Altova XMLSpy 2021 Enterprise Edition

ユーザーマニュアル

Altova XMLSpy 2021 Enterprise Edition
ユーザーマニュアル

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

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

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

公開日: 2015-2021

(C) 2015-2021 Altova GmbH

目次

1 XMLSpy とこのドキュメントについて 28

1.1 新規機能 2021 ... 29
1.1.1 バージョン 2020 ... 29
1.1.2 バージョン 2019 ... 30
1.1.3 バージョン 2018 ... 31
1.1.4 バージョン 2017 ... 32
1.2 Windows ファイルパス ... 33
1.3 RaptorXML Server について ... 34

2 XMLSpy チュートリアル 35

2.1 XMLSpy インターフェイス ... 36
2.1.1 表示 ... 37
2.1.2 ウィンドウ ... 38
2.1.3 メニューとツールバー ... 40
2.1.4 テキストビューの設定 ... 42
2.1.5 アプリケーションオプション ... 44
2.2 XML スキーマ：基本 ... 46
2.2.1 新しい XML スキーマファイルの作成 ... 46
2.2.2 名前空間の定義 ... 48
2.2.3 コンテンツモデルの定義 ... 49
2.2.4 ドラッグアンドドロップによる要素の追加 ... 54
2.2.5 コンテンツモデルビューの構成 ... 55
2.2.6 基本的なスキーマの完成 ... 57
2.3 XML スキーマ: 高度な機能 ... 61
2.3.1 複合型と単純型を扱う ... 61
2.3.2 グローバル要素の参照 ... 68
2.3.3 属性と列挙型 ... 70
2.4 XML スキーマ：XMLSpy の機能 ... 73
2.4.1 スキーマナビゲーション ... 73

2.4.2 スキーマドキュメンテーション ... 75
2.5 XML ドキュメント ... 79
2.5.1 新規 XML ファイルの作成 ... 79
2.5.2 要素の型の指定 ... 81
2.5.3 グリッドビューを使ったデータの入力 ... 83
2.5.4 テキストビューを使ったデータの入力 ... 84
2.5.5 ドキュメントの検証 ... 88
2.5.6 要素と属性の追加 ... 92
2.5.7 データベース/テーブルビューでの編集 ... 94
2.5.8 スキーマの修正 ... 98
2.6 XSLT 変換 ... 99
2.6.1 XSLT ファイルの割り当て ... 99
2.6.2 XMLファイルの変換 ... 100
2.6.3 XSLファイルの修正 ... 101
2.7 プロジェクト管理 ... 103
2.7.1 プロジェクトの利点 ... 103
2.7.2 プロジェクトの作成 ... 103
2.8 おしまい ... 106

3 インターフェイスと環境 107

3.1 グラフィカルユーザインターフェース (GUI) ... 108
3.1.1 メイン ウィンドウ ... 109
3.1.2 プロジェクト ウィンドウ ... 110
3.1.3 情報ウィンドウ ... 112
3.1.4 入力ヘルパー ... 113
3.1.5 出力ウィンドウ: XPath/XQuery ... 113
3.1.6 出力ウィンドウ: XSL アウトライン ... 114
3.1.7 出力ウィンドウ: HTTP ... 115
3.1.8 出力ウィンドウ: ファイルの中を検索 ... 115
3.1.9 出力ウィンドウ: スキーマの中を検索 ... 117
3.1.10 出力ウィンドウ: XBRL 内を検索 ... 117
3.1.11 出力ウィンドウ: チャート ... 118
3.1.12 メニューバー、ツールバー、ステータスバー ... 120
3.1.13 出力ウィンドウ: XULE ... 121

3.2 アプリケーション環境 ... 122
3.2.1 設定、カスタマイズ ... 122
3.2.2 チュートリアル、プロジェクト、サンプル ... 124
3.2.3 XMLSpy 製品機能と Altova 製品 ... 125

4 編集ビュー 126

4.1 ファイルの自動的なバックアップ ... 128
4.2 テキストビュー ... 130
4.2.1 テキストビューでのフォーマット ... 131
4.2.2 ドキュメントの表示 ... 132
4.2.3 テキストビューでの編集 ... 135
4.2.4 ドキュメント内の移動 ... 137
4.2.5 検索および置換 ... 140
4.2.6 テキストビューでの入力ヘルパー ... 145
4.2.7 分割ビュー ... 146
4.2.8 テキストビューショートカット ... 148
4.3 グリッドビュー ... 149
4.3.1 グリッドビューでの編集 ... 149
4.3.2 グリッドビューテーブル ... 151
4.3.3 グリッドビューでの入力ヘルパー ... 155
4.3.4 グリッドビューショートカット ... 156
4.4 スキーマ ビュー ... 159
4.4.1 XSD モード: XSD 1.0 と 1.1 ... 161
4.4.2 スキーマ概要 ... 164
4.4.3 コンテンツ モデル ビュー ... 176
4.4.4 属性、アサーション、同一性制約 ... 195
4.4.5 スキーマ ビューにおける入力ヘルパー ... 209
4.4.6 検証とスマートフィックス ... 219
4.4.7 アサーションメッセージ ... 220
4.4.8 基底型の修正 ... 222
4.4.9 スマート制約 ... 223
4.4.10 xml:base、xml:id、xml:lang、xml:space ... 228
4.4.11 戻ると進む: 位置の移動 ... 230
4.5 WSDL ビュー ... 231

4.5.1 メインウィンドウ ... 232
4.5.2 概要入力ヘルパー ... 236
4.5.3 詳細入力ヘルパー ... 242
4.6 XBRL ビュー ... 243
4.6.1 メインウィンドウ：要素タブ ... 243
4.6.2 メインウィンドウ：定義、プレゼンテーション、計算、フォーミュラ、テーブルタブ ... 247
4.6.3 XBRL ビューの入力ヘルパー ... 250
4.6.4 XBRL ビューの設定 ... 254
4.7 Authentic ビュー ... 255
4.8 ブラウザービュー ... 256
4.9 アーカイブビュー ... 257
4.10 共通のショートカット ... 259

5 XML 261
5.1 XML ドキュメントを作成、開く、保存する ... 262
5.2 スキーマの割り当てと検証 ... 264
5.3 テキストビュー内の XML ... 266
5.4 グリッドビュー内の XML ... 268
5.4.1 ドキュメントの表示 ... 270
5.4.2 ドキュメント構造 ... 273
5.4.3 ドキュメントコンテキスト ... 275
5.4.4 グリッドビュー内でのドラッグとドロップ ... 278
5.4.5 テーブルの表示 ... 281
5.4.6 フィルター ... 285
5.4.7 フォーミュラ ... 286
5.4.8 イメージ ... 289
5.4.9 チャート ... 290
5.4.10 グリッドビュー内のコンテキストメニュー ... 293
5.4.11 グリッドビュー設定 ... 296
5.5 Authentic ビュー内の XML ... 299
5.6 XML ドキュメントの入力ヘルパー ... 301
5.7 XML ドキュメントの検証 ... 302
5.8 XML フラグメントの挿入 ... 304
5.9 XSLT と XQuery による処理 ... 306

