算された年齢 (i) 入力引数の日付と現在の日付の期間 (i) 2 つの入力引数の日付の期間 を返します。altova:age 関数は、年齢を年数で返します、altova:age-details は年齢を年数、月数、日数からなるを3つの整数のシーケンスで返します。

age [altova:]

altova:age(StartDate *as xs:date*) を xs:integer とする XP3 XQ3

引数として提出された開始日からシステムクロックから取得された現在の日付 までの日数を数えて、あるオブジェクトの年齢の年数である正数を返します。入力引数が 1年よりも大きい場合、または一年の場合、返される値は負の数です。

サンプル

If the current date is 2014-01-15:

- altova:age(xs:date("2013-01-15")) は 1 を返します。
- altova:age(xs:date("2013-01-16")) は 0 を返します。
- altova:age(xs:date("2015-01-15")) は -1 を返します。
- altova:age(xs:date("2015-01-14")) は 0 を返します。

altova:age(StartDate *as xs:date*, **EndDate** *as xs:date*) を **xs:integer** とする **XP3 XQ3**

最初の引数として提出された開始日から第 2 の引数の終了日までの日数を数えて、あるオブジェクトの年齢の年数である正数を返します。最初の引数が 1 年または第 2 の引数よりも後の日付の場合、返される値は負の数です。

サンプル

If the current date is 2014-01-15:

- altova:age(xs:date("2000-01-15"), xs:date("2010-01-15")) は 10 を返します。
- altova:age(xs:date("2000-01-15"), current-date()) は、現在の日付が 2014-01-15 の場合 14 を返します。
- altova:age(xs:date("2014-01-15"), xs:date("2010-01-15")) は -4 を返します。

age-details [altova:]

altova:age-details(InputDate *as xs:date*) を (**xs:integer**)* とする **XP3 XQ3**

引数とシステムクロックから取得された現在の日付として提出された日付の間の年数、月数、日数の 3 つの整数を返します。返された years+months+days の合計は、2 つの日付 (入力の日付と現在の日付) の時間差です。入力の日付は現在の日付よりも早いまたは遅い値をもつことができますが、入力の日付が早いか遅いかは返される値で示されていません。戻される値は常に正の数です。

サンプル

現在の日付が 2014-01-15 の場合 :

- altova:age-details(xs:date("2014-01-16")) は (0 0 1) を返します。
- altova:age-details(xs:date("2014-01-14")) は (0 0 1) を返します。
- altova:age-details(xs:date("2013-01-16")) は (1 0 1) を返します。
- altova:age-details(current-date()) は (0 0 0) を返します。

altova:age-details(Date-1 *as xs:date*, **Date-2** *as xs:date*) を (**xs:integer**)* とする **XP3 XQ3**

二つの引数間の年数、月数、日数の 3 つの整数を返します。返された years+months+days の合計は、2 つの入力の日付の時間差です。最初の引数として提出される二つの日付はどちらが速くても、また遅くてもかまいません。返される値は入力の日付が現在の日付より早いかまた遅いかを示しません。戻される値は常に正の数です。

サンプル

- altova:age-details(xs:date("2014-01-16"), xs:date("2014-01-15")) は (0 0 1) を返します。
- altova:age-details(xs:date("2014-01-15"), xs:date("2014-01-16")) は (0 0 1) を返します。

[Top]

2.1.3 XPath/ XQuery 関数 :位置情報

以下の位置情報 XPath/XQuery 拡張関数は、XMLSpy の現在のバージョンによりサポートされています。また、次で使用することができます : (i) XSLT コンテキスト内の XPath 式、または (i) XQuery ドキュメント内の XQuery

式。

関数の名前指定と言語の適用性

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、** `http://www.altova.com/xslt-extensions` に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3

parse-geolocation [altova:]

altova:parse-geolocation(GeolocationInputString *as xs:string*) を**xs:decimal+** とする XP3 XQ3

GeolocationInputString 引数を解析して、位置情報の緯度と経度 (この通りの順番) を 2 つの xs:decimal アイテムのシーケンスとして返します。位置情報入力文字列が提供されることのできるフォーマットは以下のリストの通りです。

メモ: image-exif-data 関数とExif メタデータの @Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:parse-geolocation**("33.33 -22.22") は 2 つのxs:decimals (33.33, 22.22) のシーケンスを返します。
- **altova:parse-geolocation**("48°51'29.6""N 24°17'40.2""") は 2 つの xs:decimals (48.8582222222222, 24.2945) のシーケンスを返します。
- **altova:parse-geolocation**('48°51''29.6"N 24°17''40.2"') は 2 つの xs:decimals (48.8582222222222, 24.2945) のシーケンスを返します。
- **altova:parse-geolocation**(**image-exif-data**(//MyImages/Image20141130.01)**/@Geolocation**) は 2 つの xs:decimals のシーケンスを返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (北 から南) です。経度の値の範囲は +180 から -180 (東 から西) です。

メモ: 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、 10進の秒、方角のサフィックス付き (N/S, W/E)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'66.66"W**

- 度、分、 10進の秒、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。

`+/-D°M'S.SS" +/-D°M'S.SS"`
サンプル: `33°55'11.11" -22°44'66.66"`

- 度、分、 10進の分、方角のサフィックス付き (`N/S`, `W/E`)
  `D°M.MM'N/S D°M.MM'W/E`
  サンプル: `33°55.55'N 22°44.44'W`

- 度、分、 10進の分、プレフィックスサイン付き (`+/-`); (`N/W`) のためのプラスサインは任意です。
  `+/-D°M.MM' +/-D°M.MM'`
  サンプル: `+33°55.55' -22°44.44'`

- 10 進の度、方角のサフィックス付き (`N/S`, `W/E`)
  `D.DDN/S D.DDW/E`
  サンプル: `33.33N 22.22W`

- 10 進の度、プレフィックスサイン付き (`+/-`); (`N/W`) のためのプラスサインは任意です。
  `+/-D.DD +/-D.DD`
  サンプル: `33.33 -22.22`

フォーマットの組み合わせのサンプル:
```
33.33N -22°44'66.66"
33.33 22°44'66.66"W
33.33 22.c
```

▫ *Altova Exif 属性 :位置情報*

Altova XPath/XQuery エンジンはカスタム属性 `Geolocation` を標準 Exif メタデータタグから生成します。`Geolocation` は、4 つの Exif タグの連結です :単位の追加された (下のテーブル参照) `GPSLatitude`、`GPSLatitudeRef`、`GPSLongitude`、`GPSLongitudeRef`。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33° 51'21.91"S 151° 13'11.73"E

▾ **geolocation-distance-km [altova:]**

altova:geolocation-distance-km(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) を **xs:decimal** とする **XP3 XQ3**
2 つの位置情報の間の距離をキロメートルで計算します。位置情報入力文字列を提供することのできるフォーマットは下にリストされています。緯度の値の範囲は `+90` から `-90` (北 から南) です。経度の値の範囲は `+180` から `-180` (東 から西) です。

メモ: `image-exif-data` 関数と Exif メタデータの `@Geolocation` 属性を位置情報入力文字列を提供する際に使用することができます。

▫ *サンプル*

- `altova:geolocation-distance-km("33.33 -22.22", "48°51'29.6""N 24°17'40.2""")` は `xs:decimal 4183.08132372392` を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は `+90` から `-90` (北 から南) です。経度の値の範囲は `+180` から `-180` (東 から西) です。

メモ: 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(`"`) でハイライトされており、エスケープした単位インジケーターは青い(`""`) でハイライトされています。

- 度、分、 10進の秒、方角のサフィックス付き (`N/S`, `W/E`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  サンプル: `33°55'11.11"N  22°44'66.66"W`

- 度、分、 10進の秒、プレフィックスサイン付き (`+/-`); (`N/W`) のためのプラスサインは任意です。
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  サンプル: `33°55'11.11"  -22°44'66.66"`

- 度、分、 10進の分、方角のサフィックス付き (`N/S`, `W/E`)
  `D°M.MM'N/S  D°M.MM'W/E`
  サンプル: `33°55.55'N  22°44.44'W`

- 度、分、 10進の分、プレフィックスサイン付き (`+/-`); (`N/W`) のためのプラスサインは任意です。
  `+/-D°M.MM'  +/-D°M.MM'`
  サンプル: `+33°55.55'  -22°44.44'`

- 10 進の度、方角のサフィックス付き (`N/S`, `W/E`)
  `D.DDN/S  D.DDW/E`
  サンプル: `33.33N  22.22W`

- 10 進の度、プレフィックスサイン付き (`+/-`); (`N/W`) のためのプラスサインは任意です。
  `+/-D.DD  +/-D.DD`
  サンプル: `33.33  -22.22`

フォーマットの組み合わせのサンプル:

```
33.33N  -22°44'66.66"
33.33  22°44'66.66"W
33.33  22.c
```

Altova Exif 属性 :位置情報

Altova XPath/XQuery エンジンはカスタム属性 `Geolocation` を標準 Exif メタデータタグから生成します。`Geolocation` は、4つの Exif タグの連結です :単位の追加された (下のテーブル参照)
`GPSLatitude`、`GPSLatitudeRef`、`GPSLongitude`、`GPSLongitudeRef`。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33° 51'21.91"S 151° 13'11.73"E

geolocation-distance-mi [altova:]

altova:geolocation-distance-mi(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) を**xs:decimal** とする **XP3 XQ3**

2 つの位置情報の間の距離をマイルで計算します。位置情報入力文字列を提供することのできるフォーマットは下にリストされています。緯度の値の範囲は +90 から -90 (北 から南) です。経度の値の範囲は +180 から -180 (東 から西) です。

メモ: image-exif-data 関数と Exif メタデータの @Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。.

サンプル

- **altova:geolocation-distance-mi**("33.33 -22.22", "48°51'29.6""N 24° 17'40.2""") は xs:decimal 2599.40652340653 を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (北 から南) です。経度の値の範囲は +180 から -180 (東 から西) です。

メモ: 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、 10進の秒、方角のサフィックス付き (N/S, W/E)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'66.66"W**

- 度、分、 10進の秒、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'66.66"**

- 度、分、 10進の分、方角のサフィックス付き (N/S, W/E)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、 10進の分、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, W/E)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:

```
33.33N  -22°44'66.66"
33.33  22°44'66.66"W
33.33  22.c
```

Altova Exif 属性 :位置情報

Altova XPath/XQuery エンジンはカスタム属性 `Geolocation` を標準 Exif メタデータタグから生成します。`Geolocation` は、4 つの Exif タグの連結です :単位の追加された (下のテーブル参照) `GPSLatitude`、`GPSLatitudeRef`、`GPSLongitude`、`GPSLongitudeRef`。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33° 51'21.91"S 151° 13'11.73"E

geolocation-within-polygon [altova:]

altova:geolocation-within-polygon(Geolocation *as xs:string*, **((PolygonPoint** *as xs:string*)+)) を **xs:boolean** とする **XP3 XQ3**

`PolygonPoint` 引数により説明されている `Geolocation` (最初の引数) が多角形のエリア内に存在するかを決定します。もし、`PolygonPoint` 引数が最初と最後のポイントが同じ場合に作成される閉じられたフィギュアを作成しない場合、フィギュアを閉じるために、最初のポイントが明示的に最後のポイントとして追加されます。全ての引数 (`Geolocation` および `PolygonPoint+`) は、(下にリストされるフォーマットの) 位置情報入力文字列により提供されます。`Geolocation` 引数が多角形エリア内にある場合、関数は `true()`; を返します。その他の場合は `false()` を返します。緯度の値の範囲は `+90` から `-90` (北 から南) です。経度の値の範囲は `+180` から `-180` (東 から西) です。

メモ: `image-exif-data` 関数と Exif メタデータの `@Geolocation` 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24", "58 -32"))** は `true()` を返します。
- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24"))** は `true()` を返します。
- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48°51'29.6""N 24°17'40.2""))** は `true()` を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が 1 つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は `+90` から `-90` (北 から南) です。経度の値の範囲は `+180` から `-180` (東 から西) です。

メモ: 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケータ

ーは青い("") でハイライトされています。

- 度、分、 10進の秒、方角のサフィックス付き (N/S, W/E)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: 33°55'11.11"N 22°44'66.66"W

- 度、分、 10進の秒、プレフィックスサイン付き (+/-); (N/W) のためのプラスサインは任意です。
  +/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: 33°55'11.11" -22°44'66.66"

- 度、分、 10進の分、方角のサフィックス付き (N/S, W/E)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: 33°55.55'N 22°44.44'W

- 度、分、 10進の分、プレフィックスサイン付き (+/-); (N/W) のためのプラスサインは任意です。
  +/-D°M.MM' +/-D°M.MM'
  サンプル: +33°55.55' -22°44.44'

- 10 進の度、方角のサフィックス付き (N/S, W/E)
  D.DDN/S D.DDW/E
  サンプル: 33.33N 22.22W

- 10 進の度、プレフィックスサイン付き (+/-); (N/W) のためのプラスサインは任意です。
  +/-D.DD +/-D.DD
  サンプル: 33.33 -22.22

フォーマットの組み合わせのサンプル:

```
33.33N  -22°44'66.66"
33.33  22°44'66.66"W
33.33  22.c
```

Altova Exif 属性 :位置情報

Altova XPath/XQuery エンジンはカスタム属性 Geolocation を標準 Exif メタデータタグから生成します。Geolocation は、4つの Exif タグの連結です :単位の追加された (下のテーブル参照) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33° 51'21.91"S 151° 13'11.73"E

geolocation-within-rectangle [altova:]

altova:geolocation-within-rectangle(Geolocation *as xs:string*, **RectCorner-1** *as xs:string*, **RectCorner-2** *as xs:string*) を **xs:boolean** とする **XP3 XQ3**
長方形の対格の角を指定する第 2及び第 3引数 (RectCorner-1、RectCorner-2) により説明されている Geolocation (最初の引数) が長方形のエリア内に存在するかを決定します。全ての引数 (Geolocation、RectCorner-1 および RectCorner-2) は、 (下にリストされるフォーマットの) 位置情報入力文字列により提供されます。Geolocation 引数が長方形エリア内にある場合、関数は true(); を返します。その他の場合は

false() を返します。緯度の値の範囲は +90 から -90 (北 から南)です。経度の値の範囲は +180 から -180 (東 から西)です。

メモ: image-exif-data 関数とExif メタデータの @Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-within-rectangle("33 -22", "58 -32", "-48 24")** は true() を返します。
- **altova:geolocation-within-rectangle("33 -22", "58 -32", "48 24")** は false() を返します。
- **altova:geolocation-within-rectangle("33 -22", "58 -32", "48°51'29.6""S 24°17'40.2""")** は true() を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番)を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (北 から南)です。経度の値の範囲は +180 から -180 (東 から西)です。

メモ: 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、 10進の秒、方角のサフィックス付き (N/S, W/E)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'66.66"W**

- 度、分、 10進の秒、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'66.66"**

- 度、分、 10進の分、方角のサフィックス付き (N/S, W/E)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、 10進の分、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, W/E)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:

```
33.33N  -22°44'66.66"
33.33  22°44'66.66"W
```

33.33 22.c

- *Altova Exif 属性 :位置情報*

  Altova XPath/XQuery エンジンはカスタム属性 `Geolocation` を標準 Exif メタデータタグから生成します。`Geolocation` は、4 つの Exif タグの連結です :単位の追加された (下のテーブル参照) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33° 51'21.91"S 151° 13'11.73"E

[Top]

2.1.4 XPath/ XQuery 関数 :イメージに関連

以下のイメージに関連した XPath/XQuery 拡張関数は、XMLSpy の現在のバージョンによりサポートされています。また、次で使用することができます : (i) XSLT コンテキスト内の XPath 式、または (i) XQuery ドキュメント内の XQuery 式。

関数の名前指定と言語の適用性

Altova 拡張関数は XPath/XQuery 式で使用することができ、XPath、XQuery、および XSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は **Altova 拡張関数名前空間、`http://www.altova.com/xslt-extensions`** に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意ください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内の XPath 式で使用):	XP1 XP2 XP3
XSLT 関数 (XSLT 内の XPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内の XQuery 式で使用):	XQ1 XQ3

suggested-image-file-extension [altova:]

`altova:suggested-image-file-extension(Base64String as string)` を `string?` とする XP3 XQ3

イメージファイルの Base64 エンコードを引数として、イメージのファイル拡張子を、イメージの Base64 エンコード内の記録として返します。返された値は、エンコード内で使用することのできるイメージ型情報を基にしたヒントです。この情報が使用できない場合、空の文字列が返されます。この関数は、Base64 イメージをファイルとして保存し、適切なファイル拡張子を動的取得する際に役に立ちます。

- *サンプル*
  - `altova:suggested-image-file-extension(/MyImages/MobilePhone/Image20141130.01)` は `'jpg'` を返します。
  - `altova:suggested-image-file-extension($XML1/Staff/Person/@photo)` は `''` を返します。

上のサンプルでは、関数の引数として与えられたノードはBase64 エンコードイメージを含むと仮定します。最初のサンプルは jpg をファイルの型および拡張子として取得します。二番目のサンプルでは、与えられたBase64 エンコードは使用できる拡張子の情報を提供しません。

mt-transform-image [altova:]

altova:mt-transform-image(Base64Image *as Base64BinaryString*, **Size** *as item()+*, **Rotation** *as xs:integer*, **Quality** *as xs:integer*) を**Base64BinaryString** とする
XP3 XQ3

Base64-エンコードイメージを最初の引数として、変換されたBase64-エンコードイメージを返します。第 2、第 3、第 4引数は変換された以下のイメージパラメーターです :サイズ、回転、およびクオリティ。

- サイズ引数には 3 つのサイズ変更のオプションがあります。

(X,Y)	絶対ピクセルの値。アスペクト率は保持されません。高さ幅は自動的にイメージの長いおよび短い辺に逢わされるため、高さ幅の指示は関係ありません。 2 つの整数アイテムのシーケンスとして値が入力されます。かっこが必要です。
X	X をピクセルの新しい長い辺として、イメージの縦横比のサイズ変更が行われます。アスペクト率は保持されます。値は整数で引用符なしで入力されます。
'X%'	元のディメンションの与えられたパーセンテージにイメージのサイズを変更します。値は文字列として引用符を付けて入力される必要があります。

- 回転は以下の値を持つことができます :90、180、270、-90、-180、-270。これらの値は回転の度数です。正の値はイメージを時計回りに回転させます。負の値はイメージを反時計回りに回転させます。Altova Exif 属性 **OrientationDegree** を使用して、イメージの現在の回転の度数 (0, 90, 180, 270) をイメージのExif **Orientation** タグから取得することができます。ですが、**OrientationDegree** 属性はデータの**Orientation** タグから取得されるため、Exif データ内に **Orientation** タグが存在する場合のみ使用することができます。 (*下の OrientationDegree 説明を参照してください*)。
- クオリティは、0 から100 の値で、JPEG 圧縮の IJG クオリティスケールの値を参照していますが、クオリティのパーセンテージのインジケーターではありません。サイズとクオリティのどちらかを優先すると、もう一方の優先度が下がります。フルカラーソースのイメージでは、75 が通常最高値と見なされています。もし、75 が満足のいく結果をもたらさない場合は、値を上げてください。

メモ: Exif データが元のイメージに存在する場合、変換時に削除され、変換されたイメージにはExif データは存在しません。

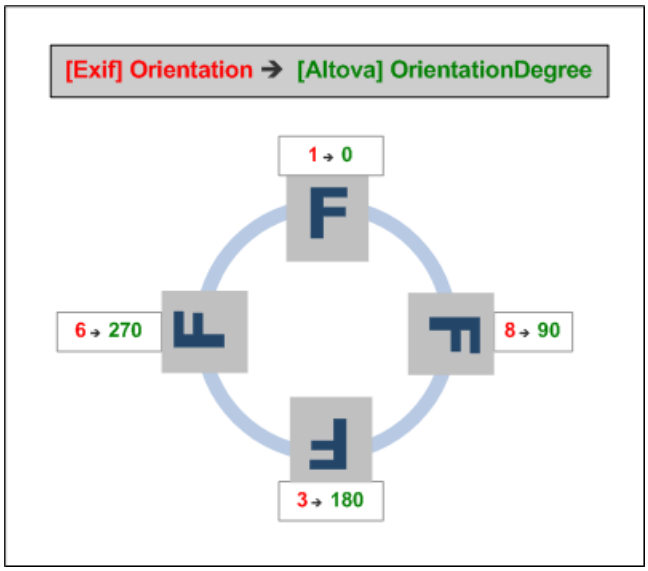
サンプル

- **mt-transform-image**(Images/Image[@id='43'], '50%', 90, 75)
  関数は、43 の@id 値を持つイメージ子孫ノード内でBase64-エンコード文字列として保管されているイメージを入力とします。関数は変換されたイメージを返します。変換されたイメージは 50% まで、サイズ変更され、時計回りに 90 度回転され、 75 のクオリティレベルを与えられます。
- **mt-transform-image**(Images/Image[@id='43'], 400, 90, 75)
  関数は前のサンプルと同じ結果を出しますが、長い辺は400 ピクセルの特定の値に設定されています。元のイメージのアスペクト率は保持されます。
- **mt-transform-image**(Images/Image[@id='43'], (400, 280), image-exif-data($XML1/$XML1/Images/ReferenceImage)/@OrientationDegree, 75)
  このサンプルは、前のサンプルと同じイメージを選択し、同じクオリティ値 (75) を設定します。イメージのサイズは400x280 ピクセルに設定されています。回転値は、ノード内のBase64-エンコードイメージの **@OrientationDegree** 属性から得られます。

Altova Exif 属性 :OrientationDegree

Altova XPath/XQuery エンジンはカスタム属性 **OrientationDegree** を Exif メタデータタグ **Orientation** から生成します。

OrientationDegree は標準 Exif タグ Orientation を正数の値 (1、8、3 または 6) からそれぞれ対応する度数の値 (0, 90, 180, 270) へ下の図に示されているように変換します。2、4、5、7 の Orientation 値の変換はないことに注意してください。(これらの向きはイメージ 1 を垂直方向中央軸で反転して、2 の値を持つイメージを取得します。そして、このイメージを 90 度ごとに時計回りにジャンプさせそれぞれ 7、4、および 5 の値を取得します。)



標準 Exif メタタグ

- ImageWidth
- ImageLength
- BitsPerSample
- Compression
- PhotometricInterpretation
- Orientation
- SamplesPerPixel
- PlanarConfiguration
- YCbCrSubSampling
- YCbCrPositioning
- XResolution
- YResolution
- ResolutionUnit
- StripOffsets
- RowsPerStrip
- StripByteCounts
- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients

- ReferenceBlackWhite
- DateTime
- ImageDescription
- Make
- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion
- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance

- DigitalZoomRatio
- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

image-exif-data [altova:]

altova:image-exif-data(Base64BinaryString *as string*) を**element?** とする **XP3 XQ3**

Base64 エンコードイメージを引数として、イメージのExif *メタデータ*を含む**Exif** という名の要素を返します。The Exif *メタデータ*は`Exif` 要素の属性の値ペアとして作成されます。属性名は、Base64エンコード内で検出された Exif *データタグ*です。Exif 仕様タグのリストは以下の通りです。ベンダー特有のタグがExif データ内に存在する場合、タグとその値も属性値のペアとして返されます。標準 Exif *メタデータタグ*に追加して (*下のリスト参照*) Altova 特有の属性値のペアもまた生成されます。これらのAltova Exif 属性は以下のとおりです。

サンプル

- 属性にアクセスするには、以下の関数を使用します :
  ```
  image-exif-data(//MyImages/Image20141130.01)/@GPSLatitude
  image-exif-data(//MyImages/Image20141130.01)/@Geolocation
  ```
- 全ての属性にアクセスするには、以下の関数を使用します :
  ```
  image-exif-data(//MyImages/Image20141130.01)/@*
  ```
- 全ての属性の名前にアクセスするには、以下の式を使用します :
  ```
  for $i in image-exif-data(//MyImages/Image20141130.01)/@* return
  name($i)
  ```
  関数により返される属性の名前を検出するために役に立ちます。

Altova Exif 属性 :位置情報

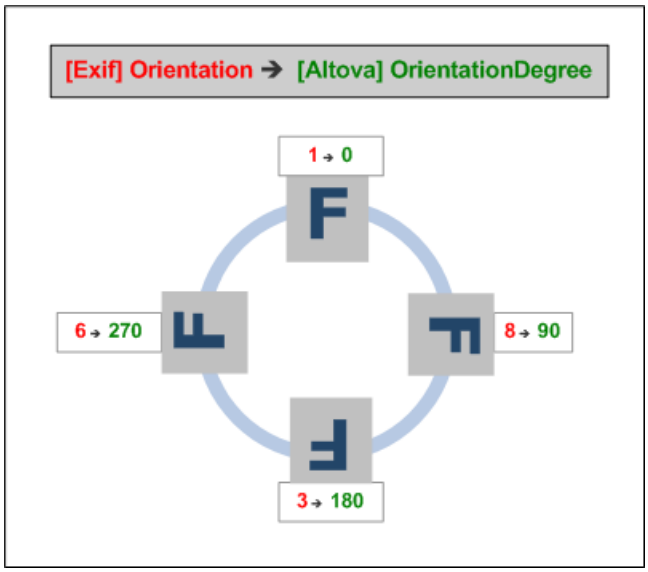
Altova XPath/XQuery エンジンはカスタム属性 **Geolocation** を標準 Exif メタデータタグから生成します。**Geolocation** は、4 つの Exif タグの連結です :単位の追加された (下のテーブル参照) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33° 51'21.91"S 151° 13'11.73"E

Altova Exif 属性 :OrientationDegree

Altova XPath/XQuery エンジンはカスタム属性 **OrientationDegree** を Exif メタデータタグ **Orientation** から生成します。

OrientationDegree は標準 Exif タグ Orientation を正数の値 (1、8、3 または 6) からそれぞれ対応する度数の値 (0, 90, 180, 270) へ下の図に示されているように変換します。2、4、5、7 の Orientation 値の変換はないことに注意してください。(これらの向きはイメージ 1 を垂直方向中央軸で反転して、2 の値を持つイメージを取得します。そして、このイメージを 90 度ごとに時計回りにジャンプさせそれぞれ 7、4、および 5 の値を取得します。)



- *標準 Exif メタデータタグのリスト*
  - ImageWidth
  - ImageLength
  - BitsPerSample
  - Compression
  - PhotometricInterpretation
  - Orientation
  - SamplesPerPixel
  - PlanarConfiguration
  - YCbCrSubSampling
  - YCbCrPositioning
  - XResolution
  - YResolution
  - ResolutionUnit
  - StripOffsets
  - RowsPerStrip
  - StripByteCounts
  - JPEGInterchangeFormat
  - JPEGInterchangeFormatLength
  - TransferFunction
  - WhitePoint
  - PrimaryChromaticities
  - YCbCrCoefficients
  - ReferenceBlackWhite
  - DateTime
  - ImageDescription
  - Make
  - Model
  - Software
  - Artist
  - Copyright

  - ExifVersion
  - FlashpixVersion

- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio
- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

```
------------------------------
```

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

[トップ]

2.1.5 XPath/ XQuery 関数 :数値

Altova の数値拡張関数はXPath とXQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の**XPath 3.0** および**XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、** `http://www.altova.com/xslt-extensions` に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意ください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3

自動付番関数

generate-auto-number [altova:]

altova:generate-auto-number(ID *as xs:string*, **StartsWith** *as xs:double*, **Increment** *as xs:double*, **ResetOnChange** *as xs:string***)** を xs:integer とする XP1 XP2 XQ1 XP3 XQ3

関数が呼び出される度に番号を生成します。関数が初めて呼び出される際に生成される最初の番号は StartsWith 引数により指定されます。その後の関数の呼び出しは新しい番号を生成します。この番号は前に生成された番号を、Increment 引数で指定された値ごとにインクリメントします。実質的には、altova:generate-auto-number 関数は、ID 引数により指定されたカウンターを作成し、このカウンターが関数が呼び出されるごとにインクリメントされます。ResetOnChange 引数の値が、前の関数呼び出しから変更されると、生成される番号の値が StartsWith 値にリセットされます。自動付番は altova:reset-auto-number 関数を使用してリセットすることができます。

サンプル

- **altova:generate-auto-number**("ChapterNumber", 1, 1, "SomeString") は、関数が呼び出される度に番号を 1 つ返します。starting with 1, から始まり、関数が呼び出される度に 1 ずつインクリメントします。その後の呼び出しの 4 番目の'SomeString"引数がである限り、インクリメントは継続されます。4 番目の引数の値が変更されると、(ChapterNumber と呼ばれる)カウンターは 1 にリセットされます。ChapterNumber の値も altova:reset-auto-number 関数の呼び出しにより altova:reset-auto-number("ChapterNumber") ようにリセットされます。

reset-auto-number [altova:]

altova:reset-auto-number(ID *as xs:string***)** XP1 XP2 XQ1 XP3 XQ3

この関数は、ID 引数内で名づけられた自動付番カウンターの番号をリセットします。引数内のカウンター名を作成した関数の引数により指定された数値にリセットされます。altova:generate-auto-number

サンプル

- **altova:reset-auto-number**("ChapterNumber") は、altova:generate-auto-number 関数により作成された ChapterNumber という名の自動付番カウンターをリセットします。
- ChapterNumber を作成した altova:generate-auto-number 関数の StartsWith 引数の値に数値をリセットします。

[Top]

数値関数

hex-string-to-integer [altova:]

altova:hex-string-to-integer(HexString *as xs:string***)** を **xs:integer** とする XP3 XQ3

10 進のシステム(10進)内の正数の16進の同値の文字列引数を必要とし、10進の整数を返します。

サンプル

- **altova:hex-string-to-integer**('1') は 1 を返します。
- **altova:hex-string-to-integer**('9') は 9 を返します。
- **altova:hex-string-to-integer**('A') は 10 を返します。
- **altova:hex-string-to-integer**('B') は 11 を返します。
- **altova:hex-string-to-integer**('F') は 15 を返します。

- altova:hex-string-to-integer('G') はエラーを返します。
- altova:hex-string-to-integer('10') は 16 を返します。
- altova:hex-string-to-integer('01') は 1 を返します。
- altova:hex-string-to-integer('20') は 32 を返します。
- altova:hex-string-to-integer('21') は 33 を返します。
- altova:hex-string-to-integer('5A') は 90 を返します。
- altova:hex-string-to-integer('USA') はエラーを返します。

integer-to-hex-string [altova:]

altova:integer-to-hex-string(Integer *as xs:integer*) を xs:string とする XP3 XQ3
正数を引数として必要とし、文字列として自身のベース16の同値を返します。

サンプル

- altova:integer-to-hex-string(1) は '1' を返します。
- altova:integer-to-hex-string(9) は '9' を返します。
- altova:integer-to-hex-string(10) は 'A' を返します。
- altova:integer-to-hex-string(11) は 'B' を返します。
- altova:integer-to-hex-string(15) は 'F' を返します。
- altova:integer-to-hex-string(16) は '10' を返します。
- altova:integer-to-hex-string(32) は '20' を返します。
- altova:integer-to-hex-string(33) は '21' を返します。
- altova:integer-to-hex-string(90) は '5A' を返します。

[トップ]

数値をフォーマットする関数

generate-auto-number [altova:]

altova:generate-auto-number(ID *as xs:string*, StartsWith *as xs:double*, Increment *as xs:double*, ResetOnChange *as xs:string*) を xs:integer とする XP1 XP2 XQ1 XP3 XQ3
関数が呼び出される度に番号を生成します。関数が初めて呼び出される際に生成される最初の番号は StartsWith 引数により指定されます。その後の関数の呼び出しは新しい番号を生成します。この番号は前に生成された番号を、Increment 引数で指定された値ごとにインクリメントします。実質的には、altova:generate-auto-number 関数は、ID 引数により指定されたカウンターを作成し、このカウンターが関数が呼び出されるごとにインクリメントされます。ResetOnChange 引数の値が、前の関数呼び出しから変更されると、生成される番号の値が StartsWith 値にリセットされます。自動付番は altova:reset-auto-number 関数を使用してリセットすることができます。

サンプル

- altova:generate-auto-number("ChapterNumber", 1, 1, "SomeString") は、関数が呼び出される度に番号を 1 つ返します。starting with 1, から始まり、関数が呼び出される度に 1 ずつインクリメントします。その後の呼び出しの 4 番目の"SomeString"引数がある限り、インクリメントは継続されます。4 番目の引数の値が変更されると、(ChapterNumber と呼ばれる)カウンターは 1 にリセットされます。ChapterNumber の値も altova:reset-auto-number 関数の呼び出しにより altova:reset-auto-number("ChapterNumber") ようにリセットされます。

[トップ]

2.1.6 XPath/ XQuery 関数 :シーケンス

Altova のシーケンス拡張関数は XPath とXQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の**XPath 3.0** および**XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、** `http://www.altova.com/xslt-extensions` に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3

attributes [altova:]

altova:attributes(AttributeName *as xs:string***)** を**attribute()*** とする XP3 XQ3
入力引数 AttributeName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、attribute:: 軸に対して行われます。これは、コンテキストノードが親要素ノードである必要があることを意味します。

サンプル

- **altova:attributes**("MyAttribute") は MyAttribute()* を返します。

altova:attributes(AttributeName *as xs:string*, **SearchOptions** *as xs:string***)** as **attribute()*** XP3 XQ3
入力引数 AttributeName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、attribute:: 軸に対して行われます。コンテキストノードが親要素ノードである必要があります。第 2引数はオプションのフラグを含みます。使用することのできるフラグは以下の通りです :
r = 正規表現検索を切り替えます ;AttributeName は正規表現検索文字列である必要があります ;
f = このオプションが指定されている場合、AttributeName は完全一致を提供します。それ以外の場合は、AttributeName は属性名に部分的に一致するとその属性を返します。例えば : **f** が指定されていない場合、MyAtt は MyAttribute を返します。
i = 大文字と小文字を一致させる検索に切り替えます。
p = 検索に名前空間プレフィックスを含みます。AttributeName は、名前空間プレフィックスを含みます。例えば : altova:MyAttribute。
フラグは順序に関わりなく書き込むことができます。無効なフラグはエラーを生成します。1つまたは複数のフラグを省略することができます。空の文字列は許可されていますが、引数を1つしか持たない関数と同じ効果を持ちます (*前の署名*)、ですが、空のシーケンスは第 2の引数としては許可されていません。

サンプル

- **altova:attributes**("MyAttribute", "rfip") は MyAttribute()* を返します。
- **altova:attributes**("MyAttribute", "pri") は MyAttribute()* を返します。
- **altova:attributes**("MyAtt", "rip") は MyAttribute()* を返します。
- **altova:attributes**("MyAttributes", "rfip") は不一致を返します。
- **altova:attributes**("MyAttribute", "") は MyAttribute()* を返します。
- **altova:attributes**("MyAttribute", "Rip") 認識されないフラグエラーを返します。

- **altova:attributes**("MyAttribute",) 見つからない第 2引数を返します。

elements [altova:]

altova:elements(ElementName *as xs:string***)** を**element()*** とする XP3 XQ3
入力引数 ElementName で与えられた名前と同じローカル名を持つすべての要素を返します。検索は大文字と小文字を区別して child:: 軸に対して実行されます。コンテキストノードは、検索される要素の親ノードである必要があります。

サンプル

- **altova:elements**("MyElement") は MyElement()* を返します。

altova:elements(ElementName *as xs:string*, **SearchOptions** *as xs:string***)** as **element()*** XP3 XQ3
入力引数 ElementName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、child:: 軸に対して行われます。コンテキストノードが親要素ノードである必要があります。第 2 引数はオプションのフラグを含みます。使用することのできるフラグは以下の通りです :
r = 正規表現検索を切り替えます ;ElementName は正規表現検索文字列である必要があります ;
f = このオプションが指定されている場合、ElementName は完全一致を提供します。それ以外の場合は、ElementName は属性名に部分的に一致するとその属性を返します。例えば : **f** が指定されていない場合、MyElem は MyElement を返します。
i = 大文字と小文字を一致させる検索に切り替えます。
p = 検索に名前空間プレフィックスを含みます。ElementName は、名前空間プレフィックスを含みます。例えば : altova:MyElement。
フラグは順序に関わりなく書き込むことができます。無効なフラグはエラーを生成します。1つまたは複数のフラグを省略することができます。空の文字列は許可されていますが、引数を1つしか持たない関数と同じ効果を持ちます (前の署名)、ですが、空のシーケンスは第 2の引数としては許可されていません。

サンプル

- **altova:elements**("MyElement", "rip") は MyElement()* を返します。
- **altova:elements**("MyElement", "pri") は MyElement()* を返します。
- **altova:elements**("MyElement", "") は MyElement()* を返します。
- **altova:attributes**("MyElem", "rip") は MyElement()* を返します。
- **altova:attributes**("MyElements", "rfip") 不一致を課wします。
- **altova:elements**("MyElement", "Rip") 認識されないフラグエラーを返します。
- **altova:elements**("MyElement",) 見つからない第 2引数を返します。

find-first [altova:]

altova:find-first((Sequence *as item()**), **(Condition(Sequence-Item** *as xs:boolean***))** を**item()?** とする XP3 XQ3
この関数は 2つの引数を必要とします。最初の引数は 1つ または 1つ以上のデータ型のアイテムのシーケンスです。第 2の引数Condition は、(1 のアリティを持つ) 1つの引数 を必要とし、boolean を返す XPath 関数に対する参照です。Condition で参照された関数の代わりに、**Sequence** の各アイテムが提出されます。(注意 : この関数は 1つの引数のみを必要とします。) **Condition** 内の関数に **true()** と評価させる最初の **Sequence** アイテムは、**altova:find-first**、反復の終了の結果として返されます。

サンプル

- **altova:find-first**(5 to 10, function($a) {$a mod 2 = 0}) は xs:integer 6 を返します。
Condition 引数は、**$a** という名のインライン関数を宣言し、定義します。XPath 3.0 インライン関数 **function()** を参照します。**altova: Sequence** 引数内の各アイテムでは、**find-first** がパスさ

れ、代わりに、**$a** を入力値とします。入力値は関数定義 (`$a mod 2 = 0`)内の条件に対してテストされます。この条件を満たす最初の入力値が**altova:find-first** (この場合は6) の結果として返されます。

- **altova:find-first**((1 to 10), (function($a) {$a+3=7})) は`xs:integer 4` を返します。

更なるサンプル
ファイル**C:\Temp\Customers.xml** が存在する場合 :

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は`xs:string C:\Temp\Customers.xml` を返します。

ファイル**C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** が存在する場合 :

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は`xs:string http://www.altova.com/index.html` を返します。

ファイル**C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** も存在しない場合 :

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) 結果無しを返します。

上のサンプルについての注意点

- XPath 3.0 関数 `doc-available` はURI として使用され、ドキュメントノードが提出されたURI で検出される場合 `true` を返す単一の引数を必要とします。 (ですから、提出されたURI でのドキュメントはXML ドキュメントである必要があります。)
- `doc-available` 関数は、`altova:find-first` の第 2引数である、**Condition** で使用することができます。これは、 1つの引数 (アリティ=1)のみを必要とするからであり、`item()` を入力 (URI として使用される文字列) として、boolean の値を返すからです。
- `doc-available` 関数は、参照されているだけで、呼び出されていない点に注意してください。アタッチされている`#1` サフィックスは関数が 1つのアリティであることを表示するためです。`doc-available#1` の意味は以下のとおりです :アリティ=1 を持つ doc-availabe() 関数を使用し、最初のシーケンスの各アイテムの代わりに単一引数としてパスします。 この結果、 2つの文字列の各自つは、文字列をURI として使用し、URI にドキュメントノードが存在するかテストする**doc-available()** にパスされます。1つが各当する場合、**doc-available()** 関数は`true()` を評価し、シーケンス内のその文字列のインデックス ポジションは、 **altova:find-first** 関数の結果として返されます。doc-available() 関数に関しての注意点 :相対パスは、デフォルトで関数がロードされるXML ドキュメントの現在のベースURI に対して相対的に解決されます。

find-first-combination [altova:]

altova:find-first-combination((Seq-01 *as item()**)**, **(Seq-02** *as item()**)**, **(Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean***))** を**item()*** とする **XP3** **XQ3**
この関数は 3つの引数を必要とします :

- 最初の 2つの引数 **Seq-01** と**Seq-02**, は、1つまたは1つ以上のデータ型のアイテムです。
- 第 3の引数 **Condition** は、(2 のアリティを持つ) 2つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 と**Seq-02** のアイテムは指定された組み合わせ (各シーケンスからの1つずつにアイテムで構成されるペア)

で、Condition 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X1 Y2), (X1 Y3) ... (X1 Yn), (X2 Y1), (X2 Y2) ... (Xn Yn)
```

Condition 関数に true() と評価するように指示する最初のペアは altova:find-first-combination の結果として返されます。以下の点に注意してください：(i) Condition 関数が提出された引数ペア内で繰り返され、true() を評価しない場合、altova:find-first-combination は、結果を返しません；(ii) altova:find-first-combination の結果が常に (データ型の) アイテムのペアである場合またはアイテムでない場合。

サンプル

- altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a +$b = 32}) は xs:integers (11, 21) のシーケンスを返します。
- altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a +$b = 33}) は xs:integers (11, 22) のシーケンを返します。
- altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a +$b = 34}) は xs:integers (11, 23) のシーケンを返します。

find-first-pair [altova:]

altova:find-first-pair((Seq-01 *as item()**), (Seq-02 *as item()**), (Condition(Seq-01-Item, Seq-02-Item *as xs:boolean*)) を item()* とする XP3 XQ3

この関数は 3 つの引数を必要とします：

- 最初の 2 つの引数 Seq-01 と Seq-02, は、1 つまたは 1 つ以上のデータ型のアイテムです。
- 第 3 の引数 Condition は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 と Seq-02 のアイテムは指定された組み合わせで、Condition 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Condition 関数に true() と評価するように指示する最初のペアは altova:find-first-pair の結果として返されます。以下の点に注意してください：(i) Condition 関数が提出された引数ペア内で繰り返され、true() を評価しない場合、altova:find-first-pair は、結果を返しません；(ii) altova:find-first-combination の結果が常に (データ型の) アイテムのペアである場合またはアイテムでない場合。

サンプル

- altova:find-first-pair(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) は xs:integers (11, 21) のシーケンスを返します。
- altova:find-first-pair(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) は *結果無し* を返します。

上の 2 つのサンプルに表示される通り、ペアの順序は以下の通りです：(11, 21) (12, 22) (13, 23)...(20, 30)。(33 を返す指示されたペアがないため) この理由で第 2 のサンプルは *結果無し* を返します。

find-first-pair-pos [altova:]

altova:find-first-pair-pos((Seq-01 *as item()**), (Seq-02 *as item()**),

(Condition(Seq-01-Item, Seq-02-Item as xs:boolean)) を**xs:integer** とする **XP3 XQ3**

この関数は 3 つの引数を必要とします :

- 最初の 2 つの引数 **Seq-01** と**Seq-02**, は、1 つまたは1つ以上のデータ型のアイテムです。
- 第 3の引数 **Condition** は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 と**Seq-02** のアイテムは指定された組み合わせで、**Condition** 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Condition 関数に**true()** を評価させる、最初に指示されたペアのインデックスポジションは、は **altova:find-first-pair-pos** の結果として返されます。関数が提出された引数ペア内で繰り返され、**true()** を一度も評価しない場合、**altova:find-first-pair-pos** *結果無し*が返します。

サンプル

- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) は1 を返します
- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) は*結果無し*を返します。

上の 2 つのサンプルに表示される通り、ペアの順序は以下の通りです :(11, 21) (12, 22) (13, 23)...(20, 30)。最初のサンプルでは、最初のペアは**Condition** 関数に**true()** を評価させ、シーケンス内のインデックスポジションには**1** が返されます。第 2 のサンプルでは、33 を返すペアがないため、*結果無し*が返します。

find-first-pos [altova:]

altova:find-first-pos((Sequence as item()***), (Condition(Sequence-Item** as xs:boolean**))** を**xs:integer** とする **XP3 XQ3**

この関数は 2 つの引数を必要とします。最初の引数は 1 つ、または 1 つ以上のデータ型のアイテムのシーケンスです。第 2 の引数 Condition は、(1 のアリティを持つ) 1 つの引数 を必要とし、boolean を返す XPath 関数に対する参照です。Condition で参照された関数の代わりに、**Sequence** の各アイテムが提出されます。(*注意* : この関数は 1 つの引数のみを必要とします。)**Condition** 内の関数に**true()** と評価させる最初の **Sequence** アイテムは、**altova:find-first-pos** の結果として返された**Sequence** 内インデックスポジションを持ちます。

サンプル

- **altova:find-first-pos**(5 to 10, function($a) {$a mod 2 = 0}) は xs:integer 2 を返します。

Condition 引数は、**$a** という名のインライン関数を宣言し、定義します。XPath 3.0 インライン関数 **function()** を参照します。 **Sequence** 引数内の各アイテムでは、**find-first-pos** がパスされ、代わりに、**$a** を入力値とします。入力値は関数定義 ($a mod 2 = 0)内の条件に対してテストされます。この条件を満たす最初の入力値が**altova:find-first-pos** ((シーケンス内で条件を満たす最初の値である 6 がシーケンスのインデックス位置 2 にあるためこの場合は 2)の結果として返されます。

- **altova:find-first-pos**((2 to 10), (function($a) {$a+3=7})) はxs:integer 3 を返します。

更なるサンプル

ファイル C:\Temp\Customers.xml が存在する場合：

- altova:find-first-pos(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) returns 1

ファイル C:\Temp\Customers.xml が存在せず、http://www.altova.com/index.html が存在する場合：

- altova:find-first-pos(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) returns 2

ファイル C:\Temp\Customers.xml が存在せず、http://www.altova.com/index.html も存在しない場合：

- altova:find-first-pos(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) 結果無しを返します。

上のサンプルについての注意点

- XPath 3.0 関数 doc-available は URI として使用され、ドキュメントノードが提出された URI で検出される場合 true を返す単一の引数を必要とします。 (ですから、提出された URI でのドキュメントは XML ドキュメントである必要があります。)
- doc-available 関数は、altova:find-first-pos の第 2 引数である、Condition で使用することができます。これは、 1 つの引数 (アリティ=1) のみを必要とするからであり、item() を入力 (URI として使用される文字列) として、boolean の値を返すからです。
- doc-available 関数は、参照されているだけで、呼び出されていない点に注意してください。アタッチされている #1 サフィックスは関数が 1 つのアリティであることを表示するためです。doc-available#1 の意味は以下のとおりです：アリティ=1 を持つ doc-availabe() 関数を使用し、最初のシーケンスの各アイテムの代わりに単一引数としてパスします。 この結果、 2 つの文字列の各自づは、文字列を URI として使用し、URI にドキュメントノードが存在するかテストする doc-available() にパスされます。1 つが各当する場合、doc-available() 関数は true() を評価し、シーケンス内のその文字列のインデックス ポジションは、 altova:find-first-pos 関数の結果として返されます。doc-available() 関数に関しての注意点：相対パスは、デフォルトで関数がロードされる XML ドキュメントの現在のベース URI に対して相対的に解決されます。

substitute-empty [altova:]

altova:substitute-empty(FirstSequence as item()*, SecondSequence as item()) を item()* とする XP3 XQ3

FirstSequence が空の場合、SecondSequence を返します。FirstSequence が空でない場合、FirstSequence を返します。

サンプル

- altova:substitute-empty((1,2,3), (4,5,6)) は (1,2,3) を返します。
- altova:substitute-empty((), (4,5,6)) は (4,5,6) を返します。

2.1.7 XPath/ XQuery 関数 :文字列

Altova の文字列拡張関数は XPath と XQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。 このセクションの関数は、Altova の **XPath 3.0** および **XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、** `http://www.altova.com/xslt-extensions` に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意ください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3

camel-case [altova:]

`altova:camel-case(InputString as xs:string)` を`xs:string` とする XP3 XQ3

入力文字列を`InputString` をキャメルケースで返します。文字列は (空白スペースのショートカットである) 正規表現 `'\s'` を使用して分析されます。空白文字または連続する空白文字のシーケンスの後の最初の非空白スペース文字は大文字です。出力文字列の最初の文字は大文字です。

サンプル

- `altova:camel-case("max")` `Max` を返します。
- `altova:camel-case("max max")` `Max Max` を返します。
- `altova:camel-case("file01.xml")` `File01.xml` を返します。
- `altova:camel-case("file01.xml file02.xml")` `File01.xml File02.xml` を返します。
- `altova:camel-case("file01.xml file02.xml")` `File01.xml File02.xml` を返します。
- `altova:camel-case("file01.xml -file02.xml")` `File01.xml -file02.xml` を返します。

`altova:camel-case(InputString as xs:string, SplitChars as xs:string, IsRegex as xs:boolean)` を`xs:string` とする XP3 XQ3

`SplitChars` を使用して、次の大文字をトリガーする文字を決定し、入力文字列を`InputString` キャメルケースに変換します。`SplitChars` は、`IsRegex = true()` の場合、または、`IsRegex = false()` の場合、プレーン文字は正規表現として使用されます。出力文字列の最初の文字は大文字です。

サンプル

- `altova:camel-case("setname getname", "set|get", true())` `setName getName` を返します。
- `altova:camel-case("altova\documents\testcases", "\", false())` `Altova \Documents\Testcases` を返します。

char [altova:]

`altova:char(Position as xs:integer)` を`xs:string` とする XP3 XQ3

`xs:string.Position` に対してのコンテキストアイテムの値を変換することにより得られた文字列内の `Position` 引数により指定されたポジションにある文字を含む文字列を返します。引数により提出されたインデックスに文字が存在しない場合、結果文字列は空です。

サンプル

コンテキストアイテムが`1234ABCD` の場合：

- `altova:char(2)` は`2` を返します。

- altova:char(5) は A を返します。
- altova:char(9) は空の文字列を返します。
- altova:char(-2) は空の文字列を返します。

altova:char(InputString *as xs:string*, **Position** *as xs:integer*) を**xs:string** とする **XP3 XQ3**
引数として提出された文字列内の Position 引数により指定されたポジションでの文字を含む文字列を返します。Position 引数により提出されたインデックスに文字が存在しない場合、結果文字列は空です。

サンプル

- altova:char("2014-01-15", 5) は - を返します。
- altova:char("USA", 1) は U を返します。
- altova:char("USA", 10) は空の文字列を返します。
- altova:char("USA", -2) は空の文字列を返します。

first-chars [altova:]

altova:first-chars(X-Number *as xs:integer*) を**xs:string** とする **XP3 XQ3**
xs:string に対するコンテキストアイテムの値を変換することにより得られた最初の X-Number 文字を含む文字列を返します。

サンプル

コンテキストアイテムが **1234ABCD** の場合：

- altova:first-chars(2) は 12 を返します。
- altova:first-chars(5) は 1234A を返します。
- altova:first-chars(9) は 1234ABCD を返します。

altova:first-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) を **xs:string** とする **XP3 XQ3**
InputString 引数として提出された文字列の最初の文字を含む文字列を返します。

サンプル

- altova:first-chars("2014-01-15", 5) は 2014- を返します。
- altova:first-chars("USA", 1) は U を返します。

last-chars [altova:]

altova:last-chars(X-Number *as xs:integer*) を**xs:string** とする **XP3 XQ3**
xs:string に対してのコンテキストアイテムの値の変換より取得された文字列の最後の X-Number 文字を含んでいる文字列を返します。

サンプル

コンテキストアイテムが **1234ABCD** の場合：

- altova:last-chars(2) は CD を返します。
- altova:last-chars(5) は 4ABCD を返します。
- altova:last-chars(9) は 1234ABCD を返します。

altova:last-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) as **xs:string** とする **XP3 XQ3**
引数として提出された文字列の最後の X-Number 文字を含んでいる文字列を返します。

サンプル

- `altova:last-chars("2014-01-15", 5)` は `01-15` を返します。
- `altova:last-chars("USA", 10)` は `USA` を返します。

pad-string-left [altova:]

`altova:pad-string-left(StringToPad as xs:string, StringLength as xs:integer, PadCharacter as xs:string)` を `xs:string` とする XP3 XQ3

`PadCharacter` 引数は1文字です。文字列の左側にパッドされ、この数が、`StringLength` 引数の整数の値と等しくなるように `StringToPad` の文字の数を増やします。`StringLength` 引数は任意の整数の値（正数または負数）を持つことができますが、パディングは、`StringLength` の値が `StringToPad` 内の文字数よりも多い場合の場合のみ発生します。もし、`StringToPad` が `StringLength` の値よりも多くの文字数を持つ場合、`StringToPad` は変更されません。

サンプル

- `altova:pad-string-left('AP', 1, 'Z')` は `'AP'` を返します。
- `altova:pad-string-left('AP', 2, 'Z')` は `'AP'` を返します。
- `altova:pad-string-left('AP', 3, 'Z')` は `'ZAP'` を返します。
- `altova:pad-string-left('AP', 4, 'Z')` は `'ZZAP'` を返します。
- `altova:pad-string-left('AP', -3, 'Z')` は `'AP'` を返します。
- `altova:pad-string-left('AP', 3, 'YZ')` は[パッド文字が長すぎます] エラーを返します。

pad-string-right [altova:]

`altova:pad-string-right(StringToPad as xs:string, StringLength as xs:integer, PadCharacter as xs:string)` を `xs:string` とする XP3 XQ3

`PadCharacter` 引数は1文字です。文字列の右側にパッドされ、この数が、`StringLength` 引数の整数の値と等しくなるように `StringToPad` の文字の数を増やします。`StringLength` 引数は任意の整数の値（正数または負数）を持つことができますが、パディングは、`StringLength` の値が `StringToPad` 内の文字数よりも多い場合の場合のみ発生します。もし、`StringToPad` が `StringLength` の値よりも多くの文字数を持つ場合、`StringToPad` は変更されません。

サンプル

- `altova:pad-string-right('AP', 1, 'Z')` を `'AP'` を返します。
- `altova:pad-string-right('AP', 2, 'Z')` を `'AP'` を返します。
- `altova:pad-string-right('AP', 3, 'Z')` を `'APZ'` を返します。
- `altova:pad-string-right('AP', 4, 'Z')` を `'APZZ'` を返します。
- `altova:pad-string-right('AP', -3, 'Z')` を `'AP'` を返します。
- `altova:pad-string-right('AP', 3, 'YZ')` は[パッド文字が長すぎます] エラーを返します。

repeat-string [altova:]

`altova:repeat-string(InputString as xs:string, Repeats as xs:integer)` を `xs:string` とする XP2 XQ1 XP3 XQ3

最初の `InputString` 引数により構成される文字列、`Repeats` 回繰り返しを生成します。

サンプル

- `altova:repeat-string("Altova #", 3)` は `"Altova #Altova #Altova #"` を返します。

substring-after-last [altova:]

altova:substring-after-last(MainString *as xs:string*, **CheckString** *as xs:string*) を**xs:string** とする XP3 XQ3
CheckString が MainString 内で検出された場合、MainString 内の CheckString が発生した後のサブ文字列が返されます。MainString 内で CheckString が検出されない場合、空の文字列が返されます。CheckString が空の文字列の場合、MainString 全体が返されます。一度以上発生する場合、CheckString の最後の発生の後のサブ文字列が返されます。

サンプル

- **altova:substring-after-last**('ABCDEFGH', 'B') は 'CDEFGH' を返します。
- **altova:substring-after-last**('ABCDEFGH', 'BC') は 'DEFGH' を返します。
- **altova:substring-after-last**('ABCDEFGH', 'BD') は '' を返します。
- **altova:substring-after-last**('ABCDEFGH', 'Z') は '' を返します。
- **altova:substring-after-last**('ABCDEFGH', '') は 'ABCDEFGH' を返します。
- **altova:substring-after-last**('ABCD-ABCD', 'B') は 'CD' を返します。
- **altova:substring-after-last**('ABCD-ABCD-ABCD', 'BCD') は '' を返します。

substring-before-last [altova:]

altova:substring-before-last(MainString *as xs:string*, **CheckString** *as xs:string*) を**xs:string** とする XP3 XQ3
CheckString が MainString 内で検出された場合、MainString 内の CheckString が発生する前のサブ文字列が返されます。MainString 内で CheckString が一度以上発生する場合、CheckString の最後の発生の前のサブ文字列が返されます。

サンプル

- **altova:substring-before-last**('ABCDEFGH', 'B') は 'A' を返します。
- **altova:substring-before-last**('ABCDEFGH', 'BC') は 'A' を返します。
- **altova:substring-before-last**('ABCDEFGH', 'BD') は '' を返します。
- **altova:substring-before-last**('ABCDEFGH', 'Z') は '' を返します。
- **altova:substring-before-last**('ABCDEFGH', '') は '' を返します。
- **altova:substring-before-last**('ABCD-ABCD', 'B') は 'ABCD-A' を返します。
- **altova:substring-before-last**('ABCD-ABCD-ABCD', 'ABCD') は 'ABCD-ABCD-' を返します。

substring-pos [altova:]

altova:substring-pos(StringToCheck *as xs:string*, **StringToFind** *as xs:string*) を**xs:integer** とする XP3 XQ3
StringToCheck 内での StringToFind の最初の発生の文字位置を整数として返します。StringToCheck の最初の文字は、位置 1 にあります。StringToFind が StringToCheck 内で発生しない場合、整数 0 が返されます。第 2 またはその後の StringToCheck, 発生を確認するには、この関数の次の署名を確認してください。

サンプル

- **altova:substring-pos**('Altova', 'to') 3 を返します。
- **altova:substring-pos**('Altova', 'tov') 3 を返します。
- **altova:substring-pos**('Altova', 'tv') returns 0 を返します。
- **altova:substring-pos**('AltovaAltova', 'to') 3 を返します。

altova:substring-pos(StringToCheck *as xs:string*, **StringToFind** *as xs:string*, **Integer** *as xs:integer*) を**xs:integer** とする XP3 XQ3
StringToCheck 内での StringToFind の最初の発生の文字位置を返します。Integer 引数により与えら

れた、文字位置から StringToFind の検索が開始されます。この位置の前の文字サブ文字列は検索されません。しかし、返された整数は、全体文字列 StringToCheck の検索された文字列の位置です。この署名は、StringToCheck 内で複数回発生する、第 2 または後の発生を検索する際に役に立ちます。StringToFind が StringToCheck 内で発生しない場合、整数 0 が返されます。

サンプル

- **altova:substring-pos**('Altova', 'to', 1) 3 を返します。
- **altova:substring-pos**('Altova', 'to', 3) 3 を返します。
- **altova:substring-pos**('Altova', 'to', 4) 0 を返します。
- **altova:substring-pos**('Altova-Altova', 'to', 0) 3 を返します。
- **altova:substring-pos**('Altova-Altova', 'to', 4) 10 を返します。

trim-string [altova:]

altova:trim-string(InputString *as xs:string*) を**xs:string** とする **XP3 XQ3**

この関数は xs:string 引数を必要とし、先頭または後続の空白を削除し、トリミングされた xs:string を返します。

サンプル

- **altova:trim-string**(" Hello World ")) は "Hello World" を返します。
- **altova:trim-string**("Hello World ")) は "Hello World" を返します。
- **altova:trim-string**(" Hello World")) は "Hello World" を返します。
- **altova:trim-string**("Hello World")) は "Hello World" を返します。
- **altova:trim-string**("Hello World")) は "Hello World" を返します。

trim-string-left [altova:]

altova:trim-string-left(InputString *as xs:string*) を**xs:string** とする **XP3 XQ3**

この関数は xs:string 引数を必要とし、先頭または後続の空白を削除し、トリミングされた xs:string を返します。

サンプル

- **altova:trim-string-left**(" Hello World ")) は "Hello World " を返します。
- **altova:trim-string-left**("Hello World ")) は "Hello World " を返します。
- **altova:trim-string-left**(" Hello World")) は "Hello World" を返します。
- **altova:trim-string-left**("Hello World")) は "Hello World" を返します。
- **altova:trim-string-left**("Hello World")) は "Hello World" を返します。

trim-string-right [altova:]

altova:trim-string-right(InputString *as xs:string*) を**xs:string** とする **XP3 XQ3**

この関数は xs:string 引数を必要とし、先頭または後続の空白を削除し、トリミングされた xs:string を返します。

サンプル

- **altova:trim-string-right**(" Hello World ")) は " Hello World" を返します。
- **altova:trim-string-right**("Hello World ")) は "Hello World" を返します。
- **altova:trim-string-right**(" Hello World")) は " Hello World" を返します。
- **altova:trim-string-right**("Hello World")) は "Hello World" を返します。
- **altova:trim-string-right**("Hello World")) は "Hello World" を返します。

2.1.8 XPath/ XQuery 関数 :その他

以下の一般使用のためのXPath/XQuery 拡張関数は、XMLSpy の現在のバージョンによりサポートされています。また、次で使用することができます : (i) XSLT コンテキスト内のXPath 式、または (ii) XQuery ドキュメント内のXQuery 式。

関数の名前指定と言語の適用性

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、** `http://www.altova.com/xslt-extensions` に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3

URI 関数

▼ **get-temp-folder [altova:]**

`altova:get-temp-folder()` を`xs:string` とする XP2 XQ1 XP3 XQ3
この関数は引数を必要としません。この関数は現在のユーザーの一時的なフォルダーへのパスを返します。

⊟ *サンプル*

- `altova:get-temp-folder()` は、マシン上で `xs:string` として `C:\Users\<UserName>\AppData\Local\Temp\` と類似したパスを返します。

[Top]

2.1.9 チャート関数

以下に示されるチャート関数により、チャートをイメージとして作成、生成、保存することができます。これらの機能は、お使いのAltova 製品で、以下に記される方法によりサポートされます。しかしながら、将来使用されるバージョンの製品において、以下にある関数のサポートが打ち切られたり、個々の振る舞いが変化する可能性があることに留意してください。将来のリリースにおけるAltova 拡張関数のサポートについては、そのバージョンのドキュメンテーションを参照ください。

(XSLT 関数ではない)チャートに関する**XPath 関数**は、2つのグループに分けられています : :

- チャートの生成と保存を行う関数
- チャートの作成を行う関数

メモ : チャート関数は、**Enterprise** および**サーバー エディション**のAltova 製品でしかサポートされていないことに注意してください。

メモ： サーバーエディションのチャートでサポートされるイメージのフォーマットは jpg、png、および bmp です。無損失圧縮のため、推奨されるオプションは png です。Enterprise エディションでサポートされるフォーマットは jpg. png、bmp、および gif です。

チャートの生成と保存を行う関数

これらの関数は、チャート作成関数により得られた) チャートオブジェクトを受け取り、イメージの生成を行うか、イメージをファイルへ保存します。

altova:generate-chart-image ($chart, $width, $height, $encoding) as atomic

以下の説明を参照ください

- $chart は altova:create-chart 関数により得られたチャート拡張アイテムです。
- $width ならびに $height により大きさを指定する必要があります。
- $encoding は binarytobase64 または binarytobase16 から選択することができます。

関数からは、指定されたエンコーディングならびにイメージフォーマットにてチャートのイメージが返されます。

---

altova:generate-chart-image ($chart, $width, $height, $encoding, $imagetype) as atomic

以下の説明を参照ください

- $chart は altova:create-chart 関数により得られたチャート拡張アイテムです。
- $width ならびに $height により大きさを指定する必要があります。
- $encoding は base64Binary または hexBinary であることができます。
- $imagetype は次のフォーマットから選択することができます :png、gif、bmp、jpg、jpeg。gif はサーバー製品でサポートされていないことにご注意してください。*ページの上のメモを参照*してください。

関数からは、指定されたエンコーディングならびにイメージフォーマットにてチャートのイメージが返されます。

---

altova:save-chart-image ($chart, $filename, $width, $height) as empty()
(Windows のみ)

以下の説明を参照ください

- $chart は altova:create-chart 関数により得られたチャート拡張アイテムです。
- $filename は保存されるチャートイメージのファイルパスならびにファイル名です。
- $width ならびに $height により大きさを指定する必要があります。

関数により $filename で指定されたファイルにチャートイメージが保存されます。

---

altova:save-chart-image ($chart, $filename, $width, $height, $imagetype) as empty() ***(Windows のみ)***

以下の説明を参照ください

- $chart は altova:create-chart 関数により得られたチャート拡張アイテムです。
- $filename は保存されるチャートイメージのファイルパスならびにファイル名です。
- $width ならびに $height により大きさを指定する必要があります。
- $imagetype は次のフォーマットから選択することができます :png、gif、bmp、jpg、jpeg。gif はサーバー製品でサポートされていないことに注意してください。*ページの上のメモを参照してください。*

指定されたイメージフォーマットで $filename により指定されたファイルにチャートイメージが保存されます。

チャートの作成を行う関数

以下の関数によりチャートの作成を行うことができます

altova:create-chart($chart-config, $chart-data-series*) as chart extension item

以下の説明を参照ください

- $chart-config は altova:create-chart-config 関数または altova:create-chart-config-from-xml 関数により取得された chart-config 拡張アイテムとなります。
- $chart-data-series は altova:create-chart-data-series 関数または altova:create-chart-data-series-from-rows 関数により取得された chart-data-series 拡張アイテムとなります。

関数により、引数により与えられたデータから作成されたチャート拡張アイテムが返されます。

---

altova:create-chart-config($type-name, $title) as chart-config extension item

以下の説明を参照ください

- $type-name により、以下から作成される種類のチャートが選択されます :Pie、Pie3d、 BarChart、BarChart3d、BarChart3dGrouped、 LineChart、ValueLineChart、RoundGauge、BarGauge
- $title によりチャートの名前が指定されます。

関数により、チャートの構成情報が含まれる chart-config 拡張アイテムが返されます。

---

altova:create-chart-config-from-xml($xml-struct) as chart-config extension item

以下の説明を参照ください

- $xml-struct はチャートの構成情報が含まれる XML 構造です。

関数により、チャートの構成情報が含まれるchart-config 拡張アイテムが返されます。この情報はXML データフラグメントから与えられます。

---

altova:create-chart-data-series($series-name?, $x-values*, $y-values*) as chart-data-series extension item

以下の説明を参照ください

- $series-name により、系列の名前が指定されます。
- $x-values により、X-軸値のリストが与えられます。
- $y-values により、Y-軸値のリストが与えられます。

関数により、チャートの作成に必要なデータの情報（系列の名前と軸のデータ）が含まれるchart-data-series 拡張アイテムが返されます。

---

altova:create-chart-data-row(x, y1, y2, y3, ...) as chart-data-x-*N*y-row extension item

以下の説明を参照ください

- x はチャートデータ行のX-軸カラムにおける値です。
- yN はY-軸カラムの値です。

関数により、1つの系列のX-軸カラムならびにY-軸カラムに対するデータが含まれるchart-data-x-*N*y-row 拡張アイテムが返されます。

---

altova:create-chart-data-series-from-rows($series-names as xs:string*, $row*) as chart-data-series extension item

以下の説明を参照ください

- $series-name は作成される系列の名前です。
- $row は系列として作成されるchart-data-x-*N*y-row 拡張アイテムです。

関数により、系列のX-軸ならびにY-軸に関するデータが含まれるchart-data-series 拡張アイテムが返されます。

---

altova:create-chart-layer($chart-config, $chart-data-series*) as chart-layer extension item

以下の説明を参照ください

- $chart-config は altova:create-chart-config 関数または altova:create-chart-config-from-xml 関数により取得された $chart-config 拡張アイテムです。
- $chart-data-series は altova:create-chart-data-series 関数または altova:create-chart-data-series-from-rows 関数により取得された chart-data-series 拡張アイテムです。

関数により、チャートのレイヤーデータが含まれる **chart-layer** 拡張アイテムが返されます。

---

altova:create-multi-layer-chart($chart-config, $chart-data-series*, $chart-layer*)

以下の説明を参照ください

- $chart-config は altova:create-chart-config 関数または altova:create-chart-config-from-xml 関数により取得された chart-config 拡張アイテムです。
- $chart-data-series は altova:create-chart-data-series 関数または altova:create-chart-data-series-from-rows 関数により取得された chart-data-series 拡張アイテムです。
- $chart-layer は altova:create-chart-layer 関数により取得された **chart-layer** 拡張アイテムです。

関数により、複数レイヤーのチャートアイテムが返されます。

---

altova:create-multi-layer-chart($chart-config, $chart-data-series*, $chart-layer*, xs:boolean $mergecategoryvalues)

以下の説明を参照ください

- $chart-config は altova:create-chart-config 関数または altova:create-chart-config-from-xml 関数により取得された chart-config 拡張アイテムです。
- $chart-data-series は altova:create-chart-data-series 関数または altova:create-chart-data-series-from-rows 関数により取得された chart-data-series 拡張アイテムです。
- $chart-layer は altova:create-chart-layer 関数により取得された **chart-layer** 拡張アイテムです。

関数により、複数レイヤーのチャートアイテムが返されます。

チャートデータの XML 構造

以下にチャートデータの XML 構造、ならびにそれが Altova のチャート拡張関数においてどのように表示されるかを示します。チャートの種類によっては、ここでの記述内容により外観が変化します。全ての要素が全種類のチャートに対して使用されるわけではなく、例えば <Pie> 要素は棒グラフに対して無視されます。

メモ: チャート関数は、**Enterprise** ならびに **サーバー エディション** の Altova 製品でしかサポートされていないことにご注意してください。