5.10 PDF フォント ... 308
5.11 チャート ... 310
5.11.1 チャートの作成 ... 313
5.11.2 ソース XPath ... 317
5.11.3 X-軸の選択 ... 320
5.11.4 Y-軸の選択 ... 325
5.11.5 チャートのデータ ... 329
5.11.6 オーバーレイ ... 331
5.11.7 チャートの設定：クイックリファレンス ... 332
5.11.8 チャートの設定と外観 ... 334
5.11.9 エクスポート ... 357
5.11.10 チャートの例：基本 ... 358
5.11.11 チャートの例：詳細 ... 360
5.11.12 チャートの例：ローソク足チャート ... 366
5.12 XML 署名 ... 370
5.12.1 XML 署名の作成 ... 372
5.12.2 XML 署名の検証 ... 375
5.12.3 証明書を使用する ... 377
5.13 その他の機能 ... 381

6 DTD と XML スキーマ 383

6.1 DTD ... 384
6.2 XML スキーマ ... 386
6.3 スキーマサブセット ... 387
6.4 スキーマルール ... 391
6.4.1 ルールセットの管理 ... 391
6.4.2 ルールセットの定義 ... 392
6.5 SchemaAgent との連携 ... 397
6.5.1 SchemaAgent Server へ接続 ... 398
6.5.2 検索パス上のスキーマを開く ... 400
6.5.3 IIR の使用 ... 401
6.5.4 SchemaAgent でスキーマを閲覧 ... 406
6.5.5 SchemaAgent 検証 ... 406
6.6 XMLSpy 内のカタログ ... 409

6.6.1 カタログのしくみ ... 409
6.6.2 XMLSpy 内のカタログ構造 ... 410
6.6.3 カタログのカスタム化 ... 411
6.6.4 環境変数 ... 413
6.7 スキーマの中を検索 ... 415
6.7.1 検索語 ... 417
6.7.2 コンポーネント ... 418
6.7.3 プロパティ ... 419
6.7.4 スコープ ... 422
6.7.5 検索と置き換えコマンド ... 423
6.7.6 結果と情報 ... 425
6.7.7 グローバルコンポーネントの検索と名前変更 ... 425

7 XSLT 428
7.1 XSLT ドキュメント ... 429
7.2 XSLT 処理 ... 431
7.3 XSL アウトライン ... 433
7.3.1 XSL アウトラインウィンドウ ... 434
7.3.2 情報ウィンドウ ... 437
7.4 XSL スピードオプティマイザー ... 440

8 XQuery 442
8.1 XQuery ドキュメントの編集 ... 444
8.1.1 XQuery ドキュメント ... 444
8.1.2 XQuery 入力ヘルパー ... 445
8.1.3 XQuery の構文による色分け ... 446
8.1.4 XQuery インテリジェント編集 ... 448
8.2 XQuery 評価 ... 451
8.3 XQuery 検証 ... 452
8.4 XQuery/Update 実行 ... 453
8.5 XQuery Update Facility ... 456
8.5.1 アップデートのプレビューと適用 ... 456
8.5.2 Update オペレーションと構文 ... 458

8.6 XQuery と XML データベース ... 462

9 XSLT/XQuery デバッガー とプロファイラー 466

9.1 XSLT と XQuery デバッガー ... 467
9.1.1 メカニズムとインターフェース ... 468
9.1.2 コマンドとツールバーアイコン ... 469
9.1.3 デバッガー設定 ... 473
9.1.4 デバッグセッションの開始 ... 474
9.1.5 情報ウィンドウ ... 475
9.1.6 ブレークポイント ... 483
9.1.7 トレースポイント ... 486
9.1.8 デバッガーショートカット ... 491
9.2 XSLT と XQuery プロファイラー ... 492
9.2.1 XSLT プロファイリング ... 497
9.2.2 XQuery プロファイリング ... 500
9.2.3 プロファイリングの結果: エクスポートとチャート ... 503

10 XPath/XQuery 条件式 506

10.1 XPath/XQuery ウィンドウについて ... 507
10.2 式の評価 ... 509
10.3 式のデバッグ ... 514
10.4 式ビルダー ... 520
10.5 JSONX のための Query 式 ... 522
10.6 注意点 ... 524

11 Authentic 526

11.1 Authentic View チュートリアル ... 528
11.1.1 AUTH-VIEW 内で XML ドキュメントを開く ... 529
11.1.2 Authentic View インターフェイス ... 530
11.1.3 ノードを操作する ... 532
11.1.4 Authentic View 内にデータを挿入する ... 535
11.1.5 属性の値を入力する ... 536
11.1.6 エンティティを入力する ... 537

11.1.7 ドキュメントを印刷する ... 538
11.2 Authentic View インターフェイス ... 540
11.2.1 GUI の概要 ... 540
11.2.2 Authentic View ツールバーアイコン ... 541
11.2.3 Authentic View メインウィンドウ ... 544
11.2.4 Authentic View エントリヘルパー ... 546
11.2.5 Authentic View コンテキストメニュー ... 550
11.3 Authentic View 内で編集する ... 553
11.3.1 基本的な編集 ... 553
11.3.2 Authentic View 内のテーブル ... 557
11.3.3 DB の編集 ... 565
11.3.4 日付と作業する ... 571
11.3.5 エンティティを定義する ... 573
11.3.6 XML 署名 ... 574
11.3.7 Authentic View 内のイメージ ... 575
11.3.8 Authentic View 内のキーストロール ... 576
11.4 Authentic スクリプト ... 577

12 HTML と CSS 579
12.1 HTML ... 580
12.2 CSS ... 582

13 JSON と JSON スキーマ 587
13.1 JSON データ ... 589
13.2 JSON スキーマ ... 592
13.3 JSON ラインと JSON コメント ... 594
13.4 JSON テキストビュー ... 595
13.5 JSON グリッドビュー ... 600
13.5.1 グリッドビュー内でのドキュメントの表示 ... 603
13.5.2 ドキュメント構成の編集 ... 606
13.5.3 ドキュメントコンテンツの編集 ... 607
13.5.4 グリッドビュー内でのドラッグとドロップ ... 610
13.5.5 テーブルの表示 ... 613