```
<chart-config>
      <General
      SettingsVersion="1"  must be provided
      ChartKind="BarChart"  Pie, Pie3d, BarChart, StackedBarChart, BarChart3d,
BarChart3dGrouped, LineChart, ValueLineChart, AreaChart, StackedAreaChart, RoundGauge,
BarGauge, CandleStick
      BKColor="#ffffff"  Color
      BKColorGradientEnd="#ffffff"  Color. In case of a gradient, BKColor and
BKColorGradientEnd define the gradient's colors
      BKMode="#ffffff"  Solid, HorzGradient, VertGradient
      BKFile="Path+Filename"  String. If file exists, its content is drawn over the background.
      BKFileMode="Stretch"  Stretch, ZoomToFit, Center, Tile
      ShowBorder="1"  Bool
      PlotBorderColor="#000000"  Color
      PlotBKColor="#ffffff"  Color
      Title=""  String
      ShowLegend="1"  Bool
      OutsideMargin="3.%"  PercentOrPixel
      TitleToPlotMargin="3.%"  PercentOrPixel
      LegendToPlotMargin="3.%"  PercentOrPixel
      Orientation="vert"  Enumeration: possible values are: vert, horz
      >
      <TitleFont
         Color="#000000"  Color
         Name="Tahoma"  String
         Bold="1"  Bool
         Italic="0"  Bool
         Underline="0"  Bool
         MinFontHeight="10.pt"  FontSize (only pt values)
         Size="8.%"  FontSize />
      <LegendFont
         Color="#000000"
         Name="Tahoma"
         Bold="0"
         Italic="0"
         Underline="0"
         MinFontHeight="10.pt"
         Size="3.5%" />
      <AxisLabelFont
         Color="#000000"
         Name="Tahoma"
         Bold="1"
         Italic="0"
         Underline="0"
         MinFontHeight="10.pt"
         Size="5.%" />
       </General>

       <Line
             ConnectionShapeSize="1.%"  PercentOrPixel
             DrawFilledConnectionShapes="1"  Bool
             DrawOutlineConnectionShapes="0"  Bool
             DrawSlashConnectionShapes="0"  Bool
             DrawBackslashConnectionShapes="0"  Bool
       />
```

```
<Bar
        ShowShadow="1"   Bool
        ShadowColor="#a0a0a0"   Color
        OutlineColor="#000000"   Color
        ShowOutline="1"   Bool
/>

<Area
        Transparency="0"   UINT ( 0-255 ) 255 is fully transparent, 0 is opaque
        OutlineColor="#000000"   Color
        ShowOutline="1"   Bool
/>

<CandleStick
        FillHighClose="0"   Bool. If 0, the body is left empty. If 1,
FillColorHighClose is used for the candle body
        FillColorHighClose="#ffffff"   Color. For the candle body when close >
open
        FillHighOpenWithSeriesColor="1"   Bool. If true, the series color is used
to fill the candlebody when open > close
        FillColorHighOpen="#000000"   Color. For the candle body when open >
close and FillHighOpenWithSeriesColor is false
/>

<Colors   User-defined color scheme: By default this element is empty except for the
style and has no Color attributes
        UseSubsequentColors ="1"  Boolean. If 0, then color in overlay is used. If 1,
then subsequent colors from previous chart layer is used
        Style="User"   Possible values are: "Default", "Grayscale", "Colorful",
"Pastel", "User"
        Colors="#52aca0"   Color: only added for user defined color set
        Colors1="#d3c15d"   Color: only added for user defined color set
        Colors2="#8971d8"   Color: only added for user defined color set
        ...
        ColorsN=""   Up to ten colors are allowed in a set: from Colors to Colors9
</Colors>

<Pie
        ShowLabels="1"   Bool
        OutlineColor="#404040"   Color
        ShowOutline="1"   Bool
        StartAngle="0."   Double
        Clockwise="1"   Bool
        Draw2dHighlights="1"   Bool
        Transparency="0"   Int (0 to 255: 0 is opaque, 255 is fully transparent)
        DropShadowColor="#c0c0c0"   Color
        DropShadowSize="5.%"   PercentOrPixel
        PieHeight="10.%"   PercentOrPixel. Pixel values might be different in the
result because of 3d tilting
        Tilt="40.0"   Double (10 to 90: The 3d tilt in degrees of a 3d pie)
        ShowDropShadow="1"   Bool
        ChartToLabelMargin="10.%"   PercentOrPixel
        AddValueToLabel="0"   Bool
        AddPercentToLabel="0"   Bool
        AddPercentToLabels_DecimalDigits="0"  UINT ( 0 – 2 )
```

```
                >
                <LabelFont
                        Color="#000000"
                        Name="Arial"
                        Bold="0"
                        Italic="0"
                        Underline="0"
                        MinFontHeight="10.pt"
                        Size="4.%"    />
        </Pie>

        <XY>
                <XAxis  Axis
                        AutoRange="1"  Bool
                        AutoRangeIncludesZero="1"  Bool
                        RangeFrom="0."  Double: manual range
                        RangeTill="1."  Double : manual range
                        LabelToAxisMargin="3.%"  PercentOrPixel
                        AxisLabel=""  String
                        AxisColor="#000000"  Color
                        AxisGridColor="#e6e6e6"  Color
                        ShowGrid="1"  Bool
                        UseAutoTick="1"  Bool
                        ManualTickInterval="1."  Double
                        AxisToChartMargin="0.px"  PercentOrPixel
                        TickSize="3.px"  PercentOrPixel
                        ShowTicks="1"  Bool
                        ShowValues="1"  Bool
                        AxisPosition="LeftOrBottom"  Enums: "LeftOrBottom",
"RightOrTop", "AtValue"
                        AxisPositionAtValue = "0"  Double
                        >
                        <ValueFont
                                Color="#000000"
                                Name="Tahoma"
                                Bold="0"
                                Italic="0"
                                Underline="0"
                                MinFontHeight="10.pt"
                                Size="3.%"    />
                </XAxis>
                <YAxis  Axis (same as for XAxis)
                        AutoRange="1"
                        AutoRangeIncludesZero="1"
                        RangeFrom="0."
                        RangeTill="1."
                        LabelToAxisMargin="3.%"
                        AxisLabel=""
                        AxisColor="#000000"
                        AxisGridColor="#e6e6e6"
                        ShowGrid="1"
                        UseAutoTick="1"
                        ManualTickInterval="1."
                        AxisToChartMargin="0.px"
                        TickSize="3.px"
                        ShowTicks="1"  Bool
                        ShowValues="1"  Bool
                        AxisPosition="LeftOrBottom"  Enums: "LeftOrBottom",
```

"RightOrTop", "AtValue"

```
                AxisPositionAtValue = "0"   Double
                >
                <ValueFont
                        Color="#000000"
                        Name="Tahoma"
                        Bold="0"
                        Italic="0"
                        Underline="0"
                        MinFontHeight="10.pt"
                        Size="3.%"/>
        </YAxis>
</XY>

<XY3d
        AxisAutoSize="1"   Bool: If false, XSize and YSize define the aspect ration
of x and y axis. If true, aspect ratio is equal to chart window
        XSize="100.%"   PercentOrPixel. Pixel values might be different in the result
because of 3d tilting and zooming to fit chart
        YSize="100.%"   PercentOrPixel. Pixel values might be different in the result
because of 3d tilting and zooming to fit chart
        SeriesMargin="30.%"   PercentOrPixel. Pixel values might be different in the
result because of 3d tilting and zooming to fit chart
        Tilt="20."   Double. -90 to +90 degrees
        Rot="20."   Double. -359 to +359 degrees
        FoV="50.">   Double. Field of view: 1-120 degree
        >
        <ZAxis
                AutoRange="1"
                AutoRangeIncludesZero="1"
                RangeFrom="0."
                RangeTill="1."
                LabelToAxisMargin="3.%"
                AxisLabel=""
                AxisColor="#000000"
                AxisGridColor="#e6e6e6"
                ShowGrid="1"
                UseAutoTick="1"
                ManualTickInterval="1."
                AxisToChartMargin="0.px"
                TickSize="3.px" >
                <ValueFont
                        Color="#000000"
                        Name="Tahoma"
                        Bold="0"
                        Italic="0"
                        Underline="0"
                        MinFontHeight="10.pt"
                        Size="3.%"/>
        </ZAxis>
</XY3d>

<Gauge
        MinVal="0."   Double
        MaxVal="100."   Double
        MinAngle="225"   UINT: -359-359
        SweepAngle="270"   UINT: 1-359
        BorderToTick="1.%"   PercentOrPixel
        MajorTickWidth="3.px"   PercentOrPixel
```

```
                MajorTickLength="4.%"   PercentOrPixel
                MinorTickWidth="1.px"   PercentOrPixel
                MinorTickLength="3.%"   PercentOrPixel
                BorderColor="#a0a0a0"   Color
                FillColor="#303535"   Color
                MajorTickColor="#a0c0b0"   Color
                MinorTickColor="#a0c0b0"   Color
                BorderWidth="2.%"   PercentOrPixel
                NeedleBaseWidth="1.5%"   PercentOrPixel
                NeedleBaseRadius="5.%"   PercentOrPixel
                NeedleColor="#f00000"   Color
                NeedleBaseColor="#141414"   Color
                TickToTickValueMargin="5.%"   PercentOrPixel
                MajorTickStep="10."   Double
                MinorTickStep="5."   Double
                RoundGaugeBorderToColorRange="0.%"   PercentOrPixel
                RoundGaugeColorRangeWidth ="6.%"   PercentOrPixel
                BarGaugeRadius="5.%"   PercentOrPixel
                BarGaugeMaxHeight="20.%"   PercentOrPixel
                RoundGaugeNeedleLength="45.%"   PercentOrPixel
                BarGaugeNeedleLength="3.%"   PercentOrPixel
                >
                <TicksFont
                      Color="#a0c0b0"
                      Name="Tahoma"
                      Bold="0"
                      Italic="0"
                      Underline="0"
                      MinFontHeight="10.pt"
                      Size="4.%"
                />
                <ColorRanges>   User-defined color ranges. By default empty with no child
element entries
                      <Entry
                            From="50. "   Double
                            FillWithColor="1"   Bool
                            Color="#00ff00"   Color
                      />
                      <Entry
                            From="50.0"
                            FillWithColor="1"
                            Color="#ff0000"
                      />
                      ...
                </ColorRanges>
          </Gauge>
</chart-config>
```

チャート関数

以下のサンプルXSLT ドキュメントでは、Altova のチャート拡張関数の使用方法が示されます。更に下には、XML ドキュメントと、Altova XSLT 2.0 または 3.0 エンジンとXSLT ドキュメントによりXML ドキュメントが処理された結果生成される出力イメージのスクリーンショットが示されます。

メモ： チャート関数は、**Enterprise** ならびに**サーバー エディション**のAltova 製品でしかサポートされていないことに注意してください。

メモ： チャートデータテーブルの作成に関する詳しい情報については、Altova XMLSpy ならびに StyleVision 製品のドキュメンテーションを参照ください。

XSLT ドキュメント

以下にあるXSLT ドキュメントでは、Altova チャート拡張関数を使用することで円グラフを生成します。更に下に記されているXML ドキュメントを処理するために使用することができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:altovaext="http://www.altova.com/xslt-extensions"
   exclude-result-prefixes="#all">
   <xsl:output version="4.0" method="html" indent="yes" encoding="UTF-8"/>
   <xsl:template match="/">
      <html>
         <head>
            <title>
               <xsl:text>HTML Page with Embedded Chart</xsl:text>
            </title>
         </head>
         <body>
            <xsl:for-each select="/Data/Region[1]">
               <xsl:variable name="extChartConfig" as="item()*">
                  <xsl:variable name="ext-chart-settings" as="item()*">
                     <chart-config>
                        <General
                           SettingsVersion="1"
                           ChartKind="Pie3d"
                           BKColor="#ffffff"
                           ShowBorder="1"
                           PlotBorderColor="#000000"
                           PlotBKColor="#ffffff"
                           Title="{@id}"
                           ShowLegend="1"
                           OutsideMargin="32%"
                           TitleToPlotMargin="3%"
                           LegendToPlotMargin="6%"
                           >
                           <TitleFont
                              Color="#023d7d"
                              Name="Tahoma"
                              Bold="1"
                              Italic="0"
                              Underline="0"
                              MinFontHeight="10pt"
                              Size="8%" />
                        </General>
                     </chart-config>
                  </xsl:variable>
                  <xsl:sequence select="altovaext:create-chart-config-from-xml($ext-chart-
settings)"/>
               </xsl:variable>
               <xsl:variable name="chartDataSeries" as="item()*">
                  <xsl:variable name="chartDataRows" as="item()*">
                     <xsl:for-each select="Year">
                        <xsl:sequence select="altovaext:create-chart-data-row(@id, (.))"/>
```

```
                    </xsl:for-each>
                </xsl:variable>
                <xsl:variable name="chartDataSeriesNames" as="xs:string*" select=" ((&quot;Series
1&quot;),&apos;&apos;)[1]"/>
                <xsl:sequence
                    select="altova:ext:create-chart-data-series-from-rows($chartDataSeriesNames,
$chartDataRows)"/>
            </xsl:variable>
            <xsl:variable name="ChartObj" select="altova:ext:create-chart($extChartConfig,
($chartDataSeries),false())"/>
            <xsl:variable name="sChartFileName" select="'mychart1.png'"/>
            <img src="{$sChartFileName,altova:ext:save-chart-image($ChartObj,$sChartFileName,
400,400)}"/>
        </xsl:for-each>
    </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

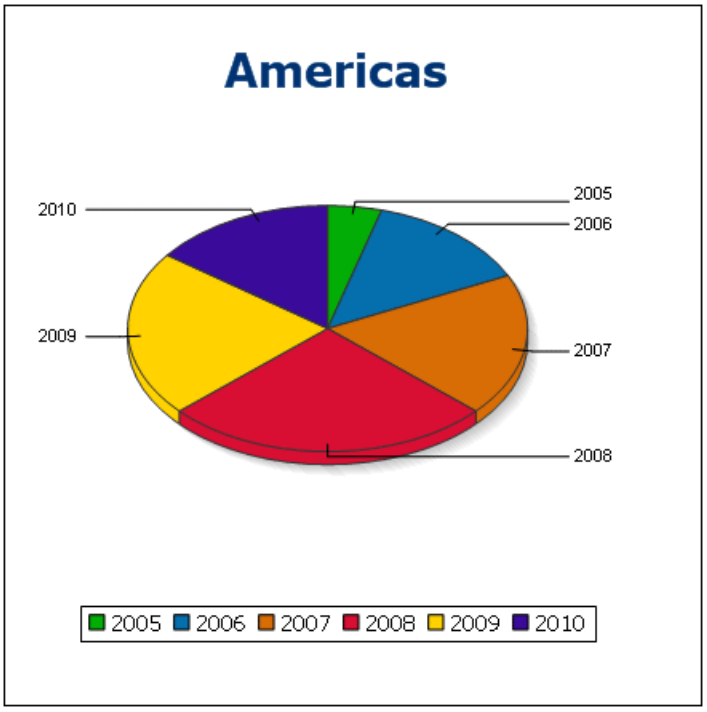
XML ドキュメント

上にあるXSLT ドキュメントにより以下のXML ドキュメントを処理することができます。XML ドキュメント内にあるデータを使用することで、下に示される円グラフを生成することができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="YearlySales.xsd">
    <ChartType>Pie Chart 2D</ChartType>
    <Region id="Americas">
        <Year id="2005">30000</Year>
        <Year id="2006">90000</Year>
        <Year id="2007">120000</Year>
        <Year id="2008">180000</Year>
        <Year id="2009">140000</Year>
        <Year id="2010">100000</Year>
    </Region>
    <Region id="Europe">
        <Year id="2005">50000</Year>
        <Year id="2006">60000</Year>
        <Year id="2007">80000</Year>
        <Year id="2008">100000</Year>
        <Year id="2009">95000</Year>
        <Year id="2010">80000</Year>
    </Region>
    <Region id="Asia">
        <Year id="2005">10000</Year>
        <Year id="2006">25000</Year>
        <Year id="2007">70000</Year>
        <Year id="2008">110000</Year>
        <Year id="2009">125000</Year>
        <Year id="2010">150000</Year>
    </Region>
</Data>
```

出力イメージ

上にあるXSLT ドキュメントを使ってXML ドキュメントを処理することで、以下にある円グラフが生成されます



2.2 その他の拡張関数

Java や C# などのプログラミング言語には、XPath 2.0 / XQuery 関数、または XSLT 2.0 関数として利用できない関数がいくつかあります。そのような関数の良い例として、Java で利用することのできる `sin()` や `cos()` という数学関数があります。XSLT スタイルシートや XQuery のクエリにてこれらの関数が利用できるのであれば、スタイルシートやクエリの適用範囲を大幅に拡張することができ、スタイルシート作成タスクの負担が大幅に軽減されます。Altova 製品で使用されている Altova エンジン (XSLT 1.0、XSLT 2.0、XQuery 1.0) では、Java や .NET および MSXSL scripts for XSLT における拡張関数の使用がサポートされます。このセクションでは、拡張機能および XSLT スタイルシート内で MSXSL スクリプト、および XQuery ドキュメントを使用する方法について記述します。使用できる拡張関数は以下のように構成されます :

- Java 拡張関数
- .NET 拡張関数
- XSLT に対する MSXSL スクリプト

記述の中では特に、(i) 関連するライブラリ内の関数がどのように呼ばれるか、(ii) 関数呼び出しを行う際に入力として使用される引数を変換するのにどのようなルールが適用され、return により値が返される際にどのような変換ルールが適用されるのか (XSLT/XQuery データオブジェクトに対する関数の結果) について説明されます。

必要条件

拡張関数のサポートを有効にするには、XSLT 変換や XQuery の実行を行うコンピューターに Java Runtime Environment (Java 関数にアクセスする場合) ならびに .NET Framework 2.0 以上 (.NET 関数にアクセスする場合) がインストールされている、またはアクセスできる環境が整っている必要があります。

2.2.1 Java 拡張関数

Java 拡張関数は、XPath または XQuery 条件式にて使用することができるほか、Java のコンストラクターを呼び出したり、Java の 静的 (または *インスタンス*) メソッドを呼び出すことができます。

Java クラスのフィールドは、引数を持たないメソッドとして扱われます。フィールドは静的または *インスタンス* として存在することができます。フィールドへのアクセス方法については、静的と *インスタンス* の両方について、以下のサブセクションにて記述されます。

このセクションは以下のサブセクションにより構成されます :

- Java : コンストラクター
- Java : 静的メソッドと静的フィールド
- Java : インスタンスメソッドとインスタンスフィールド
- データ型 : XPath/XQuery から Java へ
- データ型 : Java から XPath/XQuery へ

拡張関数のフォーム

XPath/XQuery 条件式における拡張関数では、`prefix:fname()` の形式を取る必要があります。

- `prefix:` 部により拡張関数が Java 関数として認識されます。`java:` から始まる URI のスコープ内の名前空間宣言に拡張関数を関連付けることで Java 関数であるという認識が行われます。名前空間の宣言により、例えば `xmlns:myns="java:java.lang.Math"` という Java クラスが特定されます。名前空間の宣言は、

(エラー無しの) `xmlns:myns="java"` という形式で、Java クラスの識別子を拡張関数にある `fname()` 部の左型に配置することでも行うことができます。

- `fname()` 部により、呼び出されている Java メソッドが識別され、メソッドの引数が提供されます (以下の例を参照ください)。`prefix:` 部にて識別された名前空間 URI が Java クラスを識別できなかった場合、Java クラスの識別はクラスの前にくる `fname()` 部にて行うことになり、ピリオドによりクラスから分離されることになります (以下にある2番目の XSLT サンプルを参照)。

メモ: 呼び出されるクラスはコンピューターのクラスパス上にある必要があります。

XSLT サンプル

以下に静的メソッドを呼び出す2つのサンプルを示します。最初のサンプルでは、クラス名 (`java.lang.Math`) が名前空間 URI に加えられており、`fname()` へ加えることはできません。2番目のサンプルでは、`prefix:` 部に `java:` が与えられており、`fname()` 部にてクラスとメソッドが識別されます。

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jmath="java"
              select="jmath:java.lang.Math.cos(3.14)" />
```

拡張関数内にあるメソッド名 (上の例では `cos()`) は、名前付き Java クラス (上の例では `java.lang.Math`) の public な静的メソッドの名前に一致する必要があります。

XQuery サンプル

以下に XSLT のサンプルに似た XQuery のサンプルを示します:

```
<cosine xmlns:jMath="java:java.lang.Math">
  {jMath:cos(3.14)}
</cosine>
```

ユーザー定義された Java クラス

独自の Java クラスやメソッドを作成した場合、(i) JAR ファイル (または class ファイル) を介してこれらクラスファイルへアクセスしているか、(ii) これら (JAR または class) ファイルが、カレントディレクトリ (XSLT や XQuery ドキュメントが存在するディレクトリ) に配置されているかにより、これらクラスの呼び出し方法が変わってきます。これらファイルの特定方法については、ユーザー定義クラスファイルならびにユーザー定義 JAR ファイルを参照ください。カレントディレクトリには無いクラスファイルや JAR ファイルへのパスは指定しなければならないことに注意してください。

ユーザー定義のクラスファイル

アクセスがクラスファイルを介したものである場合、4つのケースが考えられます:

- クラスファイルはパッケージである。XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている。(下のサンプルを参照)
- クラスファイルはパッケージでは無い。XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている。(下のサンプルを参照)

- クラスファイルはパッケージである。XSLT または XQuery ファイルがランダムな場所に収められている。 (下のサンプルを参照)
- クラスファイルはパッケージである。XSLT または XQuery ファイルがランダムな場所に収められている。 (下のサンプルを参照)

クラスファイルがパッケージではなく XSLT または XQuery ドキュメントと同じ場所に収められているケースを考えてみましょう。この場合、フォルダー内の全クラスを発見することができるため、ファイルの場所を指定する必要はありません。クラスの識別を行う構文は以下のようになります：

```
java:classname
```

ここで、

`java:` によりユーザー定義の Java 関数が呼ばれていることが示されます (デフォルトでカレントディレクトリにある Java クラスがロードされます)。

`classname` は目的となるメソッドのクラスが含まれているクラスの名前です。

クラスが名前空間 URI にて識別され、名前空間がメソッド呼び出しにて使用されます。

クラスファイルがパッケージで、XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている

以下の例では、`com.altova.extfunc` パッケージにある `Car` クラスの `getVehicleType()` メソッドが呼び出されています。com.altova.extfunc パッケージは `JavaProject` という名前のフォルダーに置かれており、XSLT ファイルも同じフォルダーに配置されています。

```
<xsl:stylesheet version="2.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:fn="http://www.w3.org/2005/xpath-functions"
        xmlns:car="java:com.altova.extfunc.Car" >
  <xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

  <xsl:template match="/">
    <a>
    <xsl:value-of select="car:getVehicleType()"/>
    </a>
  </xsl:template>

  </xsl:stylesheet>
```

クラスファイルがパッケージではなく、XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている

以下の例では、`com.altova.extfunc` パッケージにある Car クラスの `getVehicleType()` メソッドが呼び出されています。Car クラスファイルは `JavaProject/com/altova/extfunc` 以下に配置されています。XSLT ファイルも `JavaProject/com/altova/extfunc` フォルダーに配置されています。

```
<xsl:stylesheet version="2.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:fn="http://www.w3.org/2005/xpath-functions"
        xmlns:car="java:Car" >
```

```
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
  <a>
  <xsl:value-of select="car:getVehicleType()"/>
  </a>
</xsl:template>

</xsl:stylesheet>
```

クラスファイルがパッケージで、XSLT または XQuery ファイルがランダムな場所に収められている

以下の例では、com.altova.extfunc パッケージにあるCar クラスの getVehicleColor() メソッドが呼び出されています。com.altova.extfunc パッケージはJavaProject という名前のフォルダーに置かれており、XSLT ファイルが任意の場所に配置されています。この場合、以下のような構文でパッケージの場所をクエリ文字列としてURI 内にて指定する必要があります：

```
java:classname[?path=uri-of-package]
```

ここで、

java: によりユーザー定義のJava 関数が呼ばれていることを表します。
uri-of-package はJava パッケージのURI です。
classname は目的のメソッドが含まれているクラス名です。

クラスは名前空間 URI により特定され、名前空間がメソッド呼び出しのプレフィックスで使用されます。以下の例ではカレントディレクトリ以外にあるクラスファイルへのアクセスを行うことができます。

```
<xsl:stylesheet version="2.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:fn="http://www.w3.org/2005/xpath-functions"
        xmlns:car="java:com.altova.extfunc.Car?path=file:///C:/JavaProject/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
  <xsl:variable name="myCar" select="car:new('red')" />
  <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

クラスファイルがパッケージではなく、XSLT または XQuery ファイルがランダムな場所に収められている

以下の例では、com.altova.extfunc パッケージにあるCar クラスのgetCarColor() メソッドが呼び出されています。com.altova.extfunc パッケージはJavaProject という名前のフォルダーに置かれており、XSLT ファイルが任意の場所に配置されています。以下のような構文で、クラスファイルの場所をクエリ文字列としてURI 内にて指定する必要があります:

```
java:classname[?path=uri-of-classfile]
```

ここで

java: によりユーザー定義の Java 関数が呼ばれていることを表します。
uri-of-classfile は Java パッケージの URI です。
classname は目的のメソッドが含まれているクラス名です。

クラスは名前空間 URI により特定され、名前空間はメソッド呼び出しのプレフィックスで使用されます。以下の例ではカレントディレクトリ以外にあるクラスファイルへのアクセスを行うことができます。

```
<xsl:stylesheet version="2.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:fn="http://www.w3.org/2005/xpath-functions"
        xmlns:car="java:Car?path=file:///C:/JavaProject/com/altova/extfunc/" >

<xsl:output exclude-result-prefixes="fn car xs xsl"/>

<xsl:template match="/">
  <xsl:variable name="myCar" select="car:new('red')" />
  <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

メモ: パスが外部関数により与えられている場合、ClassLoader によりパスが追加されます。

ユーザー定義の JAR ファイル

JAR ファイル経由でアクセスが行われた場合、以下の構文により JAR ファイルの URI を指定する必要があります :

```
xmlns:classNS="java:classname?path=jar:uri-of-jarfile!/"
```

クラスの識別を行う名前空間 URI のプレフィックスを使用してメソッドが呼び出されます : `classNS:method()`

上の例に対する説明は以下のとおりです:

java: 関数が呼び出されていることを表します。
classname がユーザー定義されたクラスの名前になります。
? はクラス名とパスを分離するために使用されます。
path=jar: により、JAR ファイルへのパスが与えられていることを示します。
uri-of-jarfile は JAR ファイルの URI となります。
!/ は、終了を表すデリミタとなります。
classNS:method() により、メソッドの呼び出しが行われます。

その他にも、メソッド名とともにクラス名を与えることができます。構文の例を以下に示します :

```
xmlns:ns1="java:docx.layout.pages?path=jar:file:///c:/projects/
docs/docx.jar!/"
ns1:main()

xmlns:ns2="java?path=jar:file:///c:/projects/docs/docx.jar!/"
ns2:docx.layout.pages.main()
```

以下に JAR ファイルを使った Java 拡張関数を呼び出す XSLT サンプルを記します :

```
<xsl:stylesheet version="2.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
        xmlns:fn="http://www.w3.org/2005/xpath-functions"
        xmlns:car="java?path=jar:file:///C:/test/Car1.jar!/" >
<xsl:output exclude-result-prefixes="fn car xs lxs"/>

<xsl:template match="/">
        <xsl:variable name="myCar" select="car:Car1.new('red')" />
        <a><xsl:value-of select="car:Car1.getCarColor($myCar)"/></a>
</xsl:template>

<xsl:template match="car"/>

</xsl:stylesheet>
```

メモ： 拡張関数によりパスが与えられている場合、ClassLoader にパスが追加されます。

Java：コンストラクター

拡張関数を使用することで Java コンストラクターを呼び出すことができます。`new()` により全てのコンストラクターを呼び出すことができます。

Java コンストラクターの呼び出し結果を、[黙示的に XPath/XQuery データ型へ変換](#)できる場合、Java 拡張関数により XPath/XQuery データ型のシーケンスが返されます。Java コンストラクターの呼び出し結果が XPath/XQuery データ型へ変換できない場合、値を返すクラス名でラップした Java オブジェクトがコンストラクターにより作成されます。例えば、`java.util.Date` クラスに対するコンストラクターが呼び出された場合（`java.util.Date.new()`）、`java.util.Date` を持ったオブジェクトが返されます。返されたオブジェクトのレキシカルフォーマットは、XPath データ型のレキシカルフォーマットにマッチしない場合もあり、目的の XPath データ型に対するレキシカルフォーマットへ値の変換を行い、その後目的の XPath データ型へ変換を行う必要があります。

コンストラクターにより作成された Java オブジェクトにより 2 つのことが行えます：

- 変数への割り当てを行うことができます：
  ```
  <xsl:variable name="currentdate" select="date:new()" xmlns:date="java:java.util.Date" />
  ```
- 拡張関数への受け渡しを行うことができます（*インスタンスメソッドならびにインスタンスフィールド*を参照ください）：
  ```
  <xsl:value-of select="date:toString(date:new())" xmlns:date="java:java.util.Date" />
  ```

Java：静的メソッドと静的フィールド

静的メソッドは、Java 名ならびにメソッドの引数により直接呼び出すことができます。`E` や `PI` といった定数の静的フィールド（引数を持たないメソッド）は、引数を指定することなくアクセスすることができます。

XSLT の例

静的メソッドならびにフィールドを呼び出す例を以下に示します：

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(jMath:PI())" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:E() * jMath:cos(3.14)" />
```

上の拡張関数は prefix:fname() という形式を使用していることに注意してください。3つの例にあるプレフィックスは全てjMath: となっており、このプレフィックスは、java:java.lang.Math という名前空間 URI に関連付けられています。名前空間 URI は java: で開始しなければなりません。上の例では、クラス名 (java.lang.Math) を含むように拡張されています。拡張関数の fname() 部は (java.lang.Math のような) public クラスにマッチする必要があり、その後 public な静的メソッドが引数とともに続くか (例：cos(3.14))、public な静的フィールドが続きます (例：PI())。

上の例では、クラス名が名前空間 URI に含まれています。クラス名が名前空間 URI に含まれていない場合、以下の例にあるように、拡張関数の fname() 部にて追加する必要があります：

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math.cos(3.14)" />
```

XQuery の例

XQuery における似たようなサンプルを以下に示します：

```
<cosine xmlns:jMath="java:java.lang.Math">
  {jMath:cos(3.14)}
</cosine>
```

Java：インスタンスメソッドとインスタンスフィールド

メソッド呼び出しの第一引数としてパースされる Java オブジェクトが、*インスタンスメソッド*には与えられています。このような Java オブジェクトは通常、拡張関数を使うことで作成される (例：コンストラクターの呼び出し) か、スタイルシートパラメーター変数により作成されます。以下に XSLT サンプルを示します：

```
<xsl:stylesheet version="1.0" exclude-result-prefixes="date"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:date="java:java.util.Date"
  xmlns:jlang="java:java.lang">
  <xsl:param name="CurrentDate" select="date:new()"/>
  <xsl:template match="/">
    <enrollment institution-id="Altova School"
          date="{date:toString($CurrentDate)}"
          type="{jlang:Object.toString(jlang:Object.getClass(date:new()))}">
    </enrollment>
  </xsl:template>
</xsl:stylesheet>
```

上の例では、ノード enrollment/@type の値が以下のように作成されます：

1. java.util.Date クラスに対するオブジェクトがコンストラクター (date:new() コンストラクター) とともに作成されます。
2. jlang.Object.getClass メソッドの引数として Java オブジェクトがパースされます。
3. getClass メソッドにより得られたオブジェクトが jlang.Object.toString メソッドの引数としてパースされます。

結果 (type の値) は java.util.Date を持った文字列となります。

*インスタンスフィールド*は、引数としてインスタンスフィールドへ渡される Java オブジェクトではないという点で、理論的にはインスタンスメソッドと異なります。パラメーターや変数がその代わりに引数として渡されますが、パラメーター変数そのものに Java オブジェクトから返された値が含まれている場合もあります。例えば、CurrentDate パラメーターには java.util.Date クラスのコンストラクターから返された値が含まれます。この値は引数として、date:toString インスタンスメソッドへ渡され、/enrollment/@date の値として使用されます。

データ型：XPath/ XQuery から Java へ

XPath/XQuery 条件式内部から Java 関数が呼び出された場合、複数ある同名の Java クラスのうち、どのクラスが呼び出されたのが決定するのに、関数へ渡される引数のデータ型が重要になります。

Java では、以下のルールが適用されます：

- 同名の Java メソッドが2つ以上あり、それぞれが違う数の引数を受け取る場合、呼び出しに使用されている引数の数に一番マッチするメソッドが選択されます。
- XPath/XQuery の文字列、数値、boolean データ型は、黙示的に対応する Java データ型へ変換されます（以下のリストを参照）。与えられた XPath/XQuery 型が2つ以上の Java 型へ変換できる場合（例：`xs:integer`）、選択されたメソッドにて宣言されている Java 型が使用されます。例えば、呼び出された Java メソッドが `fx(decimal)` で、与えられた XPath/XQuery データ型が `xs:integer` の場合、`xs:integer` が Java の `decimal` データ型へ変換されます。

以下のテーブルに、XPath/XQuery の文字列、数値、boolean 型から Java データ型への黙示的な変換リストを示します。

`xs:string`	`java.lang.String`
`xs:boolean`	`boolean` (プリミティブ型), `java.lang.Boolean`
`xs:integer`	`int, long, short, byte, float, double,` ならびに `java.lang.Integer` のようなこれらのラッパークラス
`xs:float`	`float` (プリミティブ型), `java.lang.Float`, `double` (プリミティブ型)
`xs:double`	`double` (プリミティブ型), `java.lang.Double`
`xs:decimal`	`float` (プリミティブ型), `java.lang.Float`, `double`(プリミティブ型), `java.lang.Double`

上のリストにある XML スキーマデータ型（ならびに XPath や XQuery で使用されているデータ型）のサブタイプも、対応する祖先のサブタイプとして Java のデータ型へ変換されます。

場合によっては、与えられた情報から正しい Java メソッドを選択することができない場合もあります。例えば、以下のような場合を考えてみましょう：

- 与えられた引数が 10 という値を持った xs:untypedAtomic 型で、`mymethod(float)` メソッドへ渡されるのを意図している。
- しかし、そのクラスには別のデータ型を取る `mymethod(double)` というメソッドも存在する。
- メソッド名が同じで、与えられた型（`xs:untypedAtomic`）も `float` と `double` の両方に変換することができるため、`xs:untypedAtomic` が `float` ではなく `double` に変換される可能性もある。
- 結果として、意図したメソッドは選択されず、予期しない動作結果がを招く可能性がある。この問題を回避するには、意図したメソッドを使用するユーザー定義のメソッドを別の名前で新たに作成する必要があります。

上のリストでカバーされていない型（例：`xs:date`）は変換されず、エラーとなります。しかし場合によっては、Java コンストラクターを使用して、目的の Java データ型を作成することが可能だということも留意してください。

データ型：Java から XPath/ XQuery へ

Java メソッドにより値が返され、値のデータ型が文字列、数値、または boolean 型の場合、対応する XPath/XQuery 型への変換が行われます。例えば、Java の `java.lang.Boolean` や `boolean` データ型は `xsd:boolean` へ変換されます。

関数から返された一次元配列は、シーケンス (sequence) に展開されます。2次元以上の配列は変換されることが無いため、ラップして使用するべきでしょう。

Java オブジェクトや文字列、数値、boolean 以外のデータ型がラップされて返された場合、最初に (例えば `toString` という) Java メソッドを使用して Java オブジェクトを文字列へ変換することで、目的の XPath/XQuery 型への変換を行います。XPath/XQuery では、文字列を目的となる型のレキシカルフォーマットへ変換し、目的の型への変換を (例えば `cast as` 式を使用することで) 行うことができます。

2.2.2 .NET 拡張関数

.NET プラットフォームにて作業を行なっている場合、.NET 言語 (例えば C#) で記述された拡張関数を使用することができます。.NET 拡張関数は XPath や XQuery 条件式内部から使用することができ、.NET クラス内部にあるコンストラクターや (static または *インスタンス* 変数) プロパティを呼び出すことができます。

`get_PropertyName()` 構文を使用することにより .NET クラスのプロパティを呼び出すことができます。

このセクションは、以下のサブセクションにより構成されています：

- .NET コンストラクター
- .NET 静的メソッドならびに静的フィールド
- .NET：インスタンスメソッドとインスタンスフィールド
- データ型：XPath/XQuery から .NET へ
- データ型：.NET から XPath/XQuery へ

拡張関数のフォーム

XPath/XQuery 条件式にある拡張関数は、`prefix:fname()` の形式を取る必要があります。

- `prefix:` 部は、呼び出されている .NET クラスを特定する URI となります。
- `fname()` 部により、.NET クラス内にあるコンストラクター、プロパティ、または (静的または *インスタンス*) メソッドが特定され、必要な場合は引数が与えられます。
- URI は `clitype:` で開始する必要があり、これにより関数が .NET 拡張関数であることが認識されます。
- 拡張関数の `prefix:fname()` 形式は、*システム* クラスならびにロードされたアセンブリとともに使用することもできます。しかしクラスをロードする必要がある場合、必要な情報が含まれるパラメーターが必要になります。

パラメーター

アセンブリをロードするには以下のパラメーターを使用してください：

`asm`	ロードするアセンブリの名前。
`ver`	バージョン番号 (ピリオドにより分離された最大 4 桁の整数)。
`sn`	アセンブリ厳密名のキートークン (16 桁の数値)。

from	ロードするアセンブリ (DLL) の場所を特定するURI。URI が相対パスの場合、XSLT や XQuery ドキュメントに対して相対的になります。このパラメーターが指定された場合、その他のパラメーターが無視されます。
partialname	アセンブリ名の一部。Assembly.LoadWith.PartialName() へ渡され、アセンブリへのロードが試みられます。 partialname が指定された場合、その他のパラメーターが無視されます。
loc	例えば en-US というロケール。デフォルトは neutral です。

アセンブリが DLL からロードされる場合、 from パラメータが使用して、sn パラメーターは使用しないでください。アセンブリがグローバルアセンブリキャッシュ (GAC) からロードされる場合、sn パラメータを使用して from パラメーターは使用しないでください。

最初のパラメーターの前に疑問符 (?) を挿入し、パラメーター同士はセミコロンで分離する必要があります。パラメーター名へ値を受け渡すには、統合符号 (=) を使用します (以下の例を参照ください)。

名前空間宣言の例

XSLT において、システムクラス System.Environment を特定する名前空間宣言の例を以下に示します :

```
xmlns:myns="clitype:System.Environment"
```

XSLT において、ロードするクラスを Trade.Forward.Scrip として特定する名前空間宣言の例を以下に示します :

```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

XQuery において、システムクラス MyManagedDLL.testClass を特定する名前空間宣言の例を以下に示します。2つのケースが考えられます :

1. アセンブリが GAC からロードされた場合 :

```
declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL;
        ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";
```

2. アセンブリが DLL からロードされた場合 (完全参照と一部の参照) :

```
            declare namespace cs="clitype:MyManagedDLL.testClass?
from=file:///C:/Altova
            Projects/extFunctions/MyManagedDLL.dll;

     declare namespace cs="clitype:MyManagedDLL.testClass?
     from=MyManagedDLL.dll;
```

XSLT の例

システムクラス System.Math 内の関数を呼び出すための完全な XSLT の例を以下に示します :

```
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:fn="http://www.w3.org/2005/xpath-functions">
  <xsl:output method="xml" omit-xml-declaration="yes" />
  <xsl:template match="/">
    <math xmlns:math="clitype:System.Math">
      <sqrt><xsl:value-of select="math:Sqrt(9)"/></sqrt>
      <pi><xsl:value-of select="math:PI()"/></pi>
```

```
      <e><xsl:value-of select="math:E()"/></e>
      <pow><xsl:value-of select="math:Pow(math:PI(),math:E())"/></pow>
    </math>
  </xsl:template>
</xsl:stylesheet>
```

math 要素にある名前空間宣言により、math: プレフィックスと clitype:System.Math URI が関連付けられます。URI の最初にある clitype: により、それ以降の記述がシステムクラスまたはロードされたクラスを特定するものであることが示されます。XPath 条件式にある math: プレフィックスにより、拡張関数が URI (そしてクラス) System.Math に関連付けられます。拡張関数により、System.Math クラス内のメソッドが特定され、必要な場所に引数が与えられます。

XQuery の例

上の XSLT に対する例と同様の XQuery 例を以下に示します :

```
<math xmlns:math="clitype:System.Math">
  {math:Sqrt(9)}
</math>
```

上の XSLT と同様に、名前空間宣言により .NET クラス (この場合はシステムクラス) が特定されます。XQuery 式により呼び出されるメソッドが特定され、引数が与えられます。

.NET コンストラクター

拡張関数を使用することで、.NET コンストラクターを呼び出すことができます。new() により全てのコンストラクターを呼び出すことができます。クラス内に 2 つ以上のコンストラクターがある場合、与えられた引数の数が最もマッチするコンストラクターが選択されます。与えられた引数に対してマッチするコンストラクターが見つからない場合、"No constructor found" エラーが返されます。

XPath/XQuery データ型を返すコンストラクター

.NET コンストラクター呼び出しの結果が XPath/XQuery データ型へ黙示的に変換することができる場合、.NET 拡張関数から XPath/XQuery データ型のシーケンスが返されます。

.NET オブジェクトを返すコンストラクター

.NET コンストラクター呼び出しの結果が XPath/XQuery データ型へ適切に変換できない場合、値を返すクラス名でラップした .NET オブジェクトがコンストラクターにより作成されます。例えば、System.DateTime クラスのコンストラクターが (System.DateTime.new() により) 呼ばれた場合 System.DateTime 型を持ったオブジェクトが返されます。

返されたオブジェクトのレキシカルフォーマットは、目的の XPath データ型と違っている場合があります。その場合、返された値を : (i) 目的の XPath データ型のレキシカルフォーマットへ変換し、(ii) 目的の XPath データ型へキャストする必要があります。

コンストラクターにより作成された .NET オブジェクトに対して 3 つのことを行うことができます :

- 変数内で使用することができます :
  ```
  <xsl:variable name="currentdate" select="date:new(2008,4,29)"
  xmlns:date="clitype:System.DateTime" />
  ```
- 拡張関数へ渡すことができます (*インスタンスメソッドとインスタンスフィールド* を参照ください):

```
<xsl:value-of select="date:ToString(date:new(2008,4,29))"
  xmlns:date="clitype:System.DateTime" />
```

- 文字列、数値、または boolean へ変換することができます：

```
<xsl:value-of select="xs:integer(date:get_Month(date:new(2008,4,29)))"
  xmlns:date="clitype:System.DateTime" />
```

.NET：静的メソッドと静的フィールド

メソッド名と引数を与えることで、静的メソッドを直接呼び出すことができます。呼び出しに使用される名前は、クラス内にある public static メソッドと完全に一致する必要があります。関数の呼び出しに使用されたメソッド名と引数の数にマッチするものがクラス内に複数ある場合、与えられた引数が評価され、最もマッチするものが選択されます。マッチする結果が得られない場合、エラーが返されます。

メモ： .NET クラス内にあるフィールドは、引数を持たないメソッドとしてみなされます。プロパティは `get_PropertyName()` 構文により呼び出されます。

例

1つの引数とともにメソッド（`System.Math.Sin(arg)`）を呼び出す XSLT サンプルを以下に示します：

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

（引数なしのメソッドとしてみなされる）フィールド（`System.Double.MaxValue()`）を呼び出す XSLT サンプルを以下に示します：

```
<xsl:value-of select="double:MaxValue()" xmlns:double="clitype:System.Double"/>
```

（`get_PropertyName()` 構文を使って）プロパティ（`System.String()`）を呼び出す XSLT サンプルを以下に示します：

```
<xsl:value-of select="string:get_Length('my string')" xmlns:string="clitype:System.String"/>
```

1つの引数とともにメソッド（`System.Math.Sin(arg)`）を呼び出す XQuery サンプルを以下に示します：

```
<sin xmlns:math="clitype:System.Math">
  {math:Sin(30)}
</sin>
```

.NET：インスタンスメソッドとインスタンスフィールド

インスタンスメソッドには、メソッド呼び出しの第一引数として .NET オブジェクトが渡されます。通常この .NET オブジェクトは、拡張関数（例えばコンストラクター呼び出し）またはスタイルシートパラメーター変数により作成されます。以下に XSLT の例を示します：

```
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:fn="http://www.w3.org/2005/xpath-functions">
  <xsl:output method="xml" omit-xml-declaration="yes"/>
  <xsl:template match="/">
    <xsl:variable name="releasedate"
      select="date:new(2008,4,29)"
      xmlns:date="clitype:System.DateTime"/>
```

```
    <doc>
      <date>
        <xsl:value-of select="date:ToString(date:new (2008,4,29))"
          xmlns:date="clitype:System.DateTime"/>
      </date>
      <date>
        <xsl:value-of select="date:ToString($releasedate)"
          xmlns:date="clitype:System.DateTime"/>
      </date>
    </doc>
  </xsl:template>
</xsl:stylesheet>
```

上の例では、System.DateTime コンストラクター (`new(2008, 4, 29)`) が、System.DateTime 型の .NET オブジェクトの作成に使用されます。このオブジェクトは、最初に `releasedate` 変数の値として、次に `System.DateTime.ToString()` メソッドの引数として作成されます。`System.DateTime.ToString()` インスタンスメソッドは、`System.DateTime` コンストラクターの (new(2008, 4, 29)) における引数として2度呼び出されます。これらインスタンスにおいて `releasedate` 変数が .NET オブジェクトを取得するのに使用されます。

インスタンスメソッドとインスタンスフィールド

インスタンスメソッドとインスタンスフィールドの違いは理論的なものです。インスタンスメソッドでは .NET オブジェクトが直接引数に渡され、インスタンスフィールドではパラメーターや変数が (.NET オブジェクトそのものを含めることはできるものの) 代わりに渡されます。例えば上の例では、`releasedate` 変数に .NET オブジェクトが含まれており、この変数が2番目の date 要素コンストラクターにて `ToString()` の引数として渡されます。そのため、最初の date 要素にある `ToString()` インスタンスがインスタンスメソッドであるのに対し、2番目はインスタンスフィールドとしてみなされます。両方のインスタンスで求められる結果は等価です。

データ型 : XPath/ XQuery から .NET へ

.NET 拡張関数が XPath/XQuery 条件式内部で使用された場合、複数ある .NET メソッドのうち、どれが呼び出されたのか決定するのに関数の引数に使用されるデータ型が重要になります。

.NET では、以下のルールが適用されます :

- クラス内に同名のメソッドが2つ以上ある場合、呼び出しに使用された引数の数がマッチするメソッドだけが、呼び出される関数の候補に狭められます。
- XPath/XQuery の文字列、数値、boolean データ型は黙示的に対応する .NET データ型へ変換されます (以下のリストを参照)。与えられた XPath/XQuery 型が2つ以上の .NET 型へ変換できる場合 (例 : `xs:integer`)、選択されたメソッドにて宣言されている .NET 型が使用されます。例えば、呼び出された .NET メソッドが fx(fouble) で、与えられた XPath/XQuery データ型が xs:integer の場合、`xs:integer` が .NET の `double` データ型へ変換されます。

以下のテーブルに、XPath/XQuery の文字列、数値、boolean 型から .NET データ型へ行われる黙示的な変換リストを示します。

`xs:string`	`StringValue, string`
`xs:boolean`	`BooleanValue, bool`
`xs:integer`	`IntegerValue, decimal, long, integer, short, byte, double, float`
`xs:float`	`FloatValue, float, double`

xs:double	DoubleValue, double
xs:decimal	DecimalValue, decimal, double, float

上のリストにある XML スキーマ型 (ならびに XPath や XQuery で使用されているデータ型) のサブタイプも、対応する祖先のサブタイプとして .NET のデータ型へ変換されます。

場合によっては、与えられた情報から正しい .NET メソッドを選択することができない場合もあります。例えば、以下のような場合を考えてみましょう:

- 与えられた引数が 1 つある xs:untypedAtomic 値で、mymethod(float) メソッドへ渡されるのを意図している。
- しかし、そのクラスには別のデータ型をとる mymethod(double) というメソッドも存在する。
- メソッド名が同じで、与えられた型 (xs:untypedAtomic) も float と double の両方に変換することができるため、xs:untypedAtomic が float ではなく double に変換される可能性もある。
- 結果として、意図したメソッドは選択されず、予期しない動作結果が招く可能性がある。この問題を回避するには、意図したメソッドを使用するユーザー定義のメソッドを別の名前で新たに作成する必要があります。

上のリストでカバーされていない型 (例 : xs:date) は変換されずエラーとなります。

データ型 : .NET から XPath/ XQuery へ

.NET メソッドにより値が返される際に値のデータ型が文字列、数値、または boolean 型の場合、対応する XPath/ XQuery 型への変換が行われます。例えば、 .NET の decimal データ型は xsd:decimal へ変換されます。

.NET オブジェクトや文字列、数値、boolean 以外のデータ型が返された場合、最初に (例えば System.DateTime.ToString() といった) .NET メソッドを使用して .NET オブジェクトを文字列へ変換します。XPath/XQuery では、文字列を目的となる型のレキシカルフォーマットへ変換し、目的の型への変換を (例えば cast as 式を使用することで) 行うことができます。

2.2.3 XSLT に対する MSXSL スクリプト

<msxsl:script> 要素にはユーザー定義の関数や変数が含まれており、XSLT スタイルシート内の XPath 条件式内部から呼び出しを行うことができます。<msxsl:script> はトップレベル要素で、<xsl:stylesheet> または <xsl:transform> の子要素である必要があります。

<msxsl:script> 要素は urn:schemas-microsoft-com:xslt 名前空間内に存在する必要があります (以下を参照ください)。

スクリプト言語と名前空間

ブロック内で使用されるスクリプト言語は <msxsl:script> 要素の language 属性にて指定され、XPath 条件式における関数の呼び出しに対して使用される名前空間は implements-prefix 属性により特定されます (以下を参照)。

```
<msxsl:script language="scripting-language" implements-prefix="user-namespace-prefix">

   function-1 or variable-1
   ...
   function-n or variable-n
```

```
</msxsl:script>
```

<msxsl:script> 要素は Windows Scripting Runtime を使ったやりとりを行うため、お使いのコンピューターにインストールされた言語だけが <msxsl:script> 要素では使用することができません。**MSXSL スクリプトを使用するには .NET Framework 2.0 以上のプラットフォームをインストールする必要があります。結果として** <msxsl:script> 言語から .NET スクリプト言語を使用することができます。

HTML の <script> 要素における language 属性と同じ値が language 属性では受理されます。language 属性が指定されていない場合、Microsoft JScript がデフォルトとして想定されます。

implements-prefix 属性には名前空間スコープ内で宣言されたプレフィックスが与えられます。通常この名前空間は関数ライブラリのために予約されたユーザーの名前空間となります。<msxsl:script> 要素内で定義された全ての関数ならびに変数は、implements-prefix 属性にて指定されたプレフィックスで特定される名前空間に収められます。XPath 条件式内部から関数が呼ばれる場合、完全修飾関数名が同じ名前空間内に関数として定義されていなければなりません。

例

<msxsl:script> 要素内で定義された関数を使用する XSLT スタイルシートの例を以下に示します：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:fn="http://www.w3.org/2005/xpath-functions"
  xmlns:msxsl="urn:schemas-microsoft-com:xslt"
  xmlns:user="http://mycompany.com/mynamespace">

 <msxsl:script language="VBScript" implements-prefix="user">
  <![CDATA[
  'Input: A currency value: the wholesale price
  'Returns: The retail price: the input value plus 20% margin,
  'rounded to the nearest cent
  dim a as integer = 13
  Function AddMargin(WholesalePrice) as integer
    AddMargin = WholesalePrice * 1.2 + a
  End Function
 ]]>
 </msxsl:script>

 <xsl:template match="/">
  <html>
   <body>
    <p>
     <b>Total Retail Price =
      $<xsl:value-of select="user:AddMargin(50)"/>
     </b>
     <br/>
     <b>Total Wholesale Price =
      $<xsl:value-of select="50"/>
     </b>
    </p>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

データ型

スクリプトブロックとのやりとりに使用されるパラメーターの値は XPath データ型に限定されます。スクリプトブロック内にある関数にてやりとりされるデータや変数に、この制限はありません。

アセンブリ

msxsl:assembly 要素を使用することで、アセンブリをスクリプト内部へインポートすることができます。アセンブリは名前や URI により特定されます。アセンブリのインポートは、コンパイル時に行われます。以下に **msxsl:assembly** 要素の簡単な使用例を示します：

```
<msxsl:script>
       <msxsl:assembly name="myAssembly.assemblyName" />
       <msxsl:assembly href="pathToAssembly" />

       ...

</msxsl:script>
```

アセンブリ名は、以下のような完全な名前でも：:

```
"system.Math, Version=3.1.4500.1 Culture=neutral
PublicKeyToken=a46b3f648229c514"
```

`"myAssembly.Draw"` のような短い名前でも指定できます。

名前空間

`msxsl:using` 要素により名前空間の宣言を行うことができます。これにより、スクリプト内において名前空間無しでアセンブリクラスを使用することができ、タイピングの手間を軽減することができます。以下に `msxsl:using` 要素の簡単な使用例を示します：

```
<msxsl:script>
       <msxsl:using namespace="myAssemblyNS.NamespaceName" />

       ...

</msxsl:script>
```

`namespace` 属性の値は名前空間の名前となります。

3 DB 内で生成された XML スキーマのデータ型

XML スキーマがデータベース(DB)から生成されると、そのデータベース特有のデータ型が XML スキーマデータ型に変換されます。一般に使用される DB データ型から XML スキーマデータ型へのマッピングは付録に記載されています。以下のリストから選択してください。

- ADO
- MS Access
- MS SQL Server
- MySQL
- ODBC
- Oracle
- Sybase

3.1 ADO

XML スキーマが ADO データベース (DB) から生成される場合、ADO DB データ型は下のテーブルの通り XML スキーマデータ型に変換されます。

ADO データ型	XML スキーマデータ型
adGUID	xs:ID
adChar	xs:string
adWChar	xs:string
adVarChar	xs:string
adWVarChar	xs:string
adLongVarChar	xs:string
adWLongVarChar	xs:string
adVarWChar	xs:string
adBoolean	xs:boolean
adSingle	xs:float
adDouble	xs:double
adNumeric	xs:decimal
adCurrency	xs:decimal
adDBTimeStamp	xs:dateTime
adDate	xs:date
adBinary	xs:base64Binary
adVarBinary	xs:base64Binary
adLongVarBinary	xs:base64Binary
adInteger	xs:Integer
adUnsignedInt	xs:unsignedInt
adSmallInt	xs:short
adUnsignedSmallInt	xs:unsignedShort
adBigInt	xs:long
adUnsignedBigInt	xs:unsignedLong
adTinyInt	xs:byte
adUnsignedTinyInt	xs:unsignedByte

3.2 MS Access

XML スキーマが MS Access データベース (DB) から生成される場合、MS Access DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

MS Access データ型	XML スキーマデータ型
GUID	xs:ID
char	xs:string
varchar	xs:string
memo	xs:string
bit	xs:boolean
Number(single)	xs:float
Number(double)	xs:double
Decimal	xs:decimal
Currency	xs:decimal
Date/Time	xs:dateTime
Number(Long Integer)	xs:integer
Number(Integer)	xs:short
Number(Byte)	xs:byte
OLE Object	xs:base64Binary

3.3 MS SQL Server

XML スキーマがMS SQL Server データベース (DB) から生成される場合、MS SQL Server DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

MS SQL Server データ型	XML スキーマデータ型
uniqueidentifier	xs:ID
char	xs:string
nchar	xs:string
varchar	xs:string
nvarchar	xs:string
text	xs:string
ntext	xs:string
sysname	xs:string
bit	xs:boolean
real	xs:float
float	xs:double
decimal	xs:decimal
money	xs:decimal
smallmoney	xs:decimal
datetime	xs:dateTime
smalldatetime	xs:dateTime
binary	xs:base64Binary
varbinary	xs:base64Binary
image	xs:base64Binary
integer	xs:integer
smallint	xs:short
bigint	xs:long
tinyint	xs:byte

3.4 MySQL

XML スキーマが MySQL データベース (DB) から生成される場合、MySQL DB データ型は下のテーブルの通り XML スキーマデータ型に変換されます。

MySQL データ型	XML スキーマデータ型
char	xs:string
varchar	xs:string
text	xs:string
tinytext	xs:string
mediumtext	xs:string
longtext	xs:string
tinyint(1)	xs:boolean
float	xs:float
double	xs:double
decimal	xs:decimal
datetime	xs:dateTime
blob	xs:base64Binary
tinyblob	xs:base64Binary
mediumblob	xs:base64Binary
longblob	xs:base64Binary
smallint	xs:short
bigint	xs:long
tinyint	xs:byte

3.5 ODBC

XML スキーマが ODBC データベース (DB) から生成される場合、ODBC DB データ型は下のテーブルの通り XML スキーマデータ型に変換されます。

ODBC データ型	XML スキーマデータ型
SQL_GUID	xs:ID
SQL_CHAR	xs:string
SQL_VARCHAR	xs:string
SQL_LONGVARCHAR	xs:string
SQL_BIT	xs:boolean
SQL_REAL	xs:float
SQL_DOUBLE	xs:double
SQL_DECIMAL	xs:decimal
SQL_TIMESTAMP	xs:dateTime
SQL_DATE	xs:date
SQL_BINARY	xs:base64Binary
SQL_VARBINARY	xs:base64Binary
SQL_LONGVARBINARY	xs:base64Binary
SQL_INTEGER	xs:integer
SQL_SMALLINT	xs:short
SQL_BIGINT	xs:long
SQL_TINYINT	xs:byte

3.6 Oracle

XML スキーマがOracle データベース (DB) から生成される場合、Oracle DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

Oracle データ型	XML スキーマデータ型
ROWID	xs:ID
CHAR	xs:string
NCHAR	xs:string
VARCHAR2	xs:string
NVARCHAR2	xs:string
CLOB	xs:string
NCLOB	xs:string
NUMBER (with check constraint applied)*	xs:boolean
NUMBER	xs:decimal
FLOAT	xs:double
DATE	xs:dateTime
INTERVAL YEAR TO MONTH	xs:gYearMonth
BLOB	xs:base64Binary

* データ型 NUMBER の列にチェック制約が適用されている場合、またチェック制約が 0 または 1 の値をチェックする場合、この列のNUMBER データ型は、xs:boolean のXML スキーマデータ型に変換されます。このメカニズムは、生成されたXML スキーマ内にxs:boolean データ型を生成する際に役に立ちます。

3.7 Sybase

XML スキーマがSybase データベース (DB) から生成される場合、Sybase DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

Sybase データ型	XML スキーマデータ型
char	xs:string
nchar	xs:string
varchar	xs:string
nvarchar	xs:string
text	xs:string
sysname-varchar(30)	xs:string
bit	xs:boolean
real	xs:float
float	xs:float
double	xs:double
decimal	xs:decimal
money	xs:decimal
smallmoney	xs:decimal
datetime	xs:dateTime
smalldatetime	xs:dateTime
timestamp	xs:dateTime
binary<=255	xs:base64Binary
varbinary<=255	xs:base64Binary
image	xs:base64Binary
integer	xs:integer
smallint	xs:short
tinyint	xs:byte

4 XML スキーマから生成された DB のデータ型

データベース構造がXML スキーマから生成されると、そのデータベース特有のデータ型がXML スキーマデータ型から生成されます。一般に使用されるXML スキーマデータ型からDB データ型へのマッピングは付録に記載されています。以下のリストから選択してください。

- MS Access
- MS SQL Server
- MySQL
- Oracle

4.1 MS Access

MS Access データベース (DB) が XML スキーマから生成される場合、MS Access データ型は下のテーブルの通り XML スキーマデータ型に変換されます。

XML スキーマデータ型	MS Access データ型
xs:ID	GUID
xs:string	If no facets varchar (255)
	Size = either length or maxLength
	If Size <= 255 varchar (n)
	else memo
xs:normalizedString	Same as xs:string
xs:token	Same as xs:string
xs:Name	Same as xs:string
xs:NCName	Same as xs:string
xs:anyURI	Same as xs:string
xs:QName	Same as xs:string
xs:NOTATION	Same as xs:string
xs:boolean	bit
xs:float	Number (single)
xs:double	Number (double)
xs:decimal	Decimal
xs:duration	Date/Time
xs:dateTime	Date/Time
xs:time	Date/Time
xs:date	Date/Time
xs:gYearMonth	Date/Time
xs:gYear	Date/Time
xs:gMonthDay	Date/Time
xs:gDay	Date/Time
xs:gMonth	Date/Time
xs:hexBinary	If no facets varbinary (255)
	Size = either length or maxLength
	If Size <= 8000 varbinary
	else image (OLE Object)
xs:base64Binary	Same as xs:hexBinary
xs:integer	Number (Long Integer)
xs:int	Number (Long Integer)
xs:negativeInteger	Number (Long Integer); value

	constraint
xs:positiveInteger	Number (Long Integer); value constraint
xs:nonNegativeInteger	Number (Long Integer); value constraint
xs:nonPositiveInteger	Number (Long Integer); value constraint
xs:unsignedInt	Number (Long Integer)
xs:short	-- no equivalent --
xs:unsignedShort	-- no equivalent --
xs:long	-- no equivalent --
xs:unsignedLong	-- no equivalent --
xs:byte	Number (Byte)
xs:unsignedByte	Number (Byte)

4.2 MS SQL Server

MS SQL Server データベース (DB) が XML スキーマから生成される場合、MS SQL Server データ型は下のテーブルの通り XML スキーマデータ型に変換されます。

XML スキーマデータ型	MS SQL Server データ型
ID	uniqueidentifier
xs:string	If no facets
	{ if UNICODE nvarchar (255)
	else varchar (255) }
	else
	{ if UNICODE
	(Size = either length or maxLength)
	If Size <= 4000
	if FacetLengthIsSet then nChar
	else nVarChar
	if Size <= 1073741823 then nText }
	else
	{ if NON-UNICODE
	(Size = either length or maxLength)
	If Size <= 8000
	if FacetLengthIsSet then char
	else varchar
	if Size <= 2147483647 then text }
xs:normalizedString	Same as xs:string
xs:token	Same as xs:string
xs:Name	Same as xs:string
xs:NCName	Same as xs:string
xs:anyURi	Same as xs:string
xs:QName	Same as xs:string
xs:NOTATION	Same as xs:string
xs:boolean	bit
xs:float	real
xs:double	float
xs:decimal	decimal
xs:duration	datetime
xs:dateTime	datetime
xs:time	datetime

xs:date	datetime
xs:gYearMonth	datetime
xs:gYear	datetime
xs:gMonthDay	datetime
xs:gDay	datetime
xs:gMonth	datetime
xs:hexBinary	If no facets varbinary (255)
	(Size = either length or maxLength
	If Size <= 8000
	if FacetLengthIsSet then binary
	else varbinary
	if Size <= 2147483647 then image
xs:base64Binary	Same as xs:hexBinary
xs:integer	int
xs:int	int
xs:negativeInteger	Int (constrained to {...,-2,-1})
xs:positiveInteger	Int (constrained to {1,2,...})
xs:nonNegativeInteger	int (constrained to {0,1,2,...})
xs:nonPositiveInteger	int (constrained to {...,-2,-1,0})
xs:unsignedInt	int (additional constraints)
xs:short	smallint
xs:unsignedShort	smallint (additional constraints)
xs:long	bigint
xs:unsignedLong	bigint (additional constraints)
xs:byte	tinyint
xs:unsignedByte	tinyint (additional constraints)