13.5.6 フィルター ... 617
13.5.7 フォーミュラ ... 618
13.5.8 イメージ ... 622
13.5.9 チャート ... 623
13.5.10 グリッドビュー内のコンテキストメニュー ... 625
13.5.11 グリッドビューの設定 ... 627
13.5.12 JSON スキーマビュー ... 630
13.6 JSON ドキュメントの検証 ... 662
13.7 JSON フラグメントの挿入 ... 664
13.8 XSLT/XQuery を使用した JSON 変換 ... 666
13.9 JSON のための XQuery 式 ... 668
13.10 JSON インスタンスからの JSON スキーマの生成 ... 670
13.11 JSON インスタンスを JSON スキーマから生成する ... 673
13.12 JSON と XML の間の変換 ... 674

14 Avro、Avro スキーマ 675
14.1 Avro スキーマ ... 678
14.2 JSON フォーマットの Avro データ ... 680
14.3 Avro ビュー: Avro バイナリのグリッドビュー ... 681

15 WSDL と SOAP 683
15.1 WSDL チュートリアル ... 684
15.1.1 新規ドキュメントの作成 ... 684
15.1.2 ポートタイプの作成 ... 685
15.1.3 バインディングの作成 ... 687
15.1.4 サービスとポートの作成 ... 688
15.1.5 WSDL ドキュメントの検証 ... 689
15.1.6 ウェブサービスへの接続とファイルを開く ... 690
15.1.7 WSDL ファイルから SOAP リクエストを送信 ... 691
15.1.8 WSDL ドキュメンテーションの作成 ... 692
15.1.9 WSDL 2.0 への変換 ... 694
15.2 SOAP ... 695
15.2.1 SOAP 検証 ... 695

15.2.2 SOAP デバッガー ... 696

16 HTTP 711

16.1 リクエストの送信 ... 712
16.2 送信するリクエストのインポート ... 716
16.3 レスポンスの受信 ... 718

17 XBRL 721

17.1 XBRL タクソノミ マネジャー ... 722
17.1.1 XBRL タクソノミ マネジャー の実行 ... 725
17.1.2 タクソノミのインストール ... 726
17.1.3 インストール済みのタクソノミをビューする ... 728
17.1.4 タクソノミのアンインストール ... 728
17.1.5 コマンドライン インターフェイス ... 729
17.2 基本のプロシージャ ... 738
17.2.1 タクソノミ：新規と既存 ... 738
17.2.2 タクソノミファイルの概要 ... 739
17.2.3 新規タクソノミの作成 ... 741
17.2.4 タクソノミ内の名前空間 ... 747
17.2.5 タクソノミのインポート ... 748
17.2.6 タクソノミファイルの設定 ... 750
17.2.7 タクソノミへ要素の追加 ... 752
17.2.8 リレーションシップとリンクロール ... 756
17.2.9 リレーションシップの作成：パート1 ... 757
17.2.10 リレーションシップの作成：パート2 ... 760
17.3 追加プロシージャ ... 763
17.3.1 優先されるラベル ... 763
17.3.2 型付けされているドメイン ... 764
17.3.3 重複の検知と重複の解除 ... 765
17.3.4 インライン XBRL ... 766
17.4 XBRL フォーミュラエディター ... 767
17.4.1 フォーミュラ リンクベース およびリンクロール ... 767
17.4.2 フォーミュラ コンポーネント ... 769

17.4.3 コンポーネント プロパティとコンテンツの編集 785
17.4.4 フォーミュラコンポーネントリレーション 786
17.4.5 フォーミュラ パラメーター 787
17.4.6 フォーミュラ コンポーネントの検索 789
17.5 XBRL テーブル定義エディター 791
17.5.1 テーブル リンクベース およびリンクロール 792
17.5.2 テーブル構造 794
17.5.3 テーブルコンポーネント 807
17.5.4 コンポーネントプロパティとコンテンツの編集 812
17.5.5 テーブルコンポーネントリレーションシップ 813
17.5.6 テーブルパラメーター 814
17.5.7 テーブル レイアウトプレビュー 817
17.5.8 テーブルコンポーネントの検索 819
17.6 XULE 821
17.6.1 XULE ドキュメント 821
17.6.2 XULE ウィンドウ 824
17.6.3 XULE 実行 827
17.7 XBRL 内を検索 829
17.7.1 検索語 829
17.7.2 コマンドの実行 831
17.7.3 結果と情報 833

18 Office Open XML、ZIP、EPUB 834
18.1 OOXML ファイルの編集 836
18.2 OOXML サンプルファイル 838
18.3 ZIP ファイル 840
18.4 EPUB ファイル 842

19 データベース 846
19.1 データソースへの接続 847
19.1.1 接続ウィザードの開始 848
19.1.2 データベースドライバー概要 849
19.1.3 ADO 接続の設定 852

19.1.4 ADO.NET 接続のセットアップ ... 859
19.1.5 ODBC 接続の設定 ... 865
19.1.6 JDBC 接続の設定 ... 868
19.1.7 PostgreSQL 接続のセットアップ ... 872
19.1.8 SQLite 接続の設定 ... 873
19.1.9 グローバルリソースからの接続の使用 ... 876
19.1.10 データベース接続のサンプル ... 877
19.2 サポートされているデータベース ... 933

20 Altova グローバルリソース 934

20.1 グローバル リソースの定義 ... 935
20.1.1 ファイル ... 937
20.1.2 フォルダー ... 941
20.1.3 データベース ... 942
20.2 グローバルリソースを使用する ... 944
20.2.1 ファイルとフォルダーの割り当て ... 944
20.2.2 データベースの割り当て ... 947
20.2.3 構成の切り替え ... 948

21 プロジェクト 950

21.1 プロジェクトの作成と編集 ... 951
21.2 プロジェクトを使用する ... 955

22 RaptorXML Server 957

22.1 サーバーとサーバー構成の追加 ... 958
22.2 RaptorXML Server を使用して検証 ... 962
22.3 検証オプション ... 963
22.3.1 共通のオプション ... 963
22.3.2 DTD に対して XML ... 964
22.3.3 DTD ... 964
22.3.4 W3C スキーマに対して XML ... 965
22.3.5 W3C スキーマ ... 965
22.3.6 XBRL インスタンス ... 966

22.3.7 XBRL タクソノミ ... 967
22.3.8 XBRL タクソノミパッケージ ... 968
22.3.9 XBRL バージョンレポート ... 969
22.3.10 XSLT ... 969
22.3.11 XQuery ... 970
22.3.12 JSON ... 971
22.3.13 JSON スキーマ ... 972
22.3.14 AVRO ... 972
22.3.15 AVRO JSON ... 972
22.3.16 AVRO スキーマ ... 973
22.3.17 EDGAR ... 973
22.4 RaptorXML Server使用しての XSLT と XQuery ... 975

23 ファイル/ディレクトリの比較 977
23.1 ファイルの比較 ... 978
23.2 ディレクトリの比較 ... 979

24 Source Control 980
24.1 Setting Up Source Control ... 982
24.2 Supported Source Control Systems ... 983
24.3 Local Workspace Folder ... 985
24.4 Application Project ... 986
24.5 Add to Source Control ... 988
24.6 Working with Source Control ... 990
24.6.1 Add to, Remove from Source Control ... 990
24.6.2 Check Out, Check In ... 991
24.6.3 Getting Files as Read-Only ... 993
24.6.4 Copying and Sharing from Source Control ... 995
24.6.5 Changing Source Control ... 998
24.7 Source Control with Git ... 1000
24.7.1 Enabling Git Source Control with GIT SCC Plug-in ... 1001
24.7.2 Adding a Project to Git Source Control ... 1001
24.7.3 Cloning a Project from Git Source Control ... 1002

25 Visual Studio 内の XMLSpy 1005

25.1 ...XMLSpy プラグインのインストール ... 1006
25.2 ...XMLSpy スタンドアロンとの違い ... 1007
25.3 ...Visual Studio での XMLSpy デバッガー ... 1009

26 Eclipse 内の XMLSpy 1010

26.1 ...Eclipse 用の XMLSpy プラグインをインストール ... 1011
26.2 ...Eclipse 内の XMLSpy エントリポイント ... 1013
26.3 ...XMLSpy のデバッガーパースペクティブ ... 1016

27 Code Generator 1017

27.1 ...What's new 1019
27.2 ...Generating Code from XML Schemas or DTDs ... 1021
27.2.1 About Schema Wrapper Libraries (C++) ... 1023
27.2.2 About Schema Wrapper Libraries (C#) ... 1026
27.2.3 About Schema Wrapper Libraries (Java) ... 1028
27.2.4 Integrating Schema Wrapper Libraries ... 1030
27.2.5 Example: Book Library ... 1033
27.2.6 Example: Purchase Order ... 1055
27.3 ...Reference to Generated Classes (C++) ... 1062
27.3.1 altova::DateTime ... 1062
27.3.2 altova::Duration ... 1065
27.3.3 altova::DayTimeDuration ... 1067
27.3.4 altova::YearMonthDuration ... 1068
27.3.5 altova::meta::Attribute ... 1068
27.3.6 altova::meta::ComplexType ... 1069
27.3.7 altova::meta::Element ... 1070
27.3.8 altova::meta::SimpleType ... 1070
27.3.9 [YourSchema]::[CDoc] ... 1071
27.3.10 [YourSchema]::[ElementType] ... 1074
27.3.11 [YourSchema]::MemberAttribute ... 1075
27.3.12 [YourSchema]::MemberElement ... 1075

27.4 ...Reference to Generated Classes (C#) ... 1077
27.4.1 Altova.Types.DateTime ... 1077
27.4.2 Altova.Types.DateTimeFormat ... 1080
27.4.3 Altova.Types.Duration ... 1082
27.4.4 Altova.Xml.Meta.Attribute ... 1084
27.4.5 Altova.Xml.Meta.ComplexType ... 1085
27.4.6 Altova.Xml.Meta.Element ... 1085
27.4.7 Altova.Xml.Meta.SimpleType ... 1086
27.4.8 [YourSchema].[Doc] ... 1087
27.4.9 [YourSchema].[ElementType] ... 1089
27.4.10 [YourSchemaType].MemberAttribute ... 1090
27.4.11 [YourSchemaType].MemberElement ... 1090
27.5 ...Reference to Generated Classes (Java) ... 1092
27.5.1 com.altova.types.DateTime ... 1092
27.5.2 com.altova.types.Duration ... 1096
27.5.3 com.altova.xml.meta.Attribute ... 1099
27.5.4 com.altova.xml.meta.ComplexType ... 1100
27.5.5 com.altova.xml.meta.Element ... 1100
27.5.6 com.altova.xml.meta.SimpleType ... 1101
27.5.7 com.[YourSchema].[Doc] ... 1102
27.5.8 com.[YourSchema].[ElementType] ... 1103
27.5.9 com.[YourSchema].[YourSchemaType].MemberAttribute ... 1105
27.5.10 com.[YourSchema].[YourSchemaType].MemberElement ... 1105
27.6 ...Code Generation Tips ... 1107
27.7 ...Code Generator Options ... 1108
27.8 ...SPL Reference ... 1110
27.8.1 Basic SPL structure ... 1110
27.8.2 Declarations ... 1111
27.8.3 Variables ... 1113
27.8.4 Predefined variables ... 1114
27.8.5 Creating output files ... 1115
27.8.6 Operators ... 1117
27.8.7 Conditions ... 1117
27.8.8 Collections and foreach ... 1118
27.8.9 Subroutines ... 1119

27.8.10 Built in Types ... 1122

28 メニューコマンド 1127

28.1 ...ファイルメニュー ... 1128
28.1.1 新規作成 ... 1128
28.1.2 開く ... 1132
28.1.3 再ロード ... 1137
28.1.4 エンコード ... 1137
28.1.5 閉じる、全て閉じる、非アクティブを全て閉じる ... 1138
28.1.6 保存、名前を付けて保存、全て保存 ... 1138
28.1.7 メールで送信 ... 1143
28.1.8 印刷 ... 1144
28.1.9 印刷プレビュー、印刷設定 ... 1146
28.1.10 最近使用されたファイル、終了 ... 1146
28.2 ...編集メニュー ... 1148
28.2.1 元に戻す、やり直し ... 1149
28.2.2 切り取り、コピー、貼り付け、削除 ... 1149
28.2.3 XML テキストとしてコピー ... 1150
28.2.4 構造化テキストとしてコピー ... 1150
28.2.5 XPath をコピー ... 1154
28.2.6 XPointer をコピー ... 1154
28.2.7 挿入 ... 1154
28.2.8 整形 ... 1157
28.2.9 空白の削除 ... 1158
28.2.10 全て選択 ... 1158
28.2.11 検索、次を検索 ... 1159
28.2.12 置換 ... 1162
28.2.13 ファイル内の検索 ... 1164
28.2.14 ブックマーク コマンド ... 1166
28.2.15 コメントの IN/OUT ... 1167
28.3 ...プロジェクトメニュー ... 1168
28.3.1 新規プロジェクト ... 1170
28.3.2 プロジェクトを開く ... 1170
28.3.3 プロジェクトの再ロード ... 1171