4.3 MySQL

MySQL データベース (DB) が XML スキーマから生成される場合、MySQL データ型は下のテーブルの通り XML スキーマデータ型に変換されます。

XML スキーマデータ型	MySQL データ型
xs:ID	varchar(255)
xs:string	If no facets then varchar (255)
	else if facet length is set and <= 255 then char
	else if facet maxLength set and <= 255 then varchar
	else if maxLength is set and <= 65545 then text
	else if maxlength is set and <= 16777215 then mediumtext
	else if maxlength is set and <= 429496295 then longtext
xs:normalizedString	*Same as* xs:string
xs:token	*Same as* xs:string
xs:Name	*Same as* xs:string
xs:NCName	*Same as* xs:string
xs:anyURI	*Same as* xs:string
xs:QName	*Same as* xs:string
xs:NOTATION	*Same as* xs:string
xs:boolean	tinyint(1)
xs:float	float
xs:double	double
xs:decimal	decimal
xs:duration	timestamp
xs:dateTime	datetime
xs:time	time
xs:date	date
xs:gYearMonth	timestamp(4)
xs:gYear	year(4)
xs:gMonthDay	timestamp(8); constraints to check month, day
xs:gDay	timestamp(8); constraints to check day
xs:gMonth	timestamp(8); constraints to check month
xs:hexBinary	If no facets then blob (255)
	else if facet length is set and <= 255

	then blob
	else if facet maxLength is set and <= 255 then tinyblob
	else if maxlength is set and <= 65545 then blob
	else if maxlength is set and <= 16777215 then mediumblob
	else if maxlength is set and <= 429496295 then longblob
xs:base64Binary	*Same as* xs:hexBinary
xs:integer	Integer
xs:int	int
xs:negativeInteger	Integer (constrained to {...,-2,-1})
xs:positiveInteger	Integer (constrained to {1,2,...})
xs:nonNegativeInteger	Integer (constrained to {0,1,2,...})
xs:nonPositiveInteger	Integer (constrained to {...,-2,-1,0})
xs:unsignedInt	Int (additional constraints)
xs:short	Smallint
xs:unsignedShort	Smallint (additional constraints)
xs:long	Bigint
xs:unsignedLong	Bigint (additional constraints)
xs:byte	Tinyint
xs:unsignedByte	Tinyint (additional constraints)

4.4 Oracle

Oracle データベース (DB) が XML スキーマから生成される場合、Oracle データ型は下のテーブルの通り XML スキーマ データ型に変換されます。

XML スキーマデータ型	Oracle データ型
xs:ID	ROWID
xs:string	If no facets
	if UNICODE then NVARCHAR2 (255)
	else VARCHAR2 (255)
	else if UNICODE
	(Size = either length or maxLength)
	If Size <= 2000 then NCHAR
	if Size <= 4000 then NVARHCAR2
	if Size <= 4 Gigabytes then NCLOB
	else if NON-UNICODE
	(Size = either length or maxLength)
	If Size <= 2000 then CHAR
	if Size <= 4000 then VARCHAR2
	if Size <= 4 Gigabytes then CLOB
xs:normalizedString	*Same as* xs:string
xs:token	*Same as* xs:string
xs:Name	*Same as* xs:string
xs:NCName	*Same as* xs:string
xs:anyURI	*Same as* xs:string
xs:QName	*Same as* xs:string
xs:NOTATION	*Same as* xs:string
xs:boolean	NUMBER with constraint Boolean
xs:float	FLOAT
xs:double	FLOAT
xs:decimal	NUMBER
xs:duration	TIMESTAMP
xs:dateTime	TIMESTAMP
xs:time	DATE
xs:date	DATE
xs:gYearMonth	INTERVAL YEAR TO MONTH
xs:gYear	DATE
xs:gMonthDay	DATE

xs:gDay	DATE
xs:gMonth	DATE
xs:hexBinary	if no facets then RAW (255)
	(Size = either length or maxLength)
	If Size <= 2000 then RAW (X)
	else Size <= 2 Gigabytes then LONG RAW (X)
	if Size <= 4 Gigabytes then BLOB (X)
xs:base64Binary	BLOB
xs:integer	NUMBER
xs:int	NUMBER
xs:negativeInteger	NUMBER (constrained to {...,-2,-1})
xs:positiveInteger	NUMBER (constrained to {1,2,...})
xs:nonNegativeInteger	NUMBER (constrained to {0,1,2,...})
xs:nonPositiveInteger	NUMBER (constrained to {...,-2,-1,0})
xs:unsignedInt	NUMBER (additional constraints)
xs:short	NUMBER
xs:unsignedShort	NUMBER (additional constraints)
xs:long	NUMBER
xs:unsignedLong	NUMBER (additional constraints)
xs:byte	BLOB
xs:unsignedByte	BLOB (additional constraints)

5 技術データ

このセクションは、ソフトウェアの技術面に関する役に立つ背景情報を含んでいます。以下のように整理されています：

- OS とメモリ要件
- Altova XML バリデーター
- Altova XSLT とXQuery エンジン
- Unicode のサポート
- インターネットの使用

5.1 OS とメモリ要件

オペレーティングシステム

Altova ソフトウェア アプリケーションは、以下のプラットフォームでご使用いただけます :

- Windows XP/Vista, Windows 7/8/10
- Windows Server 2003/2008/2012/2016

メモリ

ソフトウェアが C++ で書かれているため、Java Runtime Environment をダウンロードする必要はなく、Java ベースのアプリケーションに比べ、通常少ないメモリを必要とします。しかしながら、各ドキュメントは完全に解析するため、また、ビューと編集の速度を向上するためにメモリにダウンロードされます。メモリの要件は、ドキュメントのサイズを増やします。

メモリ要件は、制限のない「元に戻す」履歴により影響を受けます。大きなドキュメントの大きなセクションの切り取り、貼り付け操作を繰り返し行うと、使用できるメモリがすぐに消費されます。

5.2 Altova XML バリデーター

XML ドキュメントを開くと、アプリケーションは、内蔵のXML バリデーターを使用して、指定されている場合、スキーマに対して整形式をチェックし、ツリーとインフォセットを作成します。XML バリデーターは、ドキュメントを編集する際にインテリジェントな編集ヘルプを提供し、発生する検証エラーを表示するために使用されます。

内蔵のXML バリデーターは、W3C のXML スキーマ 1.0 と 1.1 仕様の最終勧告を実装しています。 .New developments recommended by the W3C XML スキーマ作業グループにより勧告される新しい項目は、XML バリデーターに継続的に組み込まれるため、Altova 製品は最高水準の開発環境を届けることができます。

5.3 Altova XSLT と XQuery エンジン

Altova 製品は、Altova XSLT 1.0、2.0、および 3.0 エンジンと Altova XQuery 1.0 と 3.1 エンジンを使用しています。各エンジンのためのドキュメントと実装に固有の振る舞いに関しては、製品で使用されるエンジンの各ドキュメントの付属書 (エンジン情報) で確認することができます。

メモ: Altova MapForce は、XSLT 1.0、2.0 および XQuery 1.0 エンジンを使用したコードを生成します。

5.4 Unicode のサポート

Altova XML 製品は、Unicode を完全にサポートします。XML ドキュメントを編集するには、ドキュメント内で使用されているUnicode 文字をサポートするフォントが必要です。

フォントの多くは、Unicode 範囲全体の特定のサブセットを含む場合があり、このため、通常は対応する表記システムをターゲットとします。テキストの一部が、文字化けして表示された場合、理由としては、選択されたフォントが必要とする字形を含まない場合があげられます。ですから、特に、異なる言語、または、異なる言語システムのXML ドキュメントを編集する場合、範囲全体をカバーするフォントを使用することが役にたちます。典型的なUnicode フォントは、Windows PC の Arial Unicode MS で確認することができます。

アプリケーションフォルダーの /Examples フォルダー内で、異なる言語システムで表記された次の文章を含む UnicodeUTF-8.html というXHTML ファイルを確認してください：

- *When the world wants to talk, it speaks Unicode*
- *Wenn die Welt miteinander spricht, spricht sie Unicode*
- 世界的に話すなら、Unicode です。)

XHTML ファイルを開くと、Unicode の可能性を確認することができ、使用中のPC の使用することのできるフォントによりサポートされている表記システムが表示されます。

5.5 インターネットの使用

Altova アプリケーションは、次の状況でインターネット接続を開始します :

- 登録ダイアログ (**「ヘルプ| ソフトウェアのライセンス認証」**)内の 「評価キーコードをリクエスト」をクリックした場合、登録ダイアログボックス内の3つのフィールドが通常のhttp (ポート 80)接続を使用し、サーバーに転送され、無料の評価キーが顧客に通常のSMTP 電子メールを使用して送り返されます。
- Altova 製品の一部では、インターネットからファイルを開くことができます (**「ファイル| 開く| URL に切り替える」**)。この場合、ドキュメントは、次のプロトコルメソッドと接続の1つを使用して取得されます :HTTP (通常、ポート 80)、FTP (通常、ポート 20/21)、HTTPS (通常、ポート 443)。HTTP サーバーをポート 8080 で作動することもできます (URL ダイアログ内で、サーバー名とコロンの後にポートを指定します)。
- XML スキーマ または、DTD を参照するXML ドキュメント、と、URL により指定されているドキュメントを開くと、参照されている スキーマドキュメントは、HTTP 接続 (ポート 80)または URL により指定されている他のプロトコール (上のポイント 2参照)により抽出されます。XML ファイルが検証されている場合、スキーマドキュメントも抽出されます (オプションダイアログのファイルタブ内の (**「ツール| オプション」**))。アプリケーションに命令している場合、ドキュメントが開かれると検証が自動的に行われる場合もあります。
- WSDL とSOAP を使用するAltova アプリケーションでは、Web サービスを使用する接続は、WSDL ドキュメントにより定義されています。
- XMLSpy 内で、**電子メールで送信」** コマンドを使用する場合、 (**「ファイル| 電子メールで送信」**) 現在選択されている範囲、または、ファイルは、ユーザーのマシンにインストールされているMAPI コンプライアント電子メールプログラムにより送信されます。
- ソフトウェアの荒いセンス認証 とLiveUpdate の一部として、Altova ソフトウェア使用許諾書内で更に詳しい説明を確認することができます。

6 ライセンス情報

ライセンス情報

- ソフトウェアの配布 に関する情報
- ソフトウェアのアクティベーションとライセンスの計測 に関する情報
- このソフトウェアに関連する知的所有権 に関する情報
- このソフトウェア製品の使用に関するエンドユーザー使用許諾契約書

本製品を使用する前に、上記の情報をよくお読みください。ソフトウェアのインストール時に上記のすべての条件に同意したとみなされ、お客様は上記の条件に拘束されることを同意したとみなされます。

6.1 電子的なソフトウェアの配布

この製品は電子的なソフトウェアの配布により利用することが可能で、この配布方法により、以下のユニークなメリットがあります：

- 購入を決定する前に、無料でソフトウェアを試用することができます。
- ソフトウェアの購入を決定した際には、Altova Web サイトにて注文を行います。すぐにライセンス登録された製品の使用を開始することができます。
- オンラインにて注文を行うと、常に最新のソフトウェアをご利用いただけます。
- 製品パッケージには包括的なヘルプシステムが画面上に表示されます。最新バージョンのユーザーマニュアルは www.altova.com 上にあり、(i) HTML フォーマットによる閲覧、ならびに (ii) PDF フォーマットのダウンロードと印刷に対応しております。

30日間の評価期間
この製品をダウンロードした後は、最大で30日の間無料で製品の評価を行うことができます。20日間を超えた頃から、製品がライセンス登録されていないことがソフトウェアにより表示されます。このメッセージはアプリケーションが起動されるたびに表示され、30日間を超えてプログラムを使用するには、キーコードとして送られるAltova ソフトウェア使用許諾契約書 を購入し、ソフトウェアアクティベーションダイアログにて入力し、製品をアンロックする必要があります。ライセンスの購入はAltova Web サイトのオンラインショップにてお求めいただけます。

組織内でソフトウェアの評価を行う
評価版のソフトウェアを組織内のネットワークにて配布したい場合、またはインターネットに接続されていないコンピューターにてソフトウェアを使用する場合、どのような状態でも改変されていないことを条件に、セットアッププログラムだけの配布を行うことが可能です。ソフトウェアインストーラーへアクセスした人は、例外なく30日間の評価ライセンスキーコードをリクエストして、試用期間が経過した後は、製品を使い続けるためにライセンスの購入を行う必要があります。

詳細については、このセクション最後にあるAltova ソフトウェアライセンス使用許諾契約書 を参照ください。

6.2 ソフトウェアのアクティベーションとライセンスの計測

Altova のソフトウェアアクティベーションの一部として、ソフトウェアにより内部ネットワークまたはインターネットへの接続を行い、インストール時、登録時、Altova により使用されるライセンスサーバーの更新やライセンスの正当性を検証することで、ソフトウェアの不正な使用を防ぎ、顧客サービスを向上するため、ライセンスに関する情報を送信することがあります。アクティベーションにより、オペレーティングシステムや IP アドレス、日付/時刻、ソフトウェアのバージョン、コンピュータ名などのライセンスに関する情報が、お使いのコンピューターとAltova ライセンスサーバー間にてやり取りされます。

お使いのAltova 製品にはライセンス計測モジュールが内蔵されており、エンドユーザー使用許諾契約書の意図しない違反を防ぎます。お使いの製品はシングルユーザーまたはマルチユーザーとしてインストールされており、ライセンス計測モジュールにより、ライセンスされている数を超えたユーザーが同時に製品を使用することが無いことが保証されます。

このライセンス計測技術により、ローカルエリア接続 (LAN) において、別々のコンピューター間で動作しているアプリケーションインスタンス間の通信が行われます。

シングルライセンス

ライセンス計測プロセスの一部としてアプリケーションが起動すると、ソフトウェアにより短いデータグラムがブロードキャストにより送信され、同一のネットワークセグメントにある他のコンピューターにてプログラムが動作していないかのチェックが行われます。応答が無い場合は、アプリケーションの他インスタンスから送信される信号に応えるため、ポートが開かれます。

マルチライセンス

同一のLAN 内にて2つ以上のアプリケーションインスタンスが使用された場合、スタートアップ時に、これらインスタンス間において通信が行われます。これらのインスタンス間にてキーコードのやりとりが行われ、購入された数のライセンスを超えてインスタンスが起動しないように保証することができます。このようなライセンス計測システムはUNIX やデータベース開発ツールにて広く使用されているもので、Altova ユーザーはリーズナブルな価格にて同時使用マルチユーザーライセンスを購入することができます。

弊社はアプリケーションのデザインも行っており、少数の小さなネットワークパケットを送信することで、ネットワークに対する負荷を最小限に抑えております。Altova により使用される2799番 TCP/IP ポートはIANA により公式登録されており(詳細は(IANA Web サイト(http://www.iana.org/) を参照ください)、弊社のライセンス計測モジュールも既にテストされたものです。

ファイヤーウォールを使用している場合、2799番ポートにてAltova 製品が動作しているコンピューター同士が通信しているのに気づかれるかも知れません。その他の手段によりライセンス使用許諾書の内容が守られることを保証できる限り、組織間の異なるグループにおいてこのようなトラフィックをブロックすることは勿論可能です。

お使いのコンピューターがオンラインの場合、ライセンス計測技術とは関係の無い数々の便利な機能を利用することができるようになります。

6.3 知的所有権

Altova ソフトウェアならびにAltova により認められたコピーは、Altova ならびにその供給者が知的所有権を有しています。ソフトウェアの構造や構成、コードはAltova ならびにその供給者にとって価値のある企業秘密かつ機密情報となります。このソフトウェアは米国の著作権法により保護されており、国際条項ならびにソフトウェアが使用されている国の法律が適用されます。Altova はこのソフトウェアに関する特許、企業秘密、商標に関する所有権ならびにこれらに限定されないその他の知的財産権を有しており、Altova の所有権には、このソフトウェアならびにそれに付随する印刷物に抱合された画像、写真、アニメーション、映像、音声、音楽、テキスト、そして「アプレット」が含まれます。著作権侵害に関する届け出は、Altova Web Site に記されているAltova 著作権エージェントへお送りください。

Altova ソフトウェアには国際条項により保護されているサードパーティソフトウェアも含まれており、http://www.altova.com/legal_3rdparty.html にて詳細が記されている著作権法なども含め、しかしこれに限定されないかたちで知的所有権により保護されます。

その他全ての名前または商標は、対応する所有者が有するものです。

6.4 エンドユーザー使用許諾契約書

THIS IS A LEGAL DOCUMENT - - RETAIN FOR YOUR RECORDS

ALTOVA® END USER LICENSE AGREEMENT

Licensor:
Altova GmbH
Rudolfsplatz 13a/ 9
A- 1010 Wien
Austria

Important - Read Carefully. Notice to User:

This End User License Agreement ("Agreement") is a legal document between you and Altova GmbH ("Altova"). It is important that you read this document before using the Altova-provided software ("Software") and any accompanying documentation, including, without limitation printed materials, 'online' files, or electronic documentation ("Documentation"). By clicking the "I accept" and "Next" buttons below, or by installing, or otherwise using the Software, you agree to be bound by the terms of this Agreement as well as the Altova Privacy Policy ("Privacy Policy") including, without limitation, the warranty disclaimers, limitation of liability, data use and termination provisions below, whether or not you decide to purchase the Software. You agree that this agreement is enforceable like any written agreement negotiated and signed by you. If you do not agree, you are not licensed to use the Software, and you must destroy any downloaded copies of the Software in your possession or control. You may print a copy of this Agreement as part of the installation process at the time of acceptance. Alternatively, a copy of this Agreement may be found at http://www.altova.com/eula and a copy of the Privacy Policy may be found at http://www.altova.com/privacy.

1. SOFTWARE LICENSE

(a) License Grant.

(i) Upon your acceptance of this Agreement Altova grants you a non-exclusive, non-transferable (except as provided below), limited license, without the right to grant sublicenses, to install and use a copy of the Software on one compatible personal computer or workstation in the same local area network (LAN) up to the Permitted Number of computers. Subject to the limitations set forth in Section 1(c), you may install and use a copy of the Software on more than one of your compatible personal computers or workstations if you have purchased a Named-User license. Subject to the limitations set forth in Sections 1(d) and 1(e), users may use the software concurrently on a network. The Permitted Number of computers and/ or users and the type of license, e.g. Installed, Named-Users, and Concurrent-User, shall be determined and specified at such time as you elect to purchase the Software. Installed user licenses are intended to be fixed and not concurrent. In other words, you cannot uninstall the Software on one machine in order to reinstall that license to a different machine and then uninstall and reinstall back to the original machine. Installations should be static. Notwithstanding the foregoing, permanent uninstallations and redeployments are acceptable in limited circumstances such as if an employee leaves the company or the machine is permanently decommissioned. During the evaluation period, hereinafter defined, only a single user may install and use the software on one (1) personal computer or workstation. If you have licensed the Software as part of a suite of Altova software products (collectively, the "Suite") and have not installed each product individually, then the Agreement governs your use of all of the software included in the Suite.

(ii) If you have licensed SchemaAgent, then the terms and conditions of this Agreement apply to your use of the SchemaAgent server software ("SchemaAgent Server") included therein, as applicable, and you are licensed to use SchemaAgent Server solely in connection with your use of Altova Software and solely for the purposes described in the accompanying documentation.

(iii) If you have licensed Software that enables users to generate source code, your license to install and use a copy of the Software as provided herein permits you to generate

source code based on (i) Altova Library modules that are included in the Software (such generated code hereinafter referred to as the "Restricted Source Code") and (ii) schemas or mappings that you create or provide (such code as may be generated from your schema or mapping source materials hereinafter referred to as the "Unrestricted Source Code"). In addition to the rights granted herein, Altova grants you a non-exclusive, non-transferable, limited license to compile the complete generated code (comprised of the combination of the Restricted Source Code and the Unrestricted Source Code) into executable object code form, and to use, copy, distribute or license that executable. You may not distribute or redistribute, sublicense, sell, or transfer the Restricted Source Code to a third-party in the un-compiled form unless said third-party already has a license to the Restricted Source Code through their separate agreement with Altova. Notwithstanding anything to the contrary herein, you may not distribute, incorporate or combine with other software, or otherwise use the Altova Library modules or Restricted Source Code, or any Altova intellectual property embodied in or associated with the Altova Library modules or Restricted Source Code, in any manner that would subject the Restricted Source Code to the terms of a copyleft, free software or open source license that would require the Restricted Source Code or Altova Library modules source code to be disclosed in source code form. Notwithstanding anything to the contrary herein, you may not use the Software to develop and distribute other software programs that directly compete with any Altova software or service without prior written permission. Altova reserves all other rights in and to the Software. With respect to the feature(s) of UModel that permit reverse-engineering of your own source code or other source code that you have lawfully obtained, such use by you does not constitute a violation of this Agreement. Except as otherwise expressly permitted in Section 1(j) reverse engineering of the Software is strictly prohibited as further detailed therein.

(iv) In the event Restricted Source Code is incorporated into executable object code form, you will include the following statement in (1) introductory splash screens, or if none, within one or more screens readily accessible by the end-user, and (2) in the electronic and/or hard copy documentation: "Portions of this program were developed using Altova® [name of Altova Software, e.g. MapForce® 2016] and includes libraries owned by Altova GmbH, Copyright © 2007-2016 Altova GmbH (www.altova.com)."

(b) Server Use for Installation and Use of SchemaAgent. You may install one (1) copy of the Software on a computer file server within your internal network solely for the purpose of downloading and installing the Software onto other computers within your internal network up to the Permitted Number of computers in a commercial environment only. If you have licensed SchemaAgent, then you may install SchemaAgent Server on any server computer or workstation and use it in connection with your Software. No other network use is permitted, including without limitation using the Software either directly or through commands, data or instructions from or to a computer not part of your internal network, for Internet or Web-hosting services or by any user not licensed to use this copy of the Software through a valid license from Altova.

(c) Named-Use. If you have licensed the "Named-User" version of the software, you may install the Software on up to five (5) compatible personal computers or workstations of which you are the primary user thereby allowing you to switch from one computer to the other as necessary provided that only one (1) instance of the Software will be used by you as the Named-User at any given time. If you have purchased multiple Named-User licenses, each individual Named-User will receive a separate license key code.

(d) Concurrent Use in Same Local Area Network (LAN). If you have licensed a "Concurrent-User" version of the Software, you may install the Software on any compatible computers in a commercial environment only, up to ten (10) times the Permitted Number of users, provided that only the Permitted Number of users actually use the Software at the same time and further provided that the computers on which the Software is installed are on the same local area network (LAN). The Permitted Number of concurrent users shall be delineated at such time as you elect to purchase the Software licenses. Each separate local area network (LAN) requires its own set of separate Concurrent User Licenses for those wishing to use the Concurrent User versions of the Software in more than one location or on more than one network, all subject to the above Permitted Number limitations and based on the number of users using the Software. If a computer is not on the same local area network (LAN), then a locally installed user license or a license dedicated to concurrent use in a virtual environment is required.

(e) Concurrent Use in Virtual Environment. If you have purchased Concurrent-User Licenses, you may install a copy of the Software on a single host terminal server (Microsoft Terminal Server or Citrix Metaframe), application virtualization server (Microsoft App-V, Citrix XenApp, or VMWare ThinApp) or virtual machine environment within your internal network for the sole and exclusive purpose of permitting individual users within your organization to access and use the Software through a terminal server, application virtualization session, or virtual machine environment from another computer provided that the total number of users that access or use the Software concurrently at any given point in time on such network, virtual machine or terminal server does not exceed the Permitted Number; and provided that the total number of users authorized to use the Software through the terminal server, application virtualization session, or virtual machine environment does not exceed ten (10) times the Permitted Number of users. Key codes for concurrent users cannot be deployed to more than one host terminal server, application virtualization server or virtual machine environment. You must deploy a reliable and accurate means of preventing users from exceeding the Permitted Number of concurrent users. Altova makes no warranties or representations about the performance of Altova software in a terminal server, application virtualization session, or virtual machine environment and the foregoing are expressly excluded from the limited warranty in Section 5 hereof. Technical support is not available with respect to issues arising from use in such environments.

(f) Backup and Archival Copies. You may make one (1) backup and one (1) archival copy of the Software, provided your backup and archival copies are not installed or used on any computer and further provided that all such copies shall bear the original and unmodified copyright, patent and other intellectual property markings that appear on or in the Software. You may not transfer the rights to a backup or archival copy unless you transfer all rights in the Software as provided under Section 3.

(g) Key Codes, Upgrades and Updates. Prior to your purchase and as part of the registration for the thirty (30) day evaluation period, as applicable, you will receive an evaluation key code. You will receive a purchase key code when you elect to purchase the Software from either Altova GmbH or an authorized reseller. The purchase key code will enable you to activate the Software beyond the initial evaluation period. You may not re-license, reproduce or distribute any key code except with the express written permission of Altova. If the Software that you have licensed is an upgrade or an update, then the latest update or upgrade that you download and install replaces all or part of the Software previously licensed. The update or upgrade and the associated license keys does not constitute the granting of a second license to the Software in that you may not use the upgrade or updated copy in addition to the copy of the Software that it is replacing and whose license has terminated.

(h) Title. Title to the Software is not transferred to you. Ownership of all copies of the Software and of copies made by you is vested in Altova, subject to the rights of use granted to you in this Agreement. As between you and Altova, documents, files, stylesheets, generated program code (including the Unrestricted Source Code) and schemas that are authored or created by you via your utilization of the Software, in accordance with its Documentation and the terms of this Agreement, are your property unless they are created using Evaluation Software, as defined in Section 4 of this Agreement, in which case you have only a limited license to use any output that contains generated program code (including Unrestricted Source Code) such as Java, C++, C#, VB.NET or XSLT and associated project files and build scripts, as well as generated XML, XML Schemas, documentation, UML diagrams, and database structures only for the thirty (30) day evaluation period.

(i) Reverse Engineering. Except and to the limited extent as may be otherwise specifically provided by applicable law in the European Union, you may not reverse engineer, decompile, disassemble or otherwise attempt to discover the source code, underlying ideas, underlying user interface techniques or algorithms of the Software by any means whatsoever, directly or indirectly, or disclose any of the foregoing, except to the extent you may be expressly permitted to decompile under applicable law in the European Union, if it is essential to do so in order to achieve operability of the Software with another software program, and you have first requested Altova to provide the information necessary to achieve such operability and Altova has not made such information available. Altova has the right to impose reasonable conditions and to request a reasonable fee before providing such information. Any information supplied by Altova or obtained by you, as permitted hereunder, may only be used by you for the purpose described herein and may not be disclosed to any third party or used to create any software which is

substantially similar to the expression of the Software. Requests for information from users in the European Union with respect to the above should be directed to the Altova Customer Support Department.

(j) Other Restrictions. You may not loan, rent, lease, sublicense, distribute or otherwise transfer all or any portion of the Software to third parties except to the limited extent set forth in Section 3 or as otherwise expressly provided. You may not copy the Software except as expressly set forth above, and any copies that you are permitted to make pursuant to this Agreement must contain the same copyright, patent and other intellectual property markings that appear on or in the Software. You may not modify, adapt or translate the Software. You may not, directly or indirectly, encumber or suffer to exist any lien or security interest on the Software; knowingly take any action that would cause the Software to be placed in the public domain; or use the Software in any computer environment not specified in this Agreement. You may not permit any use of or access to the Software by any third party in connection with a commercial service offering, such as for a cloud-based or web-based SaaS offering.

You will comply with applicable law and Altova's instructions regarding the use of the Software. You agree to notify your employees and agents who may have access to the Software of the restrictions contained in this Agreement and to ensure their compliance with these restrictions.

(k) NO GUARANTEE. THE SOFTWARE IS NEITHER GUARANTEED NOR WARRANTED TO BE ERROR-FREE NOR SHALL ANY LIABILITY BE ASSUMED BY ALTOVA IN THIS RESPECT. NOTWITHSTANDING ANY SUPPORT FOR ANY TECHNICAL STANDARD, THE SOFTWARE IS NOT INTENDED FOR USE IN OR IN CONNECTION WITH, WITHOUT LIMITATION, THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT NAVIGATION, COMMUNICATION SYSTEMS, AIR TRAFFIC CONTROL EQUIPMENT, MEDICAL DEVICES OR LIFE SUPPORT SYSTEMS, MEDICAL OR HEALTH CARE APPLICATIONS, OR OTHER APPLICATIONS WHERE THE FAILURE OF THE SOFTWARE OR ERRORS IN DATA PROCESSING COULD LEAD TO DEATH, PERSONAL INJURY OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE. YOU AGREE THAT YOU ARE SOLELY RESPONSIBLE FOR THE ACCURACY AND ADEQUACY OF THE SOFTWARE AND ANY DATA GENERATED OR PROCESSED BY THE SOFTWARE FOR YOUR INTENDED USE AND YOU WILL DEFEND, INDEMNIFY AND HOLD ALTOVA, ITS OFFICERS AND EMPLOYEES HARMLESS FROM ANY THIRD PARTY CLAIMS, DEMANDS, OR SUITS THAT ARE BASED UPON THE ACCURACY AND ADEQUACY OF THE SOFTWARE IN YOUR USE OR ANY DATA GENERATED BY THE SOFTWARE IN YOUR USE.

2. INTELLECTUAL PROPERTY RIGHTS

You acknowledge that the Software and any copies that you are authorized by Altova to make are the intellectual property of and are owned by Altova and its suppliers. The structure, organization and code of the Software are the valuable trade secrets and confidential information of Altova and its suppliers. The Software is protected by copyright, including without limitation by United States Copyright Law, international treaty provisions and applicable laws in the country in which it is being used. You acknowledge that Altova retains the ownership of all patents, copyrights, trade secrets, trademarks and other intellectual property rights pertaining to the Software, and that Altova's ownership rights extend to any images, photographs, animations, videos, audio, music, text and "applets" incorporated into the Software and all accompanying printed materials. You will take no actions which adversely affect Altova's intellectual property rights in the Software. Trademarks shall be used in accordance with accepted trademark practice, including identification of trademark owners' names. Trademarks may only be used to identify printed output produced by the Software, and such use of any trademark does not give you any right of ownership in that trademark. Altova®, XMLSpy®, Authentic®, StyleVision®, MapForce®, UModel®, DatabaseSpy®, DiffDog®, SchemaAgent®, SemanticWorks®, MissionKit®, Markup Your Mind®, Nanonull™, RaptorXML™, RaptorXML Server™, RaptorXML +XBRL Server™, Powered By RaptorXML™, FlowForce Server™, StyleVision Server™, and MapForce Server™ are trademarks of Altova GmbH. (pending or registered in numerous countries). Unicode and the Unicode Logo are trademarks of Unicode, Inc. Windows, Windows XP, Windows Vista, Windows 7, and Windows 8 are trademarks of Microsoft. W3C, CSS, DOM, MathML, RDF, XHTML, XML and XSL are trademarks (registered in numerous countries) of the World Wide Web Consortium (W3C); marks of the W3C

are registered and held by its host institutions, MIT, INRIA and Keio. Except as expressly stated above, this Agreement does not grant you any intellectual property rights in the Software. Notifications of claimed copyright infringement should be sent to Altova's copyright agent as further provided on the Altova Web Site.

3. LIMITED TRANSFER RIGHTS

Notwithstanding the foregoing, you may transfer all your rights to use the Software to another person or legal entity provided that: (a) you also transfer this Agreement, the Software and all other software or hardware bundled or pre-installed with the Software, including all copies, updates and prior versions, and all copies of font software converted into other formats, to such person or entity; (b) you retain no copies, including backups and copies stored on a computer; (c) the receiving party secures a personalized key code from Altova; and (d) the receiving party accepts the terms and conditions of this Agreement and any other terms and conditions upon which you legally purchased a license to the Software. Notwithstanding the foregoing, you may not transfer education, pre-release, or not-for-resale copies of the Software.