28.3.4 プロジェクトを閉じる ... 1171
28.3.5 プロジェクトの保存、名前を付けて保存 ... 1171
28.3.6 ソース管理 ... 1171
28.3.7 プロジェクトにファイルを追加 ... 1185
28.3.8 プロジェクトにグローバルリソースを追加 ... 1185
28.3.9 プロジェクトに URL を追加 ... 1185
28.3.10 プロジェクトにアクティブなファイルを追加 ... 1185
28.3.11 プロジェクトにアクティブならびに関係するファイルを追加 ... 1186
28.3.12 プロジェクトにアクティブならびに関係するフォルダーを追加 ... 1186
28.3.13 プロジェクトに外部フォルダーを追加 ... 1186
28.3.14 プロジェクトに外部ウェブフォルダーを追加 ... 1189
28.3.15 スクリプトの設定 ... 1194
28.3.16 プロパティ ... 1194
28.3.17 最近使用されたプロジェクト ... 1197
28.4 ...XML メニュー ... 1198
28.4.1 挿入 ... 1198
28.4.2 追加 ... 1206
28.4.3 子の追加 ... 1213
28.4.4 変換 ... 1220
28.4.5 テーブル ... 1222
28.4.6 左へ移動 ... 1224
28.4.7 右へ移動 ... 1224
28.4.8 要素の中へ ... 1224
28.4.9 XPath を評価 ... 1225
28.4.10 整形式のチェック ... 1225
28.4.11 XML の検証 ... 1227
28.4.12 サーバー上の XML の検証(ハイパフォーマンス) ... 1231
28.4.13 WSDL ファイルの検証 ... 1232
28.4.14 編集中に検証 ... 1233
28.4.15 入力ヘルパーの更新 ... 1233
28.4.16 名前空間プレフィックス ... 1233
28.4.17 XML 署名の作成 ... 1234
28.4.18 XML 署名の検証 ... 1236
28.5 ...JSON メニュー ... 1239
28.5.1 型 ... 1239

28.5.2 子を前/後に挿入、子の付加、子の追加 1240
28.5.3 配列/オブジェクト内のラップ 1240
28.5.4 右から左への読み取り 1240
28.5.5 移動 1241
28.5.6 コピー 1241
28.5.7 展開、折りたたみ 1241
28.5.8 昇順に並べ替え/降順に並べ替え 1242
28.6 ...DTD/スキーマメニュー 1243
28.6.1 DTD の割り当て 1243
28.6.2 スキーマの割り当て 1244
28.6.3 他の DTD を読み込み 1244
28.6.4 DTD へ移動 1244
28.6.5 スキーマへ移動 1245
28.6.6 定義へ移動 1245
28.6.7 DTD/スキーマの生成 1245
28.6.8 DTD のフラット化 1247
28.6.9 DTD をスキーマに変換 1247
28.6.10 フラット化されたスキーマ 1249
28.6.11 スキーマを DTD に変換する 1249
28.6.12 UML へ変換 1250
28.6.13 MapForce で DB/Excel/EDI から XML を生成 1251
28.6.14 StyleVision で HTML/PDF/Word 出力をデザイン 1251
28.6.15 サンプル XML/JSON ファイルの生成 1251
28.6.16 プログラムコードの生成 1255
28.6.17 メモリキャッシュの破棄 1255
28.7 ...スキーマ設計メニュー 1256
28.7.1 スキーマ設定 1256
28.7.2 図の保存 1259
28.7.3 ドキュメンテーションの生成 1259
28.7.4 表示設定 1264
28.7.5 ズーム 1267
28.7.6 全てのグローバルを表示 1268
28.7.7 図を表示 1268
28.7.8 データベースに対するスキーマ拡張 1268
28.7.9 SchemaAgent Server に接続 1272

28.7.10 SchemaAgent Server から切断 ... 1273
28.7.11 SchemaAgent で表示 ... 1273
28.7.12 SchemaAgent 検証 ... 1273
28.7.13 スキーマサブセットの作成 ... 1274
28.7.14 スキーマのフラット化 ... 1275
28.8 ...XSL/XQuery メニュー ... 1277
28.8.1 XSL 変換 ... 1279
28.8.2 XSL スピードオプティマイザー ... 1279
28.8.3 XSL-FO 変換 ... 1280
28.8.4 XSL パラメーター / XQuery 実行 ... 1281
28.8.5 XQuery/Update 実行 ... 1285
28.8.6 バックマッピングの有効化 ... 1285
28.8.7 XSLT/XQuery プロファイリングの有効化 ... 1287
28.8.8 XSL の割り当て ... 1287
28.8.9 XSL-FO の割り当て ... 1288
28.8.10 サンプル XML ファイルの割り当て ... 1288
28.8.11 XSL へ移動 ... 1288
28.8.12 ソース命令に移動 ... 1288
28.8.13 コンテキストノードに移動 ... 1289
28.8.14 デバッガーの開始/実行 ... 1289
28.8.15 デバッガーの停止 ... 1289
28.8.16 デバッガーの再起動 ... 1289
28.8.17 デバッグセッションの終了 ... 1290
28.8.18 ステップイン ... 1290
28.8.19 ステップアウト ... 1290
28.8.20 ステップオーバー ... 1290
28.8.21 現在実行中のノードを表示 ... 1291
28.8.22 ブレークポイントの挿入/削除 ... 1291
28.8.23 トレースポイントの挿入/削除 ... 1291
28.8.24 ブレークポイントの有効化/無効化 ... 1291
28.8.25 トレースポイントの有効化/無効化 ... 1292
28.8.26 ブレークポイント/トレースポイント ... 1292
28.8.27 デバッグウィンドウ ... 1293
28.8.28 デバッグ設定 ... 1293
28.9 ...Authentic メニュー ... 1294

28.9.1 新規ドキュメント ... 1294
28.9.2 データベースの編集 ... 1295
28.9.3 StyleVision スタイルシートの割り当て ... 1296
28.9.4 StyleVision スタイルシートの編集 ... 1297
28.9.5 XML データの新たな行を選択し編集 ... 1297
28.9.6 XML 署名 ... 1298
28.9.7 XML エンティティの定義 ... 1299
28.9.8 マークアップの表示 ... 1301
28.9.9 リッチエディット ... 1301
28.9.10 行の追加/挿入/複製/削除 ... 1302
28.9.11 上に/下に移動 ... 1303
28.9.12 HTML、RTF、PDF、Word 2007+ ドキュメントの生成 ... 1303
28.9.13 信頼された場所 ... 1303
28.10 DB メニュー ... 1305
28.10.1 データベースをクエリする ... 1305
28.10.2 IBM DB2 ... 1318
28.10.3 SQL Server ... 1324
28.10.4 Oracle XML DB ... 1326
28.11 変換メニュー ... 1331
28.11.1 テキストファイルのインポート ... 1331
28.11.2 データベースのデータをインポート ... 1334
28.11.3 Microsoft Word ドキュメントをインポート ... 1339
28.11.4 データベース構造から XML スキーマを作成 ... 1339
28.11.5 XML スキーマを元にデータベースをインポート ... 1343
28.11.6 XML スキーマを元にデータベース構造を作成 ... 1344
28.11.7 テキストファイルへエクスポート ... 1347
28.11.8 データベースへエクスポート ... 1350
28.11.9 XML インスタンス を JSON に変換する（ JSON から XML インスタンスに変換する） ... 1352
28.11.10 XML スキーマ を JSON スキーマに変換する（ JSON スキーマから XML スキーマに変換する） ... 1355
28.12 表示メニュー ... 1358
28.12.1 テキストビュー ... 1358
28.12.2 拡張グリッドビュー ... 1359
28.12.3 スキーマデザインビュー ... 1359
28.12.4 WSDL デザインビュー ... 1359