4. PRE-RELEASE AND EVALUATION PRODUCT ADDITIONAL TERMS

If the product you have received with this license is pre-commercial release or beta Software ("Pre-release Software"), then this Section applies. In addition, this section applies to all evaluation and/or demonstration copies of Altova software ("Evaluation Software") and continues in effect until you purchase a license. To the extent that any provision in this section is in conflict with any other term or condition in this Agreement, this section shall supersede such other term(s) and condition(s) with respect to the Pre-release and/or Evaluation Software, but only to the extent necessary to resolve the conflict. You acknowledge that the Pre-release Software is a pre-release version, does not represent final product from Altova, and may contain bugs, errors and other problems that could cause system or other failures and data loss. CONSEQUENTLY, THE PRE-RELEASE AND/OR EVALUATION SOFTWARE IS PROVIDED TO YOU **"AS-IS" WITH NO WARRANTIES FOR USE OR PERFORMANCE,** AND ALTOVA DISCLAIMS ANY WARRANTY OR LIABILITY OBLIGATIONS TO YOU OF ANY KIND, WHETHER EXPRESS OR IMPLIED. WHERE LEGALLY LIABILITY CANNOT BE EXCLUDED FOR PRE-RELEASE AND/OR EVALUATION SOFTWARE, BUT IT MAY BE LIMITED, ALTOVA'S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE SUM OF FIFTY DOLLARS (USD $50) IN TOTAL. If the Evaluation Software has a time-out feature, then the software will cease operation after the conclusion of the designated evaluation period. Upon such expiration date, your license will expire unless otherwise extended. Your license to use any output created with the Evaluation Software that contains generated program code (including Unrestricted Source Code) such as Java, C++, C, VB.NET or XSLT and associated project files and build scripts as well as generated XML, XML Schemas, documentation, UML diagrams, and database structures terminates automatically upon the expiration of the designated evaluation period but the license to use such output is revived upon your purchase of a license for the Software that you evaluated and used to create such output. Access to any files created with the Evaluation Software is entirely at your risk. You acknowledge that Altova has not promised or guaranteed to you that Pre-release Software will be announced or made available to anyone in the future, that Altova has no express or implied obligation to you to announce or introduce the Pre-release Software, and that Altova may not introduce a product similar to or compatible with the Pre-release Software. Accordingly, you acknowledge that any research or development that you perform regarding the Pre-release Software or any product associated with the Pre-release Software is done entirely at your own risk. During the term of this Agreement, if requested by Altova, you will provide feedback to Altova regarding testing and use of the Pre-release Software, including error or bug reports. If you have been provided the Pre-release Software pursuant to a separate written agreement, your use of the Software is governed by such agreement. You may not sublicense, lease, loan, rent, distribute or otherwise transfer the Pre-release Software. Upon receipt of a later unreleased version of the Pre-release Software or release by Altova of a publicly released commercial version of the Software, whether as a stand-alone product or as part of a larger product, you agree to return or destroy all earlier Pre-release Software received from Altova and to abide by the terms of the license agreement for any such later versions of the Pre-release Software.

5. LIMITED WARRANTY AND LIMITATION OF LIABILITY

(a) Limited Warranty and Customer Remedies. Altova warrants to the person or entity that first purchases a license for use of the Software pursuant to the terms of this Agreement that (i) the Software will perform substantially in accordance with any accompanying Documentation for a period of ninety (90) days from the date of receipt, and (ii) any support services provided by Altova shall be substantially as described in Section 6 of this agreement. Some states and jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you. To the extent allowed by applicable law, implied warranties on the Software, if any, are limited to ninety (90) days. Altova 's and its suppliers ' entire liability and your exclusive remedy shall be, at Altova 's option, either (i) return of the price paid, if any, or (ii) repair or replacement of the Software that does not meet Altova 's Limited Warranty and which is returned to Altova with a copy of your receipt. This Limited Warranty is void if failure of the Software has resulted from accident, abuse, misapplication, abnormal use, Trojan horse, virus, or any other malicious external code. Any replacement Software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. This limited warranty does not apply to Evaluation and/ or Pre- release Software.

(b) No Other Warranties and Disclaimer. THE FOREGOING LIMITED WARRANTY AND REMEDIES STATE THE SOLE AND EXCLUSIVE REMEDIES FOR ALTOVA OR ITS SUPPLIER 'S BREACH OF WARRANTY. ALTOVA AND ITS SUPPLIERS DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, AND FOR ANY WARRANTY, CONDITION, REPRESENTATION OR TERM TO THE EXTENT WHICH THE SAME CANNOT OR MAY NOT BE EXCLUDED OR LIMITED BY LAW APPLICABLE TO YOU IN YOUR JURISDICTION, ALTOVA AND ITS SUPPLIERS MAKE NO WARRANTIES, CONDITIONS, REPRESENTATIONS OR TERMS, EXPRESS OR IMPLIED, WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE AS TO ANY OTHER MATTERS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, ALTOVA AND ITS SUPPLIERS DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, INFORMATIONAL CONTENT OR ACCURACY, QUIET ENJOYMENT, TITLE AND NON- INFRINGEMENT, WITH REGARD TO THE SOFTWARE, AND THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM STATE/ JURISDICTION TO STATE/ JURISDICTION.

(c) Limitation of Liability. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW EVEN IF A REMEDY FAILS ITS ESSENTIAL PURPOSE, IN NO EVENT SHALL ALTOVA OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF ALTOVA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, ALTOVA 'S ENTIRE LIABILITY UNDER ANY PROVISION OF THIS AGREEMENT SHALL BE LIMITED TO THE AMOUNT ACTUALLY PAID BY YOU FOR THE SOFTWARE PRODUCT. Because some states and jurisdictions do not allow the exclusion or limitation of liability, the above limitation may not apply to you. In such states and jurisdictions, Altova 's liability shall be limited to the greatest extent permitted by law and the limitations or exclusions of warranties and liability contained herein do not prejudice applicable statutory consumer rights of person acquiring goods otherwise than in the course of business. The disclaimer and limited liability above are fundamental to this Agreement between Altova and you.

(d) Infringement Claims. Altova will indemnify and hold you harmless and will defend or settle any claim, suit or proceeding brought against you by a third party that is based upon a claim that the content contained in the Software infringes a copyright or violates an intellectual or proprietary right protected by United States or European Union law ("Claim"),but only to the extent the Claim arises directly out of the use of the Software and subject to the limitations set forth in Section 5 of this Agreement except as otherwise expressly provided. You must notify Altova in writing of any Claim within ten (10) business days after you first receive notice of the Claim, and you shall provide to Altova at no cost such assistance and cooperation as Altova may reasonably request from time to time in connection with the defense of the Claim. Altova shall have sole control over any Claim (including, without limitation, the selection of counsel and the

right to settle on your behalf on any terms Altova deems desirable in the sole exercise of its discretion). You may, at your sole cost, retain separate counsel and participate in the defense or settlement negotiations. Altova shall pay actual damages, costs, and attorney fees awarded against you (or payable by you pursuant to a settlement agreement) in connection with a Claim to the extent such direct damages and costs are not reimbursed to you by insurance or a third party, to an aggregate maximum equal to the purchase price of the Software. If the Software or its use becomes the subject of a Claim or its use is enjoined, or if in the opinion of Altova 's legal counsel the Software is likely to become the subject of a Claim, Altova shall attempt to resolve the Claim by using commercially reasonable efforts to modify the Software or obtain a license to continue using the Software. If in the opinion of Altova 's legal counsel the Claim, the injunction or potential Claim cannot be resolved through reasonable modification or licensing, Altova, at its own election, may terminate this Agreement without penalty, and will refund to you on a pro rata basis any fees paid in advance by you to Altova. THE FOREGOING CONSTITUTES ALTOVA 'S SOLE AND EXCLUSIVE LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT. This indemnity does not apply to situations where the alleged infringement, whether patent or otherwise, is the result of a combination of the Altova software and additional elements supplied by you.

6. SUPPORT AND MAINTENANCE

Altova offers multiple optional "Support & Maintenance Package(s)" ("SMP") for the version of Software product edition that you have licensed, which you may elect to purchase in addition to your Software license. The Support Period, hereinafter defined, covered by such SMP shall be delineated at such time as you elect to purchase a SMP. Your rights with respect to support and maintenance as well as your upgrade eligibility depend on your decision to purchase SMP and the level of SMP that you have purchased:

(a) If you have not purchased SMP, you will receive the Software AS IS and will not receive any maintenance releases or updates. However, Altova, at its option and in its sole discretion on a case by case basis, may decide to offer maintenance releases to you as a courtesy, but these maintenance releases will not include any new features in excess of the feature set at the time of your purchase of the Software. In addition, Altova will provide free technical support to you for thirty (30) days after the date of your purchase (the "Support Period" for the purposes of this paragraph 6(a), and Altova, in its sole discretion on a case by case basis, may also provide free courtesy technical support during your thirty (30) day evaluation period. Technical support is provided via a Web-based support form only, and there is no guaranteed response time.

(b) If you have purchased SMP, then solely for the duration of its delineated Support Period, **you are eligible to receive the version of the Software edition** that you have licensed and all maintenance releases and updates for that edition that are released during your Support Period. For the duration of your SMP 's Support Period, you will also be eligible to receive upgrades to the comparable edition of the next version of the Software that succeeds the Software edition that you have licensed for applicable upgrades released during your Support Period. The specific upgrade edition that you are eligible to receive based on your Support Period is further detailed in the SMP that you have purchased. Software that is introduced as separate product is not included in SMP. Maintenance releases, updates and upgrades may or may not include additional features. In addition, Altova will provide Priority Technical Support to you for the duration of the Support Period. Priority Technical Support is provided via a Web-based support form only and Altova will make commercially reasonable efforts to respond via e-mail to all requests within forty-eight (48) hours during Altova 's business hours (MO-FR, 8am UTC – 10pm UTC, Austrian and US holidays excluded) and to make reasonable efforts to provide work-arounds to errors reported in the Software.

During the Support Period you may also report any Software problem or error to Altova. If Altova determines that a reported reproducible material error in the Software exists and significantly impairs the usability and utility of the Software, Altova agrees to use reasonable commercial efforts to correct or provide a usable work-around solution in an upcoming maintenance release or update, which is made available at certain times at Altova 's sole discretion.

If Altova, in its discretion, requests written verification of an error or malfunction discovered by you or requests supporting example files that exhibit the Software problem, you shall promptly provide such verification or files, by email, telecopy, or overnight mail, setting forth in reasonable

detail the respects in which the Software fails to perform. You shall use reasonable efforts to cooperate in diagnosis or study of errors. Altova may include error corrections in maintenance releases, updates, or new major releases of the Software. Altova is not obligated to fix errors that are immaterial. Immaterial errors are those that do not significantly impact use of the Software as determined by Altova in its sole discretion. Whether or not you have purchased the Support & Maintenance Package, technical support only covers issues or questions resulting directly out of the operation of the Software and Altova will not provide you with generic consultation, assistance, or advice under any circumstances.

Updating Software may require the updating of software not covered by this Agreement before installation. Updates of the operating system and application software not specifically covered by this Agreement are your responsibility and will not be provided by Altova under this Agreement. Altova 's obligations under this Section 6 are contingent upon your proper use of the Software and your compliance with the terms and conditions of this Agreement at all times. Altova shall be under no obligation to provide the above technical support if, in Altova 's opinion, the Software has failed due to the following conditions: (i) damage caused by the relocation of the Software to another location or CPU; (ii) alterations, modifications or attempts to change the Software without Altova 's written approval; (iii) causes external to the Software, such as natural disasters, the failure or fluctuation of electrical power, or computer equipment failure; (iv) your failure to maintain the Software at Altova 's specified release level; or (v) use of the Software with other software without Altova 's prior written approval. It will be your sole responsibility to: (i) comply with all Altova-specified operating and troubleshooting procedures and then notify Altova immediately of Software malfunction and provide Altova with complete information thereof; (ii) provide for the security of your confidential information; (iii) establish and maintain backup systems and procedures necessary to reconstruct lost or altered files, data or programs.

7. SOFTWARE ACTIVATION, UPDATES AND LICENSE METERING

(a) License Metering. The Software includes a built-in license metering module that is designed to assist you with monitoring license compliance in small local area networks (LAN). The metering module attempts to communicate with other machines on your local area network (LAN). You permit Altova to use your internal network for license monitoring for this purpose. This license metering module may be used to assist with your license compliance but should not be the sole method. Should your firewall settings block said communications, you must deploy an accurate means of monitoring usage by the end user and preventing users from using the Software more than the Permitted Number.

(b) License Compliance Monitoring. You are required to utilize a process or tool to ensure that the Permitted Number is not exceeded. Without prejudice or waiver of any potential violations of the Agreement, Altova may provide you with additional compliance tools should you be unable to accurately account for license usage within your organization. If provided with such a tool by Altova, you (a) are required to use it in order to comply with the terms of this Agreement and (b) permit Altova to use your internal network for license monitoring and metering and to generate compliance reports that are communicated to Altova from time to time.

(c) Software Activation. **The Software may use your internal network and Internet connection for the purpose of transmitting license-related data at the time of installation, registration, use, or update to an Altova Master License Server and validating the authenticity of the license-related data in order to protect Altova against unlicensed or illegal use of the Software and to improve customer service. Activation is based on the exchange of license related data between your computer and the Altova Master License Server. You agree that Altova may use these measures and you agree to follow any applicable requirements. You further agree that use of license key codes that are not or were not generated by Altova and lawfully obtained from Altova, or an authorized reseller as part of an effort to activate or use the Software violates Altova 's intellectual property rights as well as the terms of this Agreement. You agree that efforts to circumvent or disable Altova 's copyright protection mechanisms, the license management mechanism, or the Altova Master License Server violate Altova 's intellectual property rights as well as the terms of this Agreement. Altova expressly reserves the rights to seek all available legal and equitable remedies to prevent such actions and to recover lost profits, damages and costs.**

(d) LiveUpdate. Altova provides a new LiveUpdate notification service to you, which is free of charge. Altova may use your internal network and Internet connection for the purpose of transmitting license-related data to an Altova-operated LiveUpdate server to validate your license at appropriate intervals and determine if there is any update available for you.

(e) Use of Data. The terms and conditions of the Privacy Policy are set out in full at http://www.altova.com/privacy and are incorporated by reference into this Agreement. By your acceptance of the terms of this Agreement and/or use of the Software, you authorize the collection, use and disclosure of information collected by Altova for the purposes provided for in this Agreement and/or the Privacy Policy. Altova has the right in its sole discretion to amend this provision of the Agreement and/or Privacy Policy at any time. You are encouraged to review the terms of the Privacy Policy as posted on the Altova Web site from time to time.

(f) Audit Rights. You agree that Altova may audit your use of the Software for compliance with the terms of this Agreement at any time, upon reasonable notice. In the event that such audit reveals any use of the Software by you other than in full compliance with the terms of this Agreement, you shall reimburse Altova for all reasonable expenses related to such audit in addition to any other liabilities you may incur as a result of such non-compliance.

(g) Notice to European Users. Please note that the information as described in paragraph 7(d) above may be transferred outside of the European Economic Area, for purposes of processing, analysis, and review, by Altova, Inc., a company located in Beverly, Massachusetts, U.S.A., or its subsidiaries or Altova's subsidiaries or divisions, or authorized partners, located worldwide. You are advised that the United States uses a sectoral model of privacy protection that relies on a mix of legislation, governmental regulation, and self-regulation. You are further advised that the Council of the European Union has found that this model does not provide "adequate" privacy protections as contemplated by Article 25 of the European Union's Data Directive. (Directive 95/46/EC, 1995 O.J. (L 281) 31). Article 26 of the European Union's Data Directive allows for transfer of personal data from the European Union to a third country if the individual has unambiguously given his consent to the transfer of personal information, regardless of the third country's level of protection. By agreeing to this Agreement, you consent to the transfer of all such information to the United States and the processing of that information as described in this Agreement and the Privacy Policy.

8. TERM AND TERMINATION

This Agreement may be terminated (a) by your giving Altova written notice of termination; (b) by Altova, at its option, giving you written notice of termination if you commit a breach of this Agreement and fail to cure such breach within ten (10) days after notice from Altova; or (c) at the request of an authorized Altova reseller in the event that you fail to make your license payment or other monies due and payable. In addition the Agreement governing your use of a previous version of the Software that you have upgraded or updated is terminated upon your acceptance of the terms and conditions of the Agreement accompanying such upgrade or update. Upon any termination of the Agreement, you must cease all use of the Software that this Agreement governs, destroy all copies then in your possession or control and take such other actions as Altova may reasonably request to ensure that no copies of the Software remain in your possession or control. The terms and conditions set forth in Sections 1(h), 1(i), 1(j), 1(k), 1(l), 2, 5, 7, 9, 10, 11, and 11 survive termination as applicable.

9. RESTRICTED RIGHTS NOTICE AND EXPORT RESTRICTIONS

The Software was developed entirely at private expense and is commercial computer software provided with **RESTRICTED RIGHTS**. Use, duplication or disclosure by the U.S. Government or a U.S. Government contractor or subcontractor is subject to the restrictions set forth in this Agreement and as provided in FAR 12.211 and 12.212 (48 C.F.R. § 12.211 and 12.212) or DFARS 227.7202 (48 C.F.R. § 227-7202) as applicable. Consistent with the above as applicable, Commercial Computer Software and Commercial Computer Documentation licensed to U.S. government end users only as commercial items and only with those rights as are granted to all other end users under the terms and conditions set forth in this Agreement. Manufacturer is Altova GmbH, Rudolfsplatz 13a/9, A-1010 Vienna, Austria/EU. You may not use or otherwise export or re-export the Software or Documentation except as authorized by United States law

and the laws of the jurisdiction in which the Software was obtained. In particular, but without limitation, the Software or Documentation may not be exported or re-exported (i) into (or to a national or resident of) any U.S. embargoed country or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce's Table of Denial Orders. By using the Software, you represent and warrant that you are not located in, under control of, or a national or resident of any such country or on any such list.

10. U.S. GOVERNMENT ENTITIES

Notwithstanding the foregoing, if you are an agency, instrumentality or department of the federal government of the United States, then this Agreement shall be governed in accordance with the laws of the United States of America, and in the absence of applicable federal law, the laws of the Commonwealth of Massachusetts will apply. Further, and notwithstanding anything to the contrary in this Agreement (including but not limited to Section 5 (Indemnification)), all claims, demands, complaints and disputes will be subject to the Contract Disputes Act (41 U.S.C. § § 7101 *et seq.*), the Tucker Act (28 U.S.C. § 1346(a) and § 1491), or the Federal Tort Claims Act (28 U.S.C. § § 1346(b), 2401-2402, 2671-2672, 2674-2680), FAR 1.601(a) and 43.102 (Contract Modifications); FAR 12.302(b), as applicable, or other applicable governing authority. For the avoidance of doubt, if you are an agency, instrumentality, or department of the federal, state or local government of the U.S. or a U.S. public and accredited educational institution, then your indemnification obligations are only applicable to the extent they would not cause you to violate any applicable law (e.g., the Anti-Deficiency Act), and you have any legally required authorization or authorizing statute.

11. THIRD PARTY SOFTWARE

The Software may contain third party software which requires notices and/or additional terms and conditions. Such required third party software notices and/or additional terms and conditions are located at our Website at http://www.altova.com/legal_3rdparty.html and are made a part of and incorporated by reference into this Agreement. By accepting this Agreement, you are also accepting the additional terms and conditions, if any, set forth therein.

12. JURISDICTION, CHOICE OF LAW, AND VENUE

If you are located in the European Union and are using the Software in the European Union and not in the United States, then this Agreement will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht, Wien (Commercial Court, Vienna) in connection with any such dispute or claim.

If you are located in the United States or are using the Software in the United States then this Agreement will be governed by and construed in accordance with the laws of the Commonwealth of Massachusetts, USA (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the federal or state courts of the Commonwealth of Massachusetts and you further agree and expressly consent to the exercise of personal jurisdiction in the federal or state courts of the Commonwealth of Massachusetts in connection with any such dispute or claim.

If you are located outside of the European Union or the United States and are not using the Software in the United States, then this Agreement will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht Wien (Commercial Court, Vienna) in connection with any such dispute or claim. This Agreement will not be governed by the conflict of law rules of any jurisdiction or the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded.

13. TRANSLATIONS

Where Altova has provided you with a foreign translation of the English language version, you agree that the translation is provided for your convenience only and that the English language version will control. If there is any contradiction between the English language version and a translation, then the English language version shall take precedence.

14. GENERAL PROVISIONS

This Agreement contains the entire agreement and understanding of the parties with respect to the subject matter hereof, and supersedes all prior written and oral understandings of the parties with respect to the subject matter hereof. Any notice or other communication given under this Agreement shall be in writing and shall have been properly given by either of us to the other if sent by certified or registered mail, return receipt requested, or by overnight courier to the address shown on Altova 's Web site for Altova and the address shown in Altova 's records for you, or such other address as the parties may designate by notice given in the manner set forth above. This Agreement will bind and inure to the benefit of the parties and our respective heirs, personal and legal representatives, affiliates, successors and permitted assigns. The failure of either of us at any time to require performance of any provision hereof shall in no manner affect such party 's right at a later time to enforce the same or any other term of this Agreement. This Agreement may be amended only by a document in writing signed by both of us. In the event of a breach or threatened breach of this Agreement by either party, the other shall have all applicable equitable as well as legal remedies. Each party is duly authorized and empowered to enter into and perform this Agreement. If, for any reason, any provision of this Agreement is held invalid or otherwise unenforceable, such invalidity or unenforceability shall not affect the remainder of this Agreement, and this Agreement shall continue in full force and effect to the fullest extent allowed by law. The parties knowingly and expressly consent to the foregoing terms and conditions.

Last updated: 2015/ 09/ 03

Index

,248

▪

.docx, 245, 639
.NET,
XMLSpy との統合 ,772
スタンドアロン XMLSpy との違い ,775
と XMLSpy デバッガー ,777
.NET 拡張関数 ,
XSLT と XQuery, 1746
インスタンスメソッド、インスタンスフィールド ,1749
コンストラクター ,1748
データ型変換、 .NET から XPath/ XQuery へ ,1751
データ型変換、XPath/ XQuery から .NET へ ,1750
概要 ,1746
静的メソッド、静的フィールド ,1749
.NET 内の XSLT と XQuery のための拡張機能 ,
.NET 拡張関数を参照する ,1746
.pptx, 245, 639
.xslx, 245, 639

A

AAIDC ペイン ,187
Activating the software, 1197
Active configuration,
for global resources, 1151
ActiveX,
integration at application level, 1607
integration at document level, 1609
integration prerequisites, 1603
ActiveX controls,
adding to the Visual Studio Toolbox, 1605
support, 1247
Add Child command,
in Grid View, 965
ADO,
データ接続インターフェイスとして ,650
接続のセットアップ ,654
ADO.NET,
接続のセットアップ ,659
Altova Scripting Projects, 1208
Altova XML パーサー ,
について ,1773
Altova エンジン ,
Altova 製品内で ,1774
Altova グローバルリソース ,
グローバルリソース ,703
グローバルリソース を参照する ,703
Altova サポート ,122
Altova 拡張関数 ,
チャート関数 (チャート関数を参照),1680
Altova 製品 ,122
Ant,
setting the environment variables for,, 870
API,
documentation, 1261
JAVA, 1561
JAVA Classpath, 1561
overview, 1263
Append,
row (in Authentic View), 1041
Append command,
in Grid View, 958
Application,
ActiveDocument, 1298
AddMacroMenuItem, 1298
AddXSLT_XQParameter, 1298
Application, 1298
ClearMacroMenu, 1299
CurrentProject, 1299
Dialogs, 1300
Documents, 1300
GetDatabaseImportElementList, 1301
GetDatabaseSettings, 1301
GetDatabaseTables, 1301
GetExportSettings, 1302
GetTextImportElementList, 1302
GetTextImportExportSettings, 1303
GetXSLT_XQParameterCount, 1303
GetXSLT_XQParameterName, 1304
GetXSLT_XQParameterXPath, 1304
ImportFromDatabase, 1304
ImportFromSchema, 1305
ImportFromText, 1306

Application,
ImportFromWord, 1306
NewProject, 1307
OnBeforeOpenDocument, 1296
OnBeforeOpenProject, 1296
OnDocumentOpened, 1297
OnProjectOpened, 1297
OpenProject, 1308
Parent, 1308
Quit, 1308
ReloadSettings, 1309
RemoveXSLT_XQParameter, 1309
RunMacro, 1309
ScriptingEnvironment, 1309
ShowApplication, 1310
ShowForm, 1311
URLDelete, 1311
URLMakeDirectory, 1311
WarningNumber, 1312
WarningText, 1312
Application Events, 1225
Apply, 1170
Archive View, 245, 639
とEPUB ファイル ,645
とOOXML ファイル ,641
とZIP ファイル ,644
Arcroles in XBRL, 1118
ASPRJ files, 1208
Assign,
shortcut to a command, 1160
Assigning StyleVision Power Stylesheet to XML file, 1036
ATL,
plug-in sample files, 1251
ATTLIST declaration,
add as child in Grid View, 969
appending in Grid View, 964
convert to in Grid View, 972
inserting in Grid View, 957
Attribute,
add as child in Grid View, 965
appending in Grid View, 959
convert to in Grid View, 971
inserting in Grid View, 952
Attribute preview, 1176
AttributeFormDefault,
settings in Schema Design View, 1001
Authentic menu, 1034
Authentic Scripting,
security settings, 1042
trusted locations, 1042
Authentic View, 461, 464
CDATA セクション ,440
editing data in an XML DB, 1036
editing DB data in, 1035
generating output documents from PXF file, 1042
GUI の概要 ,445
opening new XML file in, 1034
SPS テーブル ,460
switching to, 1091
XML ドキュメントの印刷 ,443
XML ドキュメントを開く,435
XML/ Text として貼付け ,454
エンティティ,440
エンティティの挿入 ,442
コンテキストメニュー ,436,454
ツールバーアイコン ,446
データの入力 ,440
データ入力デバイス ,440
テーブル ,438
テーブル（SPS ならびに CALS/ HTML）,460
テキストのフォーマット,446
ドキュメントの表示 ,448
ノードの削除 ,438
ノードの挿入 ,438
ノードの追加 ,438
ノードの適用 ,438
マークアップタグの表示 ,436
マークアップの表示 ,446,448
メインウィンドウ ,448
重要な機能の使用方法 ,456
属性値の入力 ,442
特殊な文字の入力 ,440
入力ヘルパー ,436
要素のクリア ,438
Authentic View Events, 1225
Authentic View テンプレート, 435
Authentic View における SPS テーブル ,
使用方法 ,460
Authentic View におけるテーブル ,
SPS を使った（静的ならびに動的な）テーブル ,460
Authentic View における静的（SPS）テーブル ,
使用方法 ,460
Authentic View における動的（SPS）テーブル ,
使用方法 ,460
Authentic View にて ,
入力ヘルパー ,450
Authentic View における CALS/ HTML テーブル ,

Authentic Viewにおける CALS/ HTML テーブル ,
使用方法 ,461
編集アイコン ,464
Authentic Viewにおけるテーブル ,
CALS/ HTML テーブルの編集アイコン ,464
CALS/ HTML テーブルを使用する ,461
使用方法 ,460
Authentic XML, 432
Authentic ビュー ,
XML の編集 ,256
Authentic メニュー ,
マークアップの表示 ,446
動的テーブルの編集 ,446
AuthenticDataTransfer,
dropEffect, 1314
getData, 1315
ownDrag, 1315
type, 1315
AuthenticRange,
AppendRow, 1319
Application, 1320
CanPerformAction, 1320
CanPerformActionWith, 1321
Close, 1321
CollapsToBegin, 1321
CollapsToEnd, 1322
Copy, 1322
Cut, 1322
Delete, 1322
DeleteRow, 1323
DuplicateRow, 1323
ExpandTo, 1324
FirstTextPosition, 1324
FirstXMLData, 1325
FirstXMLDataOffset, 1326
GetElementAttributeNames, 1327
GetElementAttributeValue, 1327
GetElementHierarchy, 1328
GetEntityNames, 1329
Goto, 1329
GotoNext, 1330
GotoNextCursorPosition, 1330
GotoPrevious, 1331
GotoPreviousCursorPosition, 1331
HasElementAttribute, 1332
InsertEntity, 1332
InsertRow, 1332
IsCopyEnabled, 1333
IsCutEnabled, 1333
IsDeleteEnabled, 1333
IsEmpty, 1334
IsEqual, 1334
IsFirstRow, 1334
IsInDynamicTable, 1334
IsLastRow, 1335
IsPasteEnabled, 1335
IsTextStateApplied, 1335
LastTextPosition, 1336
LastXMLData, 1336
LastXMLDataOffset, 1337
MoveBegin, 1338
MoveEnd, 1338
MoveRowDown, 1339
MoveRowUp, 1339
Parent, 1339
Paste, 1340
PerformAction, 1340
Select, 1341
SelectNext, 1341
SelectPrevious, 1342
SetElementAttributeValue, 1342
SetFromRange, 1343
Text, 1344
AuthenticView, 1360
Application, 1353
AsXMLString, 1354
DocumentBegin, 1355
DocumentEnd, 1355
Event, 1356
Goto, 1357
IsRedoEnabled, 1358
IsUndoEnabled, 1358
MarkupVisibility, 1358
OnBeforeCopy, 1345
OnBeforeCut, 1346
OnBeforeDelete, 1346
OnBeforeDrop, 1347
OnBeforePaste, 1348
OnDragOver, 1348
OnKeyboardEvent, 1349
OnMouseEvent, 1350
OnSelectionChanged, 1351
Parent, 1358
Print, 1359
Redo, 1359
Selection, 1359
Undo, 1360
WholeDocument, 1361

AuthenticView, 1360
XMLDataRoot, 1361
Auto- complete,
text view enable/ disable, 1175
Auto- completion in SQL scripts, 1050
Auto- Macro setting, 1228
Automatic validation, 1173
Avro,
JSON 内のデータ構造 ,538
とRaptorXML, 533
バイナリ内のデータ構造 ,539
ファイルの型 ,533
概要 ,533
Avro スキーマ ,486
説明 ,535
用語 ,535
Avro バイナリファイル ,539
Avro ビュー ,539

B

Big- endian, 1185
Bookmark, 1196
Bookmark margin, 1094
Bookmarks,
inserting and removing, 923
navigating, 923
Bookmarks in SQL scripts, 1050
Breakpoint,
dialog box, 1032
Browse,
Oracle XML Db, 1064
Browser, 1176
View, 1092
Browser menu, 1096
Browser pane,
in Database Query window, 1047
Browser View, 1096
back, 1096
font size, 1097
forward, 1096
refresh content, 1096
separate window, 1097
stop loading page, 1096
build.xml,
enabling the zip64mode in„ 870

C

C#,
code, 788
code generation settings, 871
integration of XMLSpy, 1613
C++,
code, 788
code generation settings, 871
CALS/ HTML テーブル ,
CALS/ HTML テーブルを使用する ,461
CALS/ HTML テーブルアイコン ,464
Carriage Return キー ,
Enter キーを参照 ,476
Cascade,
Window, 1194
Catalog,
Oasis XML, 977
CDATA,
add as child in Grid View, 966
appending in Grid View, 959
convert to in Grid View, 971
inserting in Grid View, 952
CDATA セクション ,
Authentic View にて挿入 ,456
Chapters, 1196
Character,
position, 1093
Character- Set,
encoding, 1185
Check,
spelling checker, 1135
Class,
JAVA, 1561
Class ID,
in XMLSpy integration, 1615
ClassPath statement, 1561
Code,
built in types, 882
SPL, 872
Code Generator, 788, 999
Code page, 1178
CodeGeneratorDlg,
Application, 1362
CPPSettings_DOMType, 1363
CPPSettings_LibraryType, 1364

CodeGeneratorDlg,
CPPSettings_UseMFC, 1364
CSharpSettings_ProjectType, 1364
OutputPath, 1364
OutputPathDialogAction, 1365
OutputResultDialogAction, 1365
Parent, 1365
ProgrammingLanguage, 1365
PropertySheetDialogAction, 1366
TemplateFileName, 1366
Collapse,
unselected, 1092
Color, 1178, 1183
tab, 1183
table, 1183
COM- API,
documentation, 1261
Command, 1163
add to toolbar/ menu, 1155
context menu, 1163
delete from menu, 1163
reset menu, 1163
Command line, 1202
Commands,
listing in key map, 1197
Comment,
add as child in Grid View, 966
appending in Grid View, 960
convert to in Grid View, 971
inserting in Grid View, 953
Commenting XML text in and out, 924
Comments in SQL scripts, 1050
Comparing directories, 1145
Comparing files, 1143
options, 1147
Configure,
XMLSPY UI, 1248
Configure view,
dialog for Content Model View, 1009
Connecting to SchemaAgent Server, 1016
Content Model,
save diagram, 1003
Content Model View,
configuring, 1009
Context menu,
commands, 1163
for customization, 1167
Convert,
database data to XML, 1068
database schema to XML Schema, 1072
MS Word data to XML, 1072
Oracle XML Db, 1062
schema to DB structure, 1077
text file to XML, 1066
Convert menu, 1066
Convert To command,
in Grid View, 970
Copy,
as XML text, 909
XML as structured text, 909
Copy command, 908
Copy XPath, 912
Copy XPointer, 912
CR&LF, 1170
Create,
DB based on schema, 1077
CSS, 479
ドキュメントアウトライン ,482
プロパティ,482
構文による色分け ,482
自動補完 ,482
情報ウィンドウ ,482
CSS 情報ウィンドウ ,482
CSV file,
import as XML, 1066
Custom dictionary, 1135
CustomCatalog, 977
Customize, 1163
context menu, 1163
Customize context menu, 1167
macros, 1164
menu, 1163
toolbar/ menu commands, 1155
Cut command, 908