28.12.5 XBRL タクソノミビュー ... 1360
28.12.6 Authentic View ... 1360
28.12.7 ブラウザービュー ... 1360
28.12.8 展開 ... 1361
28.12.9 縮退 ... 1361
28.12.10 全て展開 ... 1361
28.12.11 未選択を縮退 ... 1361
28.12.12 最適な幅 ... 1362
28.12.13 ワードラップ ... 1362
28.12.14 行/文字へ移動 ... 1362
28.12.15 ファイルへ移動 ... 1363
28.12.16 テキストビュー設定 ... 1363
28.13 ブラウザーメニュー ... 1365
28.13.1 戻る ... 1365
28.13.2 進む ... 1365
28.13.3 中止 ... 1366
28.13.4 最新の状態に更新 ... 1366
28.13.5 フォント ... 1366
28.13.6 別のウィンドウ ... 1366
28.14 WSDL メニュー ... 1367
28.14.1 WSDL 1.1 コンポーネント ... 1367
28.14.2 WSDL 2.0 コンポーネント ... 1370
28.14.3 型、ならびに図の保存 ... 1374
28.14.4 ドキュメンテーションの生成 ... 1375
28.14.5 WSDL ドキュメントの再解析 ... 1378
28.14.6 WSDL 2.0 に変換 ... 1379
28.14.7 MapForce で WSDL プログラムコードを生成 ... 1379
28.15 SOAP メニュー ... 1380
28.15.1 新規 SOAP リクエストを作成 ... 1380
28.15.2 サーバーにリクエストを送信 ... 1382
28.15.3 SOAP リクエスト設定 ... 1383
28.15.4 SOAP デバッグセッション ... 1387
28.15.5 実行 ... 1388
28.15.6 単一のステップ ... 1389
28.15.7 次のリクエストでブレーク ... 1389

28.15.8 次のレスポンスでブレーク ... 1389
28.15.9 プロキシサーバーの停止 ... 1389
28.15.10 SOAP デバッガーオプション ... 1389
28.16 XBRL メニュー ... 1391
28.16.1 アークロール ... 1391
28.16.2 リンクロール ... 1393
28.16.3 名前空間プレフィックス ... 1394
28.16.4 ターゲット名前空間の設定 ... 1395
28.16.5 パラメーターの値 ... 1396
28.16.6 インポート/参照 ... 1396
28.16.7 ID を使用してコンポーネントを検索する ... 1398
28.16.8 ドキュメンテーションの生成 ... 1398
28.16.9 ビューの設定 ... 1401
28.16.10 MapForce で DB、Excel、CSV から XBRL を生成 ... 1403
28.16.11 StyleVision で HTML/PDF/Word として XBRL を表示 ... 1403
28.16.12 (サーバーでの)フォーミュラの実行 ... 1403
28.16.13 (サーバーでの)テーブルの生成 ... 1406
28.16.14 (サーバー上での)重複の検知 ... 1408
28.16.15 XULE の実行 ... 1409
28.16.16 インライン XBRL の変換 ... 1410
28.16.17 サーバー上で EDGAR を検証 ... 1410
28.16.18 処理オプション ... 1411
28.17 ツールメニュー ... 1413
28.17.1 スペリング ... 1414
28.17.2 スペルチェックのオプション ... 1418
28.17.3 スクリプトエディター ... 1422
28.17.4 マクロ ... 1423
28.17.5 比較 ... 1423
28.17.6 グローバルリソース ... 1433
28.17.7 アクティブな構成 ... 1434
28.17.8 Raptor Server の管理 ... 1434
28.17.9 Raptor Server と構成 ... 1438
28.17.10 XBRL タクソノミ マネジャー ... 1438
28.17.11 カスタマイズ ... 1438
28.17.12 ツールバーとウィンドウの復元 ... 1453

28.17.13 オプション ... 1454
28.18 ウィンドウメニュー ... 1493
28.18.1 重ねて表示 ... 1493
28.18.2 上下に並べて表示 ... 1493
28.18.3 左右に並べて表示 ... 1493
28.18.4 プロジェクトウィンドウ ... 1493
28.18.5 情報ウィンドウ ... 1494
28.18.6 入力ヘルパー ... 1494
28.18.7 出力ウィンドウ ... 1494
28.18.8 プロジェクトおよび入力ヘルパー ... 1494
28.18.9 全てオン/オフ ... 1494
28.18.10 現在開かれているウィンドウのリスト ... 1495
28.19 ヘルプメニュー ... 1496
28.19.1 目次、インデックス、検索 ... 1496
28.19.2 キーボードマップ ... 1497
28.19.3 ライセンス登録、注文フォーム、登録、最新情報のチェック ... 1498
28.19.4 他のコマンド ... 1501
28.20 コマンドライン ... 1502

29 プログラマーのレファレンス 1503

29.1 Scripting Editor ... 1505
29.1.1 Creating a Scripting Project ... 1506
29.1.2 Built-in Commands ... 1518
29.1.3 Enabling Scripts and Macros ... 1528
29.2 IDE Plugins ... 1530
29.2.1 Registration of IDE PlugIns ... 1530
29.2.2 ActiveX Controls ... 1531
29.2.3 Configuration XML ... 1531
29.2.4 ATL Sample Files ... 1534
29.2.5 IXMLSpyPlugIn ... 1539
29.3 Application API ... 1544
29.3.1 Overview ... 1545
29.3.2 Interfaces ... 1579
29.3.3 Interfaces (obsolete) ... 1806

29.3.4 Enumerations ... 1843
29.3.5 Application API for Java (obsolete) ... 1857
29.4 ActiveX Integration ... 1905
29.4.1 必要条件 ... 1905
29.4.2 ActiveX コントロールをツールボックスに追加する ... 1906
29.4.3 アプリケーションレベルの統合 ... 1908
29.4.4 ドキュメントレベルの統合 ... 1910
29.4.5 ActiveX 統合のサンプル ... 1913
29.4.6 コマンド レファレンス ... 1932
29.4.7 オブジェクトレファレンス ... 1951

30 付録 1973
30.1 XSLT および XQuery エンジンに関する情報 ... 1974
30.1.1 XSLT 1.0 ... 1974
30.1.2 XSLT 2.0 ... 1974
30.1.3 XSLT 3.0 ... 1976
30.1.4 XQuery 1.0 ... 1976
30.1.5 XQuery 3.1 ... 1979
30.2 XSLT と XPath/XQuery 関数 ... 1980
30.2.1 Altova 拡張関数 ... 1981
30.2.2 その他の拡張関数 ... 2064
30.3 DB 内で生成された XML スキーマのデータ型 ... 2081
30.3.1 ADO ... 2081
30.3.2 MS Access ... 2082
30.3.3 MS SQL Server ... 2082
30.3.4 MySQL ... 2083
30.3.5 ODBC ... 2084
30.3.6 Oracle ... 2085
30.3.7 Sybase ... 2085
30.4 XML スキーマから生成された DB のデータ型 ... 2087
30.4.1 MS Access ... 2087
30.4.2 MS SQL Server ... 2088
30.4.3 MySQL ... 2090
30.4.4 Oracle ... 2092

30.5 ...技術データ ... 2094
30.5.1 OS とメモリ要件 ... 2094
30.5.2 Altova XML バリデーター ... 2094
30.5.3 Altova XSLT と XQuery エンジン ... 2094
30.5.4 Unicode のサポート ... 2094
30.5.5 インターネットの使用 ... 2095
30.6 ...ライセンス情報 ... 2096
30.6.1 電子的なソフトウェアの配布 ... 2096
30.6.2 ソフトウェアのアクティベーションとライセンスの計測 ... 2096
30.6.3 エンドユーザー使用許諾契約書 ... 2097

インデックス 2098

1 XMLSpy とこのドキュメントについて

Altova XMLSpy 2021 Enterprise Edition はJSON、XML、XML スキーマ、XSLT、XQuery、SOAP、W SDL, W eb サービス、OOXML、およびXBRL 技術を含むデザイン、編集、デバッグのためのエンタープライズクラスアプリケーションのための最も高度で広く使用されているXML とJSON エディターです。

XMLSpy はプラットフォーム更新済みのW indows 7 SP1、W indows 8、W indows 10 とプラットフォーム更新済みのW indows Server 2008 R2 SP1 またはは以降 で作動します。XMLSpy は64-ビット と32-ビット バージョンで使用することができ、お手頃価格の Altova MissionKit ツールパッケージの一部です。

このドキュメントは以下のセクションにより構成されています:

- 使用を開始するための チュートリアル 35
- 技術またはXMLSpy-固有の機能により分類されたXMLSpy 機能の説明
- メニューコマンド 1127 の説明
- プログラマーのレファレンス 1503
- XMLSpy の(i) XSLT とXQuery エンジン、および(ii) Altova のXPath/XQuery 拡張関数に関する情報を含む付録 1973

最終更新日: *2021 年 03 月 26 日*

1.1 新規機能 2021

バージョン 2021 リリース 2

- XML ドキュメントのグリッドビュー[268] が多種の新規機能と共に拡張されました。
- XML ドキュメントのグリッドビュー[268] に関するコマンドと設定が簡単使用のために整理されました グリッドビュー設定[296] とオプションダイアログ[1462] を参照してください。
- Settings of the JSON ドキュメントのグリッドビュー[600] の設定が簡単使用のために整理されました グリッドビュー設定[627] とオプションダイアログ[1462] を参照してください。
- DTD ドキュメントのグリッドビュー[384] が新規機能と共に拡張されました。
- XML グリッドビュー[278] と JSON グリッドビュー[610] 内の新規のドラッグオーバーレイ機能はグリッドビュー内のドラッグアンドドロップの使用を促進する重要な情報を提供します。
- 新規機能により XML フラグメント[304] と JSON フラグメント[664] を外部ソースから簡単に素早く追加することができます。
- グリッドビュー内で（ XML コメントなどの）追加コンポーネントをコンポーネント固有の書式設定[1472] に与えることができます。各コンポーネント型はグリッドビュー内で割り当てられた書式設定で表示されます。例えば、コメントはグリッドビュー内で黄色でハイライトされています。
- 編集中にドキュメントを検証する[1233] 新規機能を をオン/オフに切り替えることができます。
- Eclipse へのサポート[1010] が拡張され、以下のバージョンがカバーされています: 2020-12; 2020-09; 2020-06; 2020-03.

バージョン 2021

- XMLSpy との使用のためにタクソノミを簡単にインストール、更新、および管理可能にする XBRL XBRL タクソノミ マネジャー[722] ツール。
- JSON メニュー[1239] は以下の JSON グリッドビューの拡張機能を提供します: グリッドビューコンポーネントを選択し、アイテムを前後に挿入する。選択されたグリッドビューコンポーネントを 上/下、または、左/右に移動する。ドキュメントからすべてのコメントを削除する。
- 新規の JSON グリッドビュー機能: (i) コンテキストメニュー[625] が簡素化されました。(ii) グリッドをズームすることができます[603]。(iii) 複数の兄弟コンポーネントを展開/折りたたむ[603] ことができます。(iv) 文字列を含むセルは読み取りの方向を右から左に設定することができます（アラブ語およびヘブライ語など読み取りが右から左の言語と作業する際にこの機能は役に立ちます）。コピーコマンド[625] が拡張されました。
- JSON スキーマ検証[630] が $id プロパティを考慮するようになりました。
- 新規の スキーマ情報にアクセスするための Altova 拡張関数[2020]。
- Eclipse へのサポート[1010] が拡張され、以下をカバーしています: 2020.06; 2020.03; 2019.12; 2019.09.