D

Database,
create DB based on schema, 1077
editing records of, 1054
export of XML data to, 1082
import data as XML, 1068
import structure as XML Schema, 1072
Oracle XML Db, 1062
Database Query,
Browser pane in DB Query window, 1047

Database Query,
Connecting to DB for query, 1045
creating the query, 1053
Editing results, 1054
Messages pane, 1054
Results of, 1054
Database Query window, 1044
DatabaseConnection,
ADOConnection, 1367
AsAttributes, 1367
CreateMissingTables, 1368
CreateNew, 1368
DatabaseKind, 1368
ExcludeKeys, 1369
File, 1369
IncludeEmptyElements, 1370
NumberDateTimeFormat, 1370
ODBCConnection, 1371
SQLSelect, 1372
TextFieldLen, 1372
Databases,
editing in Authentic View, 1035
DatabaseSpy, 293
3D チャート,301
チャートX-軸 ,298
チャートY-軸 ,300
チャートZ-軸 ,301
チャートグリッド,298,300,301
チャートサイズ ,302
チャートのタイトル ,291
チャートの色 ,296
チャートの背景 ,291
チャートの凡例 ,291
チャートフォント,303
チャートフォントオプション ,303
DB,
Authentic View においてテーブルのナビゲーション ,466
Authentic View にて表示をフィルタリング ,467
Authentic View にて編集 ,470
Authentic Viewにおける編集 ,466
Authentic Viewにおける問い合わせ ,466
DB クエリ内のパラメーター ,467
クエリの作成 ,467
DB XML,
assigning XML Schemas to, for IBM DB2, 1059
managing XML Schemas, for IBM DB2, 1056
db2-fn:sqlquery, 395
db2-fn:xmlcolumn, 395
Debugger,
breakpoints/ tracepoints dialog box, 1032
debug windows, 1033
enable/ disable breakpoint, 1032
enable/ disable tracepoint, 1032
end session, 1031
for SOAP, 1115
insert/ remove breakpoint, 1032
insert/ remove tracepoint, 1032
options for SOAP, 1117
restart XSLT Debugger, 1031
settings, 1033
show curr. exec. nodes, 1031
start XSLT Debugger, 1030
step into, 1031
step out, 1031
step over, 1031
stop XSLT Debugger, 1030
Debugging macros, 1232
Default,
encoding, 1185
menu, 1163
Default editor, 1173
Default view,
setting in Main Window, 1173
defaultOpenContent, 164
Delete,
Application.URLDelete, 1311
command from context menu, 1163
command from toolbar, 1155
icon from toolbar, 1155
row (in Authentic View), 1041
shortcut, 1160
toolbar, 1157
Delete command, 908
DF,
XMLSpy における変換 ,365
Dialogs,
Application, 1373
CodeGeneratorDlg, 1373
DTDSchemaGeneratorDlg, 1374
FileSelectionDlg, 1373
GenerateSampleXMLDlg, 1374
Parent, 1374
SchemaDocumentationDlg, 1374
Dictionary, 1135
adding custom, 1135
modifying existing, 1135
spelling checker, 1135
directories,

directories,
comparing two, 1145
creating with Application.URLMakeDirectory, 1311
Disable,
breakpoint - XSLT debugger, 1032
tracepoint - XSLT debugger, 1032
Disconnect XMLSpy from SchemaAgent, 1017
Display all globals, 1012
Display diagram, 1013
Display Settings, 1183
DocEditEvent (obsolete),
altKey (obsolete), 1503
altLeft (obsolete), 1504
button (obsolete), 1505
cancelBubble (obsolete), 1506
clientX (obsolete), 1507
clientY (obsolete), 1508
ctrlKey (obsolete), 1509
ctrlLeft (obsolete), 1510
dataTransfer (obsolete), 1511
fromElement (obsolete), 1512
keyCode (obsolete), 1512
propertyName (obsolete), 1513
repeat (obsolete), 1513
returnValue (obsolete), 1513
shiftKey (obsolete), 1514
shiftLeft (obsolete), 1515
srcElement (obsolete), 1516
type (obsolete), 1517
DocEditView (obsolete),
ApplyTextState (obsolete), 1523
CurrentSelection (obsolete), 1524
EditClear (obsolete), 1524
EditCopy (obsolete), 1525
EditCut (obsolete), 1525
EditPaste (obsolete), 1526
EditRedo (obsolete), 1526
EditSelectAll (obsolete), 1527
EditUndo (obsolete), 1527
event (obsolete), 1528
GetAllowedElements (obsolete), 1528
GetNextVisible (obsolete), 1531
GetPreviousVisible (obsolete), 1532
IsEditClearEnabled (obsolete), 1533
IsEditCopyEnabled (obsolete), 1533
IsEditCutEnabled (obsolete), 1534
IsEditPasteEnabled (obsolete), 1534
IsEditRedoEnabled (obsolete), 1535
IsEditUndoEnabled (obsolete), 1535
IsRowAppendEnabled (obsolete), 1536
IsRowDeleteEnabled (obsolete), 1536
IsRowDuplicateEnabled (obsolete), 1537
IsRowInsertEnabled (obsolete), 1537
IsRowMoveDOwnEnabled (obsolete), 1538
IsRowMoveUpEnabled (obsolete), 1538
IsTextStateApplied (obsolete), 1539
IsTextStateEnabled (obsolete), 1539
LoadXML (obsolete), 1540
RowAppend (obsolete), 1541
RowDelete (obsolete), 1541
RowDuplicate (obsolete), 1542
RowInsert (obsolete), 1542
RowMoveDown (obsolete), 1543
RowMoveUp (obsolete), 1543
SaveXML (obsolete), 1544
SelectionMoveTabOrder (obsolete), 1545
SelectionSet (obsolete), 1546
XMLRoot (obsolete), 1547
Dockable window, 1194
DOCTYPE declaration,
add as child in Grid View, 968
appending in Grid View, 962
convert to in Grid View, 971
inserting in Grid View, 955
Document, 1387
Application, 1380
AssignDTD, 1380
AssignSchema, 1380
AssignXSL, 1381
AssignXSLFO, 1381
AuthenticView, 1381
browse Oracle XML Db, 1064
Close, 1382
ConvertDTDOrSchema, 1382
CreateChild, 1384
CreateSchemaDiagram, 1385
CurrentViewMode, 1385
DataRoot, 1385
DocEditView, 1386
Encoding, 1386
EndChanges, 1386
ExecuteXQuery, 1387
ExportToDatabase, 1387
ExportToText, 1388
FullName, 1389
GenerateDTDOrSchema, 1390
GenerateProgramCode, 1390
GenerateSampleXML, 1391

Document, 1387
GenerateSchemaDocumentation, 1391
GetExportElementList, 1393
GetPathName, 1393
GridView, 1394
IsModified, 1394
IsValid, 1394
IsWellFormed, 1395
Name, 1396
OnBeforeCloseDocument, 1378
OnBeforeSaveDocument, 1377
OnBeforeValidate, 1378
OnCloseDocument, 1379
OnViewActivation, 1379
Path, 1396
RootElement, 1396
Save, 1397
SaveAs, 1397
Saved, 1397
SaveInString, 1397
SaveToURL, 1398
SetActiveDocument, 1398
SetEncoding, 1398
SetExternalIsValid, 1400
SetPathName, 1400
Spelling checker, 1135
StartChanges, 1400
SwitchViewMode, 1401
Title, 1401
TransformXSL, 1402
TransformXSLFO, 1402
UpdateViews, 1403
UpdateXMLData, 1403
XQuery, 1387
Document Events, 1225
Documentation,
of WSDL files, 1104
of XBRL taxonomies, 1125
of XML Schema files, 1004
Document- level,
examples of integration of XMLSpy, 1613
Documents,
Count, 1404
Item, 1404
NewAuthenticFile, 1404
NewFile, 1405
NewFileFromText, 1405
OpenAuthenticFile, 1405
OpenFile, 1406
OpenURL, 1406
OpenURLDialog, 1407
DTD, 325
assigning to XML document, 989
Attlist declaration in, 957, 964, 969, 972
converting to UML, 995
converting to XML Schema, 992
Element declaration in, 956, 963, 969, 972
Entity declaration in, 957, 964, 969, 972
generate outline XML file from, 996
generating code from, 999
generating from XML document, 991
go to definition in from XML document, 990
go to from XML document, 990
including entities, 990
menu commands related to, 989
Notation declaration in, 957, 964, 969, 972
reference from XML to external DTD, 956
XML スキーマから生成 (Enterprise ならびに Professional エディションのみ) ,327
XML スキーマへの変換 (Enterprise ならびに Professional エディションのみ) ,326
XML ドキュメントの生成 ,326
グリッドビューで編集 (Enterprise ならびに Professional エディションのみ) ,326
テキストビューで編集 ,326
DTD/ Schema menu, 989
DTDs, 1170, 1173
DTDSchemaGeneratorDlg,
Application, 1408
AttributeTypeDefinition, 1408
DTDSchemaFormat, 1408
FrequentElements, 1408
GlobalAttributes, 1409
MaxEnumLength, 1409
MergeAllEqualNamed, 1409
OnlyStringEnums, 1409
OutputPath, 1409
OutputPathDialogAction, 1410
Parent, 1410
ResolveEntities, 1410
TypeDetection, 1410
ValueList, 1410
Duplicate,
row (in Authentic View), 1041

E

Eclipse における XMLSpy パースペクティブ ,787

Eclipse に対応した XMLSpy プラグイン ,
インストール ,779
Eclipse プラットフォーム ,
Eclipse における ,784
XMLSpy デバッガーパースペクティブ ,787
XMLSpy 統合パッケージ ,779
とXMLSpy, 778
Eclipse 内部で XMLSpy を使用する ,778
Edit,
macro button, 1167
Edit menu, 907
Edited with XMLSPY, 1170
Editing database records, 1054
Element,
add as child in Grid View, 965
appending in Grid View, 959
convert to in Grid View, 971
inserting in Grid View, 952
ELEMENT declaration,
add as child in Grid View, 969
appending in Grid View, 963
convert to in Grid View, 972
ELEMENT declaration in DTD,
inserting in Grid View, 956
ElementFormDefault,
settings in Schema Design View, 1001
ElementList,
Count, 1411
Item, 1411
RemoveElement, 1411
ElementListItem,
ElementKind, 1412
FieldCount, 1412
Name, 1412
RecordCount, 1412
E- mail,
sending files with, 903
Empty elements, 1173
Enable breakpoint - XSLT debugger, 1032
Enable tracepoint - XSLT debugger, 1032
Enclose in Element command, 975
Encoding,
default, 1185
of files, 897
End,
debugger session, 1031
End- of- line markers, 1094
Enhanced Grid View, 1090
Enter キー ,
その効果 ,476
ENTITY declaration,
add as child in Grid View, 969
appending in Grid View, 964
convert to in Grid View, 972
inserting in Grid View, 957
Entry helpers,
toggling display on and off, 1195
updating, 983
Entry- Helper, 1195
Enumerations,
in XMLSpyControl, 1664
SPYAttributeTypeDefinition, 1548
SPYAuthenticActions, 1548
SPYAuthenticDocumentPosition, 1548
SpyAuthenticElementActions, 1549
SPYAuthenticElementKind, 1549
SPYAuthenticMarkupVisibility, 1549
SPYDatabaseKind, 1550
SPYDialogAction, 1550
SPYDOMType, 1550
SPYDTDSchemaFormat, 1551
SPYEncodingByteOrder, 1551
SPYExportNamespace, 1551
SPYFrequentElements, 1551
SPYKeyEvent, 1552
SPYLibType, 1552
SPYLoading, 1552
SPYMouseEvent, 1553
SPYNumberDateTimeFormat, 1553
SPYProgrammingLanguage, 1554
SPYProjectItemTypes, 1554
SPYProjectType, 1554
SPYSampleXMLGenerationOptimization, 1555
SPYSampleXMLGenerationSchemaOrDTDAssignment, 1555
SPYSchemaDefKind, 1556
SPYSchemaDocumentationFormat, 1556
SPYTextDelimiters, 1557
SPYTextEnclosing, 1557
SPYTypeDetection, 1557
SPYURLTapes, 1557
SPYViewModes, 1558
SPYVirtualKeyMask, 1559
SPYXMLDataKind, 1559
Environment variables,
ANT_OPS, 870
EPUB ファイル ,645
Evaluation key,

Evaluation key,
for your Altova software, 1197
Event, 1296, 1297, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1377, 1378, 1379, 1431, 1432, 1433
Event handlers,
in Scripting Project, 1225
overview, 1213
Events, 1225, 1268
and event handlers, 1215
Excel 2007, 245, 639
Expand,
fully, 1092
Explorer, 1173
Exporting XML data to database, 1082
Exporting XML data to text files, 1079
ExportSettings,
CreateKeys, 1413
ElementList, 1413
EntitiesToText, 1413
ExportAllElements, 1414
FromAttributes, 1414
FromSingleSubElements, 1414
FromTextValues, 1414
IndependentPrimaryKey, 1414
Namespace, 1415
SubLevelLimit, 1415
Extended validation,
in SchemaAgent, 1017
External applications,
opening files in, 1158
External ID declaration,
add as child in Grid View, 969
appending in Grid View, 963
convert to in Grid View, 971
inserting in Grid View, 956
External parsed entites, 1173
External XSL processor, 1186

F

Favorites, 1196
File, 1170
closing, 898
creating new, 889
default encoding, 1185
encoding, 897
opening, 893
opening options, 1170
printing options, 904
saving, 898
sending by e-mail, 903
tab, 1170
File extensions,
customizing, 977
File menu, 889
File types, 1173
Files,
adding to source control, 934
comparing two, 1143
comparison options, 1147
most recently used, 906
FileSelectionDlg,
Application, 1416
DialogAction, 1416
FullName, 1416
Parent, 1416
Find,
and replace text in document, 920
text in document, 917
Find in Files command, 921
Firebird,
ODBC を使用しての接続 ,672
Folding margin, 1094
Font, 1179
schema, 1179
Schema Documentation, 1179
Font size,
in Browser View, 1097
FOP,
フォント,261
Form Object Palette, 1210
Form Object properties, 1220
Forms,
and built-in commands, 1234
and event handling, 1222
and Form Objects, 1220
creating new, 1219
in Scripting Projects, 1219
invocation of, 1215
naming, 1219
overview, 1213
properties of, 1219
setting tab sequence of objects, 1220

G

Generate,
code from schema, 788
DB structure based on schema, 1077
Generate Sample XML, 1548, 1555
GenerateSampleXMLDlg,
Application, 1427
FillWithSampleData, 1428
NonMandatoryAttributes, 1429
NonMandatoryElements, 1429
Parent, 1429
RepeatCount, 1430
TakeFirstChoice, 1430
Global,
settings, 1170
Global attribute groups, 164
Global declarations, 1217
overview, 1213
Global objects,
in SPL, 875
Global resources,
active configuration for, 1151
defining, 1150
toolbar activation, 1157
Global scripting project,
of XMLSpy, 1208
Go to File, 1093
Go to line/ char, 1093
Grammar, 1173
Gray bar, 1092
Grid fonts, 1178
Grid view, 1090, 1092
editing, 974, 975
switching to Table View, 972
Grid View Events, 1225
GridView,
CurrentFocus, 1433
Deselect, 1433
IsVisible, 1433
OnBeforeDrag, 1431
OnBeforeDrop, 1431
OnBeforeStartEditing, 1432
OnEditingFinished, 1432
OnFocusChanged, 1433
Select, 1434
SetFocus, 1434
GUI 詳細 ,97

H

Help,
contents, 1196
key map, 1197
Help menu, 1196
Help system, 1196
Hide, 1194, 1195
Hide markup (in Authentic View), 1040
Hotkey, 1160
HTML, 479
integration of XMLSpy, 1617
HTML example,
of XMLSpyControl integration, 1615, 1616, 1617
HTML output,
generating in Authentic View from PXF file, 1042
HTML ドキュメント,
情報ウィンドウ ,480
編集 ,480
HTML 情報 ウィンドウ ,480

I

IBM DB2,
assigning XML Schemas to XML file, 1059
managing XML Schemas, 1056
ODBC を使用しての接続 ,675
schema management and assignment, 1056
IBM DB2 for i,
ODBC を使用しての接続 ,679
IBM Informix,
JDBC を使用しての接続 ,682
Icon,
add to toolbar/ menu, 1155
show large, 1167
Import,
database data as XML, 1068
database data based on XML Schema, 1076
database structure as XML Schema, 1072
MS Word document as XML, 1072
text file as XML, 1066
Importing taxonomies, 1123

Indentation,
in Text View, 916
Indentation guides, 1094
Info Window, 1194, 1195
Info ウィンドウ XSLT タブ,
XSL アウトラインも参照 ,370
とXSLT ドキュメント,370
とZIP フォルダーの作成 ,370
とプロジェクト,370
の記述 ,370
Insert,
breakpoint - XSLT debugger, 1032
row (in Authentic View), 1041
tracepoint - XSLT debugger, 1032
Insert command,
in Grid View, 951
Integrating,
XMLSpy in applications, 1602
Intelligent Editing, 1175
Internet, 1200
Introduction,
code generator, 789

J

Java, 1622
API, 1561
avoiding exceptions in generated code, 870
ClassPath, 1561
code, 788
Java 拡張関数 ,
XSLT とXQuery, 1738
インスタンスメソッド、インスタンスフィールド,1744
コンストラクター ,1743
データ型変換、Java から XPath/ XQuery へ ,1746
データ型変換、XPath/ XQuery から Java へ ,1745
ユーザー定義の JAR ファイル ,1742
ユーザー定義のクラスファイル ,1739
概要 ,1738
静的メソッド、静的フィールド,1743
JDBC,
データ接続インターフェイスとして ,650
接続の設定 (Windows), 667
JRE,
Eclipse に対応した XMLSpy プラグイン ,779
JSON, 479
convert JSON instance to/ from XML instance, 1084
convert JSON schema to/ from XML schema, 1087
JSON スキーマ ,486
allOf, 528
any, 527
anyOf, 528
JSON インスタンスからの生成 ,504
multiple, 527
not, 528
oneOf, 528
アトミック方 ,525
オブジェクト,516,519
オブジェクトと依存性 ,521
オブジェクトプロパティ,516
演算子 ,528
基本型 ,525
型セレクター (any, multiple), 527
指定されていないプロパティ,519
数値定義 ,525
説明 ,491
単純型 ,525
配列 ,524
文字列定義 ,525
用語 ,491
JSON スキーマビュー ,509
デザインビュー ,515
構成 ,529
JSON データ,
オブジェクト,488
サンプル ,488
型 ,488
配列 ,488
JSON ドキュメント,
XML から、または、へ変換 ,508
XML からの変換 ,493
XML へ変換 XML から変換 ,498
XMLSpy 内で開く,486
グリッドビューで変換 ,498
テキストビューでの編集 ,493
検証 ,502
新規作成 ,486
JSON ドキュメントの構文チェック ,493,498
JSON ドキュメントの整形式チェック ,493
JSON ドキュメントの整形式のチェック ,498

K

Key map, 1197

Keyboard shortcut, 1160
Key- codes,
for your Altova software, 1197

L

Language,
scripting language - changing, 1215
Large markup (in Authentic View), 1040
Library, 883
Licenses,
for your Altova software, 1197
Line,
go to, 1093
Line length,
word wrap in text view, 1093
Line margin, 1094
Line- breaks, 1170
Linkbases,
referencing, 1123
Linkroles in XBRL, 1120
Little- endian, 1185
loading, 1406

M

Macro,
add to menu/ toolbar, 1164
edit button, 1167
Macros,
creating with Scripting Editor, 1228
debugging, 1232
editing with Scripting Editor, 1228
execution of, 1215
functions for, in Global Declarations, 1217
how to use in Scripting Project, 1228
overview, 1213
running, 1230
running application macros, 1143
setting as Auto- Macro in Scripting Editor, 1228
MainCatalog, 977
MapForce, 996
Markup (in Authentic View),
hide, 1040
show small/ large/ mixed, 1040
Maximum cell width, 1176
Memory,
storage of schema information, 1000
Menu, 1163
add macro to, 1164
add/ delete command, 1155
Authentic, 1034
Convert, 1066
customize, 1163
Default/ XMLSPY, 1163
delete commands from, 1163
DTD/ Schema, 989
Edit, 907
Help, 1196
Project, 925
Schema Design, 1001
SOAP, 1109
Tools, 1135
View, 1090
Window, 1194
WSDL, 1098
XML, 951
XSL/ XQuery, 1020
Menu Browser, 1096
Method names in generating code,
reserving, 870
Microsoft Access,
ADO を介しての接続 ,654
ADO を使用しての接続 ,683
Microsoft Office 2007, 245, 639
Microsoft SQL Server,
ADO を使用しての接続 ,684
ODBC を使用しての接続 ,687
Microsoft® SharePoint® Server, 943
MIME, 1173
Mixed markup (in Authentic View), 1040
Mostly recently used files,
list of, 906
Move Left command, 974
Move Right command, 974
Move up/ down,
row (Authentic View, 1041
MS SQL Server,
schema extensions, 1014
schema settings, 1015
MSXML, 1186
MSXML library,
generating code for, 871
msxsl:script, 1751

Multi- user, 1170
MySQL,
ODBC を使用しての接続 ,688

N

Named schema relationships,
MS SQL Server schema settings, 1015
Namespace Prefix,
inserting in Grid View, 983
Namespaces,
settings in Schema Design View, 1001
Namespaces in XBRL, 1121
New file,
creating, 889
Node,
show curr. exec. node, 1031
Non- XML files, 1173
NOTATION declaration,
add as child in Grid View, 969
appending in Grid View, 964
convert to in Grid View, 972
inserting in Grid View, 957

O

OASIS,
XML catalog, 977
Object Locator,
in Database Query window, 1047
ODBC,
データ接続インターフェイスとして ,650
接続のセットアップ ,664
ODBC ドライバー ,
使用可能か確認する ,664
Office Open XML, 245, 639
OLE DB,
データ接続インターフェイスとして ,650
OOXML,
Office Open XML を参照 ,245
Office Open XML 以下を参照 ,639
Open,
file, 893
Open Office XML,
Archive View にて作成 ,641
Archive View にて編集 ,641
サンプルファイル ,643
openContent, 164
Opening options,
file, 1170
Optimal Widths, 1093, 1176
Oracle,
schema extensions, 1013
schema settings, 1014
Oracle XML Db, 1062
Browse database, 1064
manage XML Schemas, 1063
Oracle データベース ,
JDBC を使用しての接続 ,694
ODBC を使用しての接続 ,690
Ordering Altova software, 1197
OS,
Altova 製品のための ,1772
Out of memory exceptions,
resolving, 870
Output formatting, 1170
Output windows,
toggling display on and off, 1195
Overview,
of XMLSpy API, 1263

P

Parameters,
passing to stylesheet via interface, 1023
Parent, 1396
Parser,
XSLT, 1186
Paste command, 908
PDF output,
generating in Authentic View from PXF file, 1042
PDF フォント,261
Plug- in,
ATL sample files, 1251
registration, 1246
User interface configuration, 1248
XMLSPY, 1245
Position,
Character, 1093
Line, 1093
PostgreSQL,
ODBC を使用しての接続 ,695

PostgreSQL,
直接接続する (ネイティブ), 669
PowerPoint 2007, 245, 639
Presentation, 1176
Pretty- print,
in Text View, 916
Print setup, 905
Printing options, 904
Processing Instruction,
add as child in Grid View, 966
appending in Grid View, 960
convert to in Grid View, 971
inserting in Grid View, 953
Profiler, 1029
Program settings, 1170
Programming points,
in Scripting Project, 1233
Progress OpenEdge データベース ,
JDBC を使用しての接続 , 699
ODBC を使用しての接続 , 697
Project,
properties, 947
Project menu, 925
Project Window, 1194, 1195
toggling display on and off, 1195
Projects,
adding active files to, 941
adding external folders to, 941
adding external Web folders to, 943
adding files to, 940
adding folders to, 941
adding global resources to, 940
adding related files to, 941
adding to source control, 934
adding URL to, 940
closing, 928
creating new, 927
most recently used, 950
opening, 927
overview, 925
reloading, 927
saving, 928
Properties and Events pane, 1210
PUBLIC,
identifier - catalog, 977
PXF file,
generating output documents from Authentic View, 1042

Q

Query,
see under Database Query window, 1044
see under Query Database, 1044
see under XQuery, 1044
Query Database command, 1044
Query pane,
in Database Query window, 1050

R

Redo command, 907
Regions in SQL scripts, 1050
Register,
plug- in, 1246
Registering your Altova software, 1197
Registry,
settings, 1170
Regular expressions, 921
in search string, 917
Reload, 1170
Reloading,
changed files, 897
Remove,
breakpoint - XSLT debugger, 1032
tracepoint - XSLT debugger, 1032
Repeated elements, 1175
Replace,
text, 917
text in document, 920
text in multiple files, 921
Reset,
menu commands, 1163
shortcut, 1160
toolbar & menu commands, 1157
Restart,
XSLT debugger, 1031
Return キー ,
Enter キーを参照 , 476
RichEdit, 1040
Row,
append (in Authentic View), 1041
delete (in Authentic View), 1041

Row,
duplicate (in Authentic View), 1041
insert (in Authentic View), 1041
move up/ down, 1041
Row in Grid View,
appending, 973
inserting, 973
RTF output,
generating in Authentic View from PXF file, 1042

S

save, 1398
Saving files,
encoding of, 897
schema, 1305
also see XML Schema, 989
assigning to DB XML, 1059
code generator, 788
converting to UML, 995
create DB based on schema, 1077
Design view, 1091
Documentation font, 1179
management and assignment in IBM DB2 databases, 1056
settings, 1170
Schema Design menu, 1001
Schema Design View,
Display all globals, 1012
Display diagram, 1013
zoom feature, 1012
Schema fonts, 1179
Schema Subsets, 1017, 1019
SchemaAgent, 348
connect to server from XMLSpy, 1016
disconnect from server, 1017
display schemas in, 1017
extended validation, 1017
XMLSpy からスキーマを開く,348
作業する ,344,348
SchemaAgent Server,
接続 ,342
SchemaAgent Server へ接続 ,342
SchemaDocumentationDlg,
AllDetails, 1435
Application, 1435
IncludeAll, 1437
IncludeAttributeGroups, 1437
IncludeComplexTypes, 1437
IncludeGlobalElements, 1438
IncludeGroups, 1438
IncludeIndex, 1438
IncludeLocalElements, 1439
IncludeRedefines, 1439
IncludeSimpleTypes, 1440
OptionsDialogAction, 1440
OutputFile, 1441
OutputFileDialogAction, 1441
OutputFormat, 1441
Parent, 1442
ShowAnnotations, 1442
ShowAttributes, 1442
ShowChildren, 1442
ShowConstraints, 1443
ShowDiagram, 1443
ShowEnumerations, 1443
ShowNamespace, 1443
ShowPatterns, 1444
ShowProgressBar, 1444
ShowProperties, 1444
ShowResult, 1445
ShowSingleFacets, 1445
ShowSourceCode, 1445
ShowType, 1445
ShowUsedBy, 1446
schemanativetype, 873
Schemas,
in memory, 1000
managing for IBM DB2, 1056
Script language, 1189
Scripting, 1189
Scripting Editor,
GUI description, 1210
Main Window, 1210
starting, 1142
Scripting Environment, 1206
usage overview, 1208
Scripting language, 1215
Scripting Project,
and Events, 1225
application event handlers, 1225
Event Handlers, 1213
Forms, 1213
Forms in, 1219
Global Declarations, 1213
Global Declarations in, 1217
Macros, 1213

Scripting Project,
Macros in, 1228
programming points, 1233
steps for creating, 1215
Scripting Project Tree pane, 1210
Scripting Projects,
for XMLSpy, 1208
for XMLSpy Project, 1208
Search,
see Find, 920
Select All command, 917
Settings, 1170
scripting, 1189
XSLT Debugger, 1033
Settings for file comparison, 1147
SharePoint® Server, 943
Shortcut, 1160
assigning/ deleting, 1160
show in tooltip, 1167
Show, 1194, 1195
Show curr. exec. nodes,
XSLT debugger, 1031
Side- by- side, 1176
Signature,
see XML Signature, 984
Small markup (in Authentic View), 1040
SOAP, 540, 550
create new request, 1109
debugger options, 1117
debugger session, 1115
request, 1117
request parameters, 1112
requests, 1111, 1116, 1117
send request to server, 1111
start proxy server, 1116
stop proxy server, 1117
WSDL からリクエストを送信 ,547
SOAP menu, 1109
SOAP コミュニケーション処理 ,553
SOAP デバッガー ,551
Visual Studio .NET 内の SOAP デバッガー ,777
ブレークポイントのセット,561
SOAP 検証 ,550
Source control, 1192
add to source control, 934
changing provider, 939
checking out, 931
enabling, disabling, 930
get latest version, 930
getting files, 930
installing a source- control plug- in, 749
open project, 929
properties, 938
refresh status, 939
removing from, 934
sharing from, 934
show differences, 937
show history, 936
supported providers, 928
undo check out, 933
Source control manager, 939
Spelling checker, 1135
custom dictionary, 1135
Spelling options, 1139
SPL, 872
code blocks, 872
conditions, 878
foreach, 878
global objects, 875
subroutines, 879
using files, 876
variables, 874
Splash screen, 1176
SPP file locations, 925
SPS,
assigning to new XML file, 889
SPS file,
assigning to XML file, 1036
SPS テーブル ,
動的テーブルの編集 ,446
SpyProject,
CloseProject, 1447
ProjectFile, 1447
RootItems, 1447
SaveProject, 1447
SaveProjectAs, 1448
SpyProjectItem,
ChildItems, 1448
FileExtensions, 1448
ItemType, 1449
Name, 1449
Open, 1449
ParentItem, 1449
Path, 1449
ValidateWith, 1449
XMLForXSLTransformation, 1450
XSLForXMLTransformation, 1450
XSLTransformationFileExtension, 1450

SpyProjectItem,
XSLTransformationFolder, 1450
SpyProjectItems,
AddFile, 1450
AddFolder, 1451
AddURL, 1451
Count, 1451
Item, 1452
RemoveItem, 1452
SQL Editor,
creating query in, 1053
description of, 1050
in Database Query window, 1050
SQL Server,
ADO を介しての接続 ,654
ADO.NET を使用して接続する ,659
manage XML Schemas, 1061
SQLite,
接続の設定 (Windows), 670
Start,
XSLT debugger, 1030
Start group,
add (context menu), 1167
Step into,
XSLT debugger, 1031
Step out,
XSLT debugger, 1031
Step over,
XSLT debugger, 1031
Stop,
XSLT debugger, 1030
Strip whitespace, 917
Structured text, 1175
Style, 1178
Stylesheet PI, 1029, 1030
StyleVision, 996
and XBRL, 1129, 1130
for editing StyleVision Power Stylesheet, 1036
StyleVision Power Stylesheet,
assigning to XML file, 1036
editing in StyleVision, 1036
Support Center, 1200
Sybase,
JDBC を使用しての接続 ,700
Syntax-coloring, 1173, 1176

T

Tab characters, 1170
Tab size,
and pretty-printing, 1094
setting, 1094
Table,
build automatically, 1173
colors, 1183
Table command,
in Grid View, 972
Table of contents, 1196
Table View, 1175
and switching to Grid View, 972
append row, 973
insert row, 973
sorting columns, 973, 974
Target namespaces in XBRL, 1122
Taxonomies,
importing, 1123
namespaces in, 1121
target namespaces in, 1122
Technical Support, 1200
Template files,
for new documents, 889
Templates,
of XML documents in Authentic View, 1034
terminate, 1308
Text,
add as child in Grid View, 966
appending in Grid View, 959
convert to in Grid View, 971
find and replace, 920
finding in document, 917
inserting in Grid View, 952
pretty-printing, 916
Text file,
export of XML data to, 1079
import as XML, 1066
Text view, 1090
auto-complete enable/disable, 1175
schema fonts, 1179
Text View Events, 1225
Text View Settings dialog, 1094
TextImportExportSettings,
DestinationFolder, 1452

TextImportExportSettings,
EnclosingCharacter, 1453
Encoding, 1453
EncodingByteOrder, 1453
FieldDelimiter, 1453
FileExtension, 1453
HeaderRow, 1453
ImportFile, 1454
Tile,
horizontally, 1194
vertically, 1194
Toggle, 1194, 1195
Toolbar,
activate/deactivate, 1157
add command to, 1155
add macro to, 1164
create new, 1157
reset toolbar & menu commands, 1157
show large icons, 1167
Tools,
see also External applications, 1158
Tools menu, 1135
Tooltip, 1167
show, 1167
show shortcuts in, 1167
Topic,
view on TOC, 1196
Tracepoint,
dialog box, 1032
Transformation,
see XSLT transformation, 1022
Trusted locations for Authentic scripts, 1042
Turn off automatic validation, 1173
Types,
built in, 882

U

UCS-2, 1185
UML,
converting schemas to, 995
Undo command, 907
Unicode のサポート,
Altova 製品内で ,1775
Unnamed element relationships,
MS SQL Server schema settings, 1015
Unselected, 1092
Update Entry Helpers command, 983
URL, 1311, 1398, 1406, 1407
sending by e-mail, 903
User interface,
configure using plug-in, 1248
User Reference, 888
UTF-16, 1185

V

Validation, 977
assigning DTD to XML document, 989
assigning XML Schema to XML document, 990
WSDL files, 982
Variables,
in SPL, 874
View,
Browser view, 1092
Collapse, 1092
Enhance Grid view, 1090
Expand, 1092
Go to File, 1093
Go to line/char, 1093
Optimal widths, 1093
Schema Design view, 1091
Text View, 1090
View menu, 1090
Visual Studio,
adding the XMLSpy ActiveX Controls to the toolbox, 1605
generating code for, 871
Visual Studio .Net,
とXMLSpy, 772
とXMLSpy デバッガー ,777
とXMLSpy の違い ,775
VS .NET,
とXMLSpy 統合パッケージ ,773
VS .NET に対応した XMLSpy プラグイン ,
インストール ,773

W

Watch for changes, 1170
Web Server, 1200
Well-formedness check, 975
Whitespace,

Whitespace,
removing, 917
Whitespace markers, 1094
Window,
Cascade, 1194
Entry-Helper, 1195
Info, 1194, 1195
Open, 1195
Project, 1194, 1195
Tile horizontally, 1194
Tile vertically, 1194
Window menu, 1194
Windows,
Altova 製品のためのサポート,1772
フロート、ドッキング、タブ ,97
自動非表示 ,97
Word 2007, 245, 639
Word 2007+ output,
generating in Authentic View from PXF file, 1042
Word document,
import as XML, 1072
Word wrap,
enable/disable, 1093
Wrap,
word wrap enable/disable, 1093
Wrapper classes,
in generated code, 871
WSDL, 540, 545
1.1 components, 1098
2.0 components, 1100
binding in 1.1, 1099
binding in 2.0, 1102
converting from 1.1 to 2.0, 1107
generate documentation, 1104
interface in 2.0, 1100
messages in 1.1, 1098
operations in 1.1, 1099
portType in 1.1, 1099
PortTypes の作成 ,542
reparse document, 1107
service in 1.1, 1100
service in 2.0, 1102
SOAP デバッガー ,551
SOAP リクエストを送信 ,547
types, 1103
web service, 1109
ウェブサービスへ接続 ,545
オペレーションの作成 ,542
サービスの作成 ,544
スキーマを開く,545
ドキュメンテーションの作成 ,547
バインディングの作成 ,543
パラメーターの作成 ,542
ポートの作成 ,544
メッセージの作成 ,542
検証 ,545
新規ドキュメントの作成 ,541
名前空間 ,541
WSDL files,
Validation, 982
WSDL fonts, 1180
WSDL menu, 1098
WSDL チュートリアル ,541
WSDL デザインビュー ,
WSDL の記述 ,221
サービス ,222
バインディング ,222
ファイルの閲覧 ,221
ポートタイプ ,222
メインウィンドウ ,222
機能 ,221
WSDL ビュー ,
WSDL ドキュメントにインポート,225
概要入力ヘルパー ,225
詳細入力ヘルパー ,225
入力ヘルパー ,225

X

XBRL, 565
and MapForce, 1129
and StyleVision, 1130
arcroles, 1118
generate documentation, 1125
linkroles, 1120
namespaces, 1121
target namespaces, 1122
XBRL fonts, 1181
XBRL menu, 1118
XBRL View,
settings, 1128
XBRL タクソノミ,566
タクソノミも参照 ,567
XBRL タクソノミにおけるリンクロール ,581
XBRL テーブル定義エディター ,609
XBRL におけるタクソノミ,566,567

XBRL における検索 ,
XBRL 内を検索を参照 ,634
XBRL ビュー ,232
メインウィンドウ内の プレゼンテーション タブ ,236
メインウィンドウ内の 計算 タブ ,236
メインウィンドウ内の 定義 タブ ,236
メインウィンドウ内の要素 ,232
入力ヘルパー ,238
XBRL ビュー、検索 ,
XBRL 内を検索を参照 ,634
XBRL フォーミュラエディター ,588
XBRL 内を検索 ,634
ウィンドウ ,637
結果 ,637
検索語 ,634
XBRL 内を検索ウィンドウ ,116
Xerces XML library,
generating code for, 871
XInclude, 252
adding as child in Grid View, 966
appending in Grid View, 960
inserting in Grid View, 913, 953
inserting in Text View, 913
XML ドキュメントに挿入 ,252
XML,
copying as structured text, 909
Oasis catalog, 977
spelling checker, 1135
XML data,
exporting to database, 1082
exporting to text file, 1079
XML DB,
loading new data row into Authentic View, 1036
新たな XML データ行のロード ,466
XML Diff,
comparing directories, 1145
comparing files, 1143, 1147
XML document,
assigning to XSLT stylesheet, 1030
browse Oracle XML Db, 1064
XML file,
generate from DTD or XML Schema, 996
XML Import,
based on schema, 1076
XML menu, 951
XML prolog,
add as child in Grid View, 966
appending in Grid View, 960
convert to in Grid View, 971
inserting in Grid View, 953
XML Schema, 989
also see Schema, 989
assigning to DB XML, 1059
assigning to XML document, 990
configuring Content Model View, 1009
content model diagram, 1003
converting to DTD, 992
generate outline XML file from, 996
generating code from, 999
generating documentation of, 1004
generating from XML document, 991
go to definition in from XML document, 990
go to from XML document, 990
management and assignment in IBM DB2 databases, 1056
managing for IBM DB2, 1056
menu commands related to, 989
MS SQL Server extensions, 1014
MS SQL Server schema settings, 1015
namespaces settings in Schema Design View, 1001
Oracle extensions, 1013
Oracle schema settings, 1014
settings in Schema Design View, 1001
XML Signature, 1037
XML Signatures,
creating, 984
verifying, 986
XML text,
copying, 909
XML スキーマ ,30,325
<alternative> 要素 ,182
DTD, 250
DTD から生成 (Enterprise ならびに Professional エディションのみ) ,326
XML スキーマへの変換 (Enterprise ならびに Professional エディションのみ) ,327
XML ドキュメントの生成 ,327
XML ドキュメントを編集中に修正する ,81
インクルードされたスキーマのフラット化 ,328
グリッドビューで編集 (Enterprise ならびに Professional エディションのみ) ,327
グローバル コンポーネント ,164
グローバルリソース ,250
コンテンツ モデルの編集 ,170
コンポーネンツの追加 ,34
スマートフィックス ,210
チュートリアル ,30
テキストビューで編集 ,327
デザインビューでのナビゲーション ,56
ビューの設定 ,39

XML スキーマ , 30, 325
基本的なスキーマの作成 , 30
検証 , 210
新しいファイルの作成 , 31
他のスキーマを含める , 328
名前空間の定義 , 33
要素を追加する , 38
XML スキーマのスマートフィックス , 210
XML スキーマの検証 , 210
XML スキーマ定義 ,
高度な機能 , 44
XML ドキュメント, 247
Authentic View にて開く, 435
Authentic ビューにて編集 , 256
DTD から生成 , 326
XInclude の挿入 , 252
XML スキーマから生成 (Enterprise ならびに Professional エディションのみ) , 327
XPath 条件式の評価 , 323
XQuery 実行の自動化 , 259
XSLT による変換 , 259
XSLT 変換の自動化 , 259
エンコーディング , 323
グリッドビューにて編集 (Enterprise ならびに Professional エディションのみ) , 254
スキーマの生成 , 323
スキーマを割り当て (DTD を含む) , 250
テキストのインポートとエクスポート, 323
テキストビューにて修正 , 68
テキストビュー編集機能 , 252
デフォルトビュー , 248
と XQuery, 259
とテキストビューにおけるコメント, 252
とテキストビューにおける空白行 , 252
ノードの XPath 条件式 , 252
ファイルパスの挿入 , 252
開く, 248
検索と置き換え , 323
検証 , 250
自動検証 , 248
新規に作成 , 63
整形式のチェック , 250
妥当性をチェックする , 71
保存 , 248
XML ドキュメントの検証 , 250
XML ドキュメントの整形式をチェック , 250
XML ドキュメント作成 ,
チュートリアル , 62
XML パーサー ,
について , 1773
XML 署名 , 314, 474
検証 , 317
作成 , 316
xml:base, 218
xml:id, 218
xml:lang, 218
xml:space, 218
XML- Conformance, 1173
XMLData,
AppendChild, 1492
EraseAllChildren, 1493
EraseCurrentChild, 1493
GetChild, 1494
GetChildKind, 1495
GetCurrentChild, 1495
GetFirstChild, 1496
GetNextChild, 1496
HasChildren, 1498
HasChildrenKind, 1498
InsertChild, 1498
IsSameNode, 1499
Kind, 1499
MayHaveChildren, 1500
Name, 1500
Parent, 1500
TextValue, 1501
XMLSchemas,
flattening included schemas, 1019
splitting into subsets, 1017
XMLSpy, 888
integration, 1602
plug- in registration, 1246
ヘルプ , 122
機能 , 122
XMLSpy API,
documentation, 1261
overview, 1263
XMLSpy command table, 1630
XMLSpy Enterprise Edition,
ユーザーマニュアル , 3
XMLSpy integration,
example of, 1615, 1616, 1617
XMLSPY plug- in, 1245
XMLSpy でのプロジェクト,
その利点 , 88
作成方法 , 89
XMLSpy でのプロジェクト管理 , 87
XMLSpy における SchemaAgent, 341
XMLSpy におけるデータベース , 649

XMLSpy 統合パッケージ ,773 ,779
XMLSpy 内の XQuery Update, 389
XMLSpyCommand,
in XMLSpyControl, 1647
XMLSpyCommands,
in XMLSpyControl, 1649
XMLSpyControl, 1649
documentation of, 1602
example of integration at application level, 1615, 1616, 1617
examples of integration at document level, 1613
integration using C#, 1613
integration using HTML, 1617
object reference, 1647
XMLSpyControlDocument, 1656
XMLSpyControlPlaceHolder, 1662
XMLSpyDocumentEditor,
MarkUpView, 1540
XMLSpyLib, 1261, 1263
Application, 1294
AuthenticDataTransfer, 1314
AuthenticRange, 1318
AuthenticSelection (obsolete), 1518
AuthenticView, 1344
CodeGeneratorDlg, 1361
DatabaseConnection, 1366
Dialogs, 1372
DocEditEvent (obsolete), 1502
DocEditView (obsolete), 1521
Document, 1376
Documents, 1403
DTDSchemaGeneratorDlg, 1407
ElementList, 1411
ElementListItem, 1412
ExportSettings, 1413
FileSelectionDlg, 1415
GenerateSampleXMLDlg, 1426
GridView, 1431
ProjectItem, 1448
SchemaDocumentationDlg, 1434
SpyProject, 1446
SpyProjectItems, 1450
TextImportExportSettings, 1452
XMLData, 1491
XML- Text, 1175
XPath,
XML ドキュメントでノードを生成 ,252
評価 ,103
XPath 1.0,
in XPath Evaluator, 975
XPath 2.0,
in XPath Evaluator, 975
XPath Evaluator,
usage, 975
XPath of selected node in XML document,
copying to the clipborad, 912
XPath ウィンドウ ,103
XPath の評価 ,103
XPaths,
トレースポイントの設定 ,414
XPath- Watch ウィンドウ ,
XSLT/ XQuery デバッガー ,408
XPointer,
XML ドキュメントでノードを生成 ,252
XPointer of selected node in XML document,
copying to the clipborad, 912
XQuery, 379, 385
DB サポート,395
IBM DB2 に対する関数 ,395
passing variables to the XQuery document, 1023
XML データベースの問い合わせ ,395
インテリジェントな編集機能 ,380
テキストビューにて編集 ,375
ドキュメント検証 ,385
ファイルを開く,377
拡張関数 ,1738
実行 ,385
入力ヘルパー ,378
XQuery documents,
analyzing execution time of, 1029
XQuery Execution, 1026
XQuery options, 1188
XQuery Update,
PUL, 389
プレビュー ,389
XQuery Update Facility, 389
XQuery Update options, 1188
XQuery Update 内のPUL, 389
XQuery デバッガー ,
XSLT/ XQuery デバッガーを参照 ,399
XQuery ドキュメント,
実行時間の分析 ,420
XQuery ファイル ,
XMLSpy にてファイル拡張子をセット,377
XQuery プロセッサー ,
Altova 製品内で ,1774
XQuery プロファイラー ,
結果のチャート,430

XQuery 更新 ,386
XQuery 実行 ,386
XSD 1.0 と 1.1,
編集モード,156
XSD モード,156
XSD 検証 ,210
xsi:type,
使用方法 ,65
XSL,
see XSLT, 1030
XSL Speed Optimizer, 1021
XSL Speed Optimizer options, 1188
XSL アウトライン ,367
XSL アウトラインウィンドウ ,368
XSL スピードオプティマイザー ,373
XSL 概要ウィンドウ ,114
XSL 変換 ,
XSLT を参照 ,83
XSL/ XQuery menu, 1020
xsl:call- template, 368
XSL:FO,
とXSLT 変換 ,365
xsl:param, 368
xsl:with- param, 368
XSLT, 1093
processor, 1186
XMLSpy における機能 ,362
XMLSpy を使った変換 ,365
XMLSpyにて修正する ,86
テキストビューにおける自動補完 ,363
ドキュメント,363
とバッチ変換 ,365
拡張関数 ,1738
検証 ,363
入力ヘルパー ,363
XSLT Debugger,
breakpoints/ tracepoints dialog box, 1032
debug windows, 1033
enable/ disable breakpoint, 1032
enable/ disable tracepoint, 1032
end debugger session, 1031
insert/ remove breakpoint, 1032
insert/ remove tracepoint, 1032
restart debugger, 1031
settings, 1033
show curr. exec. nodes, 1031
start debugger, 1030
step into, 1031
step out, 1031
step over, 1031
stop debugger, 1030
XSLT parameters,
passing to stylesheet via interface, 1023
XSLT stylesheet,
assigning to XML document, 1029
assigning XML document to, 1030
opening, 1030
XSLT stylesheet for FO,
assigning to XML document, 1029
XSLT stylesheets,
analyzing execution time of, 1029
XSLT transformation, 1021, 1022
to FO, 1022
to PDF, 1022
XSLT スタイルシート,
実行時間の分析 ,420
XSLT デバッガー ,
Visual Studio .NET 内の XSLT デバッガー ,777
XSLT テンプレート,
XSL アウトラインウィンドウにて ,368
XSLT とXQuery のためのJava 拡張関数 ,
Java 拡張関数を参照する ,1738
XSLT とXQuery のための拡張関数 ,1738
XSLT ドキュメント,
XSL アウトラインを使った編集と管理 ,367
とInfo ウィンドウ ,370
XSLT ドキュメント構造 ,114
XSLT に対する MSXSL スクリプト,1751
XSLT プロセッサー ,
Altova 製品内で ,1774
XSLT プロファイラー ,
結果のチャート,430
XSLT へ変換 ,
XSLT ファイルを割り当てる ,84
XSLT 関数 ,
XSL アウトラインウィンドウにて ,368
XSLT 変換 ,
XMLSpyにて ,85
チュートリアル ,83
XSLT/ XQuery デバッガー ,400
XPath- Watch ウィンドウ ,408
インターフェースの記述 ,400
キーボードショートカット,419
コーススタックウィンドウ ,408
コンテキストウィンドウ ,407
セッションの開始 ,405
ツールバーアイコン ,401
テンプレートウィンドウ ,409

XSLT/ XQuery デバッガー ,400
トレースウィンドウ ,410
トレースポイントウィンドウ ,414
ブレークポイントの使用方法 ,412
メカニズムの記述 ,400
メッセージウィンドウ ,409
機能と使用方法 ,399
情報ウィンドウ ,406,410
設定 ,403
変数ウィンドウ ,407
XSLT/ XQuery デバッグ ,
使用するファイル ,399
XSLT/ XQuery プロファイラー ,420
XSLT/ XQuery 内のスクリプト,
拡張関数で参照する ,1738

Z

ZIP ファイル ,245,639
Archive View にて作成 ,644
Archive View にて編集 ,644
zip64mode,
enabling in the build.xml file, 870
Zoom feature,
in Schema Design View, 1012
アサーションメッセージ ,211
アップデートリストの保留 (PUL), 389
イメージフォーマット,
Authentic View にて ,475
イメージ形式 ,
Authentic View にて ,475
インターネットの使用 ,
Altova 製品内で ,1776
ウェブサービス ,
接続する ,545
エイリアス ,
グローバルリソース参照 ,703
エリアチャートの機能 ,293
ゲージチャートの機能 ,293
ローソク足チャートの機能 ,293
円グラフの機能 ,293
折れ線グラフの機能 ,293
棒グラフの機能 ,293
エンティティ,
Authentic View にて削除 ,456
Authentic View にて挿入 ,442
Authentic View にて定義する ,472
Authentic Viewにて挿入 ,456
XML スキーマをベースにする XML にて ,250
エンティティ入力ヘルパー ,
Authentic View にて ,450
エンドユーザー使用許諾契約書 ,1777,1781
オーバーライド,164
オープン コンテンツ モデル ,185
オプション ,
3D チャート,301
エリアチャートの機能 ,293
ゲージチャートの機能 ,293
チャートX-軸 ,298
チャートY-軸 ,300
チャートZ-軸 ,301
チャートグリッド,298,300,301
チャートサイズ ,302
チャートのタイトル ,291
チャートの色 ,296
チャートの背景 ,291
チャートの凡例 ,291
チャートフォント,303
ローソク足チャートの機能 ,293
円グラフの機能 ,293
折れ線グラフの機能 ,293
棒グラフの機能 ,293
カスタマイズ ,120
カタログ ,336
クエリ ,
Authentic View にて DB クエリを表示 ,467
グリッドビュー ,75,145
XML ドキュメントの編集 (Enterprise ならびに Professional エディションのみ) ,254
データの入力 ,67
テーブル ,147
とデータのインポート/エクスポート,147
とテーブルビュー ,77
入力ヘルパー ,150
入力ヘルパを使用する ,75
編集 ,146
要素と属性の追加 ,75
グローバル リソース ,
定義 ,704
グローバル リソース XML ファイル ,704
グローバルスキーマコンポーネント,
検索と名前変更 ,360
グローバルなコメント,
表示される行 ,160
グローバルな属性 ,164
グローバルな要素 ,164

グローバルリソース ,
データベースの型の定義 ,711
ファイルの型の定義 ,706
ファイル型とフォルダー型の使用 ,713
フォルダーの型の定義 ,709
構成の変更 ,716
使用 ,713,716
グローバルリソースの ,
構成 ,704
グローバルリソースの構成 ,716
グローバル要素 ,
XML スキーマを使用する ,51
ゲージチャートの機能 ,293
コーススタックウィンドウ ,
XSLT/ XQuery デバッガー ,408
コールグラフプロファイリング ,420
コミュニケーション処理 ,
SOAP デバッガー ,553
コメント,160
コメントアウトならびにアンコメント,
テキストビューにおける XML ドキュメント,252
コンテキストウィンドウ ,
XSLT/ XQuery デバッガー ,407
コンテキストメニュー ,
Authentic Viewにて ,454
コンテンツ モデル ビュー ,
インターフェイスの説明 ,172
コンポジッターとコンポーネント,172
一般的な説明 ,170
条件型のい割り当て ,182
図に表示されるオブジェクト,172
編集 ,177
コンテンツモデル ,
基本的なモデルの作成 ,34
切り替え属性 ,53
コンテンツモデルビュー ,31
コンポーネント定義 ,
再利用する ,45
コンポーネント入力ヘルパー ,202,206
コンポジッター ,
シーケンスコンポジッター ,34
サポートオプション ,122
シーケンスコンポジッター ,
使用する ,34
システム DSN,
セットアップ ,664
スキーマ ,
WSDL スキーマを開く,545
XML スキーマを参照 ,30
ドキュメンテーション ,58
スキーマ コンポーネント,
のコンテンツ モデル ,170
スキーマ ビュー ,
コンポーネント入力ヘルパー ,202,206
詳細入力ヘルパー ,205
入力ヘルパー ,202
戻る、進む ,219
スキーマ ビューにおけるアサーション ,187,192
スキーマ ビューにおけるコメント,164
スキーマ ビューにおけるコンポジッター ,172
スキーマ ビューにおける記法 ,164
スキーマ ビューにおける処理命令 ,164
スキーマ ビューにおける属性 ,187,189
スキーマ ビューにおける属性グループ ,189
スキーマ ビューにおける注釈 ,164
スキーマ ビューにおける同一性制約 ,187,195
スキーマ ビューにおける入力ヘルパー ,202
スキーマ、検索 ,
スキーマの中を検索を参照 ,351
スキーマサブセット,328
スキーマの修正 ,
コンテンツ モデル ,170
スキーマの中を検索 ,351
ウィンドウ ,359
グローバルコンポーネント,360
グローバルコンポーネントの名前変更 ,360
スキーマの中を検索を参照 ,351
プロパティとプロパティの値による検索結果の絞り込み ,
355
結果 ,359
検索ならびに置き換えコマンドの実行 ,358
検索をコンポーネントに絞り込む ,353
検索語 ,352
置き換え語 ,352
範囲の設定 ,357
スキーマの中を検索ウィンドウ ,116
スキーマの内を検索 ,
検索コマンドの実行 ,636
スキーマビュー ,154
ビューの設定 ,39
スキーマビュー ,検索 ,
スキーマの中を検索を参照 ,351
スキーマルール ,330
スキーマへルールセットを追加 ,330
定義 ,331
スキーマ概要 ,31
アイコン ,160
グローバルなコメント,160

スキーマ概要 , 31
コンポーネントの並べ替え , 160
とグローバル コンポーネント, 160
とグローバル コンポーネントの表示 , 160
とコンテンツ モデル ビュー , 159, 160
編集 , 160
スキーマ型の生成 , 214
ステータスバー , 118
スマート制約 , 214
ソフトウェア製品ライセンス , 1781
タクソノミ,
インポート, 574
ターゲット名前空間 , 573
とリレーションシップ , 581, 582, 585
とリンクベース , 576
とリンクロール , 581
と新規タクソノミウィザード, 569
ファイル内におけるタクソノミ, 576
ファイル内部にあるタクソノミ, 567
作成ステップ , 566
新たに作成 , 569
名前空間 , 573
要素の追加 , 578
タクソノミ内のリレーションシップ , 581, 582, 585
タクソノミ内のリンクベース , 576
チャート,
3D 設定 , 301
XSLT/ XQuery プロファイラーにて , 430
X- 軸 , 298
X- 軸選択 , 272
Y- 軸 , 300
Y- 軸選択 , 276
Z- 軸 , 301
エクスポート, 303
エリアチャートの機能 , 293
オーバーレイ , 281
グリッドライン , 298, 300, 301
ゲージチャートの機能 , 293
サイズ , 302
サンプル（ローソク足チャート）, 311
サンプル（基本）, 304
サンプル（詳細）, 305
ソース XPath, 269
タイトル , 291
チャートデータ, 279
チャート設定 , 281
フォント, 303
ローソク足チャートの機能 , 293
円グラフの機能 , 293
外観 , 283
概要 , 263
系列色 , 296
再ロード, 263
色の定義 , 296
色の範囲 , 296
折れ線グラフの機能 , 293
塗りつぶし, 296
背景色 , 291
複数タブで使用 , 263
棒グラフの機能 , 293
凡例の削除 , 291
凡例の追加 , 291
目盛り間隔 , 302
余白 , 302
チャートウィンドウ , 117
チャートデータテーブル ,
作成方法 , 265
チャート関数 ,
サンプル , 1734
チャートデータ構造 , 1729
リスト, 1725
チュートリアル ,
WSDL, 541
ツールバー , 118
ディレクトリの比較 , 746, 748
データのインポート/エクスポート,
とグリッドビュー , 147
データベース ,
DB も参照 , 466
XMLSpy 内でのサポート, 702
とグローバルリソース , 716
データベース /テーブルビュー ,
使用方法 , 77
データベースドライバー ,
概要 , 652
データベース接続 ,
ウィザードの開始 , 651
グローバルリソースからの再使用 , 671
セットアップサンプル , 672
設定 , 650
テーブル ,
Authentic View にて , 438
グリッドビューのテーブル , 147
動的（SPS）テーブルの編集 , 446
テーブル レイアウトプレビュー , 630
フォーミュラの作成 , 632
テーブルパラメーター , 628
テーブルビュー ,

テーブルビュー ,
使用方法 ,77
テキスト,
Authentic Viewにてフォーマット,456
Authentic Viewにて編集 ,456
テキストビュー ,125
XML ドキュメントにおける空白行 ,252
XML ドキュメントに対する特別な編集機能 ,252
XML ドキュメントへコメントを追加 ,252
インテリジェントな編集機能 ,130
インデント,125,127
ソース折りたたみ ,127
テキストのフォーマット,125
フォントプロパティ,125
ブックマーク ,127
ワードラップ,125
行番号 ,127
入力ヘルパー ,141
編集 ,68
テキストビューにおけるインデント,125
テキストビューにおけるソースの折りたたみ ,127
テキストビューにおけるフォーマット,125
テキストビューにおけるフォント,125
テキストビューにおけるブックマーク ,127
テキストビューにおけるワードラップ ,125
テキストビューにおける行番号 ,127
テキストビューにおける入力ヘルパー ,141
テキストビューにおける編集 ,130
テキストビューのズーム ,127
デバッガー ,
SOAP, 551
デバッガーウィンドウ ,
配置 ,410
デフォルト オープン コンテンツ モデル ,185
テンプレート XML ファイル ,
Authentic View にて ,435
テンプレートウィンドウ ,
XSLT/ XQuery デバッガー ,409
ドキュメンテーション ,
スキーマ ,58
ドッキングウィンドウ ,97
トレースウィンドウ ,414
XSLT/ XQuery デバッガー ,410
トレースポイント,
XSLT/ XQuery デバッガー ,414
ナビゲーション ,
スキーマデザインでのショートカット,56
パーサー ,
Altova 製品に内蔵の ,1773
パラメーター ,
DB クエリにて ,467
バリデーター ,
Altova 製品内で ,1773
ファイル DSN,
セットアップ ,664
ファイルの中を検索ウィンドウ ,114
ファイルの比較 ,746,747
ファイルパス ,
XML ドキュメントに挿入 ,252
ファイル拡張子 ,
XQuery ファイル ,377
ファイルの種類に対して拡張子を設定 ,336
フォーミュラ,
テーブル レイアウトプレビュー内でのフォーミュラの作成 ,
632
プラットフォーム ,
Altova 製品のための ,1772
ブレークポイント,
SOAP デバッガーを使用する ,561
XSLT/ XQuery デバッガーで使用する ,412
ブレークポイントダイアログ ,401
プログラマーのレファレンス ,1204
プロジェクト,
プロパティ,719
を使ったバッチ処理 ,723
概要 ,718
作成と編集の方法 ,719
使用する ,723
使用する利点 ,723
保存 ,719
名前をつける ,719
プロジェクトウィンドウ ,99
プロファイラー ,420
プロファイリング ,420
コールグラフ ,420
呼び出し回数 ,420
マークアップ ,
Authentic View にて ,446,448
マークアップを隠す ,446,448
メインウィンドウ ,98
メインウィンドウにおけるドキュメント,98
メッセージウィンドウ ,102
XSLT/ XQuery デバッガー ,409
メニューバー ,118
メモリ要件 ,1772
モデルグループ ,164
ユーザー DSN,
セットアップ ,664

ユーザーインターフェイス詳細 ,97
ユーザーマニュアル ,3,94
ライセンス ,1781
情報 ,1777
ライセンス計測 ,
Altova 製品にて ,1779
リンク ,
Authentic View にて移動 ,456
ルール ,
スキーマ検証 (スキーマルールを参照) ,330
ローソク足チャートの機能 ,293
移動ウィンドウ ,97
印刷 ,
Authentic View から印刷 ,443
円グラフ ,276
円グラフの機能 ,293
拡張グリッドビュー ,
グリッドビューを参照 ,67
拡張スキーマ検証 ,330
拡張検証 ,348
基底型 ,
修正 ,213
技術データ ,1771
空白行 ,
テキストビューにおける XML ドキュメント ,252
型 ,
XML ドキュメントでの拡張 ,65
検索 ,
正規表現の使用 ,135
検証 ,120
SchemaAgent により関連するスキーマの検証 ,348
WSDL ファイル ,545
XML ドキュメント ,71
スキーマルールを使った拡張 ,330
検証メッセージ ,102
呼び出し回数プロファイリング ,420
構文による色分け ,379
XQuery, 379
混合マークアップの表示 ,446,448
再定義 ,164
試用期間 ,
Altova ソフトウェア製品の試用 ,1778,1780
自動非表示ウィンドウ ,97
出現 ,
回数 ,34
署名 ,
XML 署名を参照 ,314
小さなマークアップの表示 ,446,448
省略可能な要素 ,
作成 ,41
証明書 ,319
証明書ストア ,319
証明書の公開鍵 ,319
証明書の秘密鍵 ,319
詳細入力ヘルパー ,34,205
情報 ウィンドウ ,
CSS ドキュメントに対して ,482
情報ウィンドウ ,101
HTML ドキュメントに対して ,480
XSLT/ XQuery デバッガー ,410
配置 ,410
条件型の割り当て ,182
条件型の代替 ,182
新規 XML ドキュメント,
作成 ,63
新規タクソノミの作成ウィザード ,569
新規機能 ,6
進む ,
スキーマ ビューにて ,219
整形式のチェック ,
XML ドキュメントに対して ,71
正規表現 ,
検索と置換の条件式 ,135
折れ線グラフの機能 ,293
設定 ,120
3D 設定 ,301
エリアチャートの機能 ,293
ゲージチャートの機能 ,293
チャート X- 軸 ,298
チャート Y- 軸 ,300
チャート Z- 軸 ,301
チャートグリッド ,298,300,301
チャートサイズ ,302
チャートのタイトル ,291
チャートの色 ,296
チャートの背景 ,291
チャートの凡例 ,291
チャートフォント ,303
ローソク足チャートの機能 ,293
円グラフの機能 ,293
折れ線グラフの機能 ,293
棒グラフの機能 ,293
選択されたノードへの XPath, 445
属性 ,53,164
コンテンツモデルビューに切り替え ,53
スキーマ定義にて ,53
属性値 ,
Authentic View にて入力 ,442

属性入力ヘルパー ,
Authentic View にて ,450
大きなマークアップの表示 ,446,448
単純型 ,
アサーション ,206
グローバル ,164
サンプル値 ,206
スキーマ定義にて ,45
パターン ,206
ファセット,206
匿名型 ,164
名前付き ,164
列挙 ,206
単純型のアサーション ,206
単純型のサンプル値 ,206
単純型のパターン ,206
単純型のファセット,206
単純型の派生 ,205
単純型の列挙 ,206
置き換え ,114
置換 ,
正規表現の使用 ,135
著作権情報 ,1777
定義 ,
3D 設定 ,301
X-軸設定 ,298
Y-軸設定 ,300
Z-軸設定 ,301
エリアチャートの機能 ,293
グリッドライン ,298,300,301
ゲージチャートの機能 ,293
チャートサイズ ,302
チャートのタイトル ,291
チャートの色 ,296
チャートフォント,303
チャート内のフォント,303
ローソク足チャートの機能 ,293
円グラフの機能 ,293
折れ線グラフの機能 ,293
棒グラフの機能 ,293
定義概要グリッド,509
貼付け ,
XML として ,454,456
テキストとして ,454,456
動的テーブル ,
編集 ,446
同一性制約 ,
コンテンツモデルビューに切り替え ,53
日付 ,
手動で変更 ,472
日付の選択 ,
Authentic View にて使用 ,471
入力ヘルパー ,101
XML ドキュメント,258
XQuery, 378
グリッドビューにて ,75
詳細 ,34
派生型 ,
基底型の切り替え ,213
背景情報 ,1771
配布 ,
Altova ソフトウェア製品の配布 ,1778,1780
Altova のソフトウェア製品 ,1777
比較 ,
ディレクトリの比較 ,748
ファイルとディレクトリの比較 ,746
ファイルの比較 ,747
表示の変更 ,
Authentic View へ ,446
評価期間 ,
Altova のソフトウェア製品 ,1777
複合型 ,45
グローバル ,164
スキーマ定義にて ,45
定義を拡張する ,45
匿名型 ,164
名前付き ,164
変数ウィンドウ ,
XSLT/ XQuery デバッガー ,407
編集ビュー ,124
法的な情報 ,1777
棒グラフの機能 ,293
名前空間 ,
WSDL ドキュメントにて ,541
XBRL におけるタクソノミ,573
スキーマ内の ,33
名前付きテンプレート,368
名前付きテンプレートの呼び出し,368
戻る ,
スキーマ ビューにて ,219
要素 ,41
省略可能にする ,41
内容を制限する ,41
要素の型 ,
XML ドキュメントで指定する ,65
要素入力ヘルパー ,
Authentic View, 450
履歴 ,219

列挙 ,
属性を定義する ,53