1.1.1 バージョン 2020

バージョン 2020 リリース 2

- 予期なくプログラムが終了した場合、編集中のファイルは 自動的にバックアップされ復元されます[128]。自動的にバックアップを実行し[1455] その頻度を設定することができます。
- 変更時に編集 機能はテキストビューと JSON グリッドビュー[607] 内での記入時に検証および整形式エラーをフラグします。この機能をプログラムオプション[1455] およびツールバーアイコンでオン/オフに切り替えることができます。
- スキーマビュー[159] 内では 多種のコンテキストメニュー内の型定義に移動[184] コマンドによりコンポーネントの型定義（単純型または複合型）に移動が可能になりました。
- 新規の JSON グリッドビューオプション により以下を行うことができます: (i) フィルターとフォーミュラ[617] がリクエストでアプリケーション全体のメタデータファイルに自動的に保存されるか、または保存されないかを決定することができます。(ii) この JSON メタデータ

ファイルの場所を指定することができます。メタデータファイルを XMLSpy の複数のインスタンス全体の関連したファイル上に保管されているフィルターとフォーミュラに後で適用するために使用され、このためにフィルターとフォーミュラにポータビリティを提供します。
- JSON グリッドビュー[625]: コンテキストメニューは (i) 未選択のすべてのコンポーネントを折りたたむコマンド (ii) 任意で (コメントとして保管されている) フォーミュラを含むドキュメントから全てのコマンドを削除するコマンドを提供します。
- JSON グリッドビュー[600]: 改善された検索機能
- インライン XBRL 空白スペースの処理を指定する設定[1485]。
- コード生成の新しいメソッド[1019]。

バージョン 2020

- JSON グリッドビュー[600] は追加の編集機能を提供します: (I) 型の自動検知。(ii) セル内のコマンド。(iii) ビューの変更のための XQuery (iv) JSON データからの追加出力の生成のための XQuery フォーミュラ。
- JSON ラインと JSON コメント[594] のための検証と編集サポート
- JSON グリッドビュー内では イメージを表示し[622] また チャートを作成し表示することができます[623]。
- オプションダイアログ[1454] (「ツール | オプション」) はテキストビュー内の JSON グリッドビュー とテキストビューの内の JSON ドキュメントの整形式 ための設定を提供します。
- XULE バリデーターと XULE プロセッサーが追加されました。XULE は XBRL インスタンスドキュメントのクエリ言語です。新規の XULE 機能の概要のために セクション XBRL > XULE[821] を参照してください。
- XULE ウィンドウ[824] によりインタラクティブに XBRL インスタンスドキュメントをクエリできるようになりました。
- XULE ドキュメントの編集のためのヘルプ[821].
- 以前は各ドキュメントを個別に処理していましたが 複数のインライン XBRL ドキュメント[766] をセットとして処理することができるようになりました。
- XPath/XQuery ウィンドウ[506] の式エバリュエーターと式ビルダーは再デザインされ使用が簡単になりました。
- XPath/XQuery ウィンドウ[113] の XPath デバッガー[514] 機能に新規機能が搭載されました (i) ウォッチ式 (ii) より良い概要を表示する改善されたインターフェイス (iii) 更にパワフルな分析機能。この結果、XPath/XQuery 式を XML と JSON ファイルに対して更に簡単にテストしデバッグできるようになりました。
- 以前は異なるファイル型を持つファイルのデフォルトの編集ビューは オプションダイアログ[1458] 内で選択されていました。デフォルトの編集ビューは更に 編集ビュー自身内で直接選択できるようになりました[126]。
- Visual Studio[1005] 内での XMLSpy の統合へのサポートが Visual Studio 2019 に拡張されました。
- Eclipse[1010] 内での XMLSpy の統合へのサポートが Eclipse 4.11 と 4.12 に拡張されました。

1.1.2 バージョン 2019

バージョン 2019 リリース 3

XMLSpy バージョン 2019r3 にて新たに追加された機能は以下のとおりです。
- 分割ビュー[146] で拡張されたテキストビュー。これにより、ドキュメントの他の部分を参照する単一ビューをスクロールする代わりに、長いドキュメントの異なる部分を横に並べる編集中のドキュメントに 2 つのビューを提供します。
- 新しい JSON スキーマバージョンへのサポート。draft-06 と draft-07。ドキュメンテーションの JSON セクション[587] を参照してください。
- オプションダイアログ[1454] (「ツール | オプション」) は自動的に検知されたパスをオーバーライドする Java 仮想マシンを設定するオプションを提供します。
- Windows Server 2019 のためのサポート。
- Eclipse[1010] へのサポートが Eclipse 4.9 と 4.10 に拡張されました。

バージョン 2019

XMLSpy バージョン 2019 にて新たに追加された機能は以下のとおりです。

- データベースへのサポートは以下のように拡張されました Firebird 3.0、IBM Informix 12.10、MariaDB 10.3、Microsoft SQL Server 2017、MySQL 8.0、PostgreSQL 10。
- Eclipse 4.8 との統合
- Generic Preferred Label 1.0 Recommendation により検証リレーションシップなどのXBRL リソースのために指定される優先されるラベル[763] が有効化されます。
- XBRL 型付けされているドメインの定義[764] が詳細入力ヘルパーの個別の型付けされているドメインタブ内に表示されます。
- XBRL インスタンスドキュメント内の重複ファクトを検知するための[1408]新規のXBRL メニューコマンド。
- XBRL 処理オプションダイアログ[1485] には以下のオプションが含まれています: (i) 重複ファクトの無視。および(ii) テーブルの生成。
- オプションダイアログ(「ツール | オプション」) のXBRL セクションによりXBRL インスタンスドキュメントのための検証オプション[1485] を含むタブが提供されます: (i) Dimensions 1.0 とUnits Registry 1.0 準拠、(ii) ファクトの重複、および(iii) 計算から結果する不整合性。
- XBRL 要素のカスタム属性も詳細入力ヘルパー[253] 内で表示されるようになりました。

1.1.3 バージョン 2018

バージョン 2018 リリース 2

XMLSpy バージョン 2018r2 にて新たに追加された機能は以下のとおりです。

- WADL ファイルからリクエストをインポートすることが可能なことに加え、HTTP 出力ウィンドウ[711] 内では、WSDL 1.1 または WSDL 2.0 ファイルからHTTP リクエストをインポート[716] することができます。
- JSON のためのXPath/XQuery 3.1 のサポートがJSON 変換とJSON クエリ[666] 内で実装されました: (i) XPath/XQuery 出力ウィンドウ[113] が拡張され、JSON ドキュメントをクエリできるようになりました。(ii) XSLT 3.0 とXQuery 3.0 ドキュメントを使用してJSON ドキュメントが変換可能になりました[666]。(iii) XSLT とXQuery デバッガー[467] が拡張されJSON ファイルをデバッグできるようになりました。
- テキストビュー内のXSLT 3.0 値[429] のためにXPath インテリジェントな編集のサポートが追加されました。
- テキストビューの編集が拡張され(i) 構文に関連下テキストの書式オプション[1470] (表示される空白など) が使用できるようになり、(ii) 多くの追加言語)に対して適用できるようになりました。
- 次のファイル型のためのサポートの編集機能が追加されました CSS、Python、C-スタイル言語、オプションダイアログ[1454] のファイルタイプセクション[1458] 内で設定することのできる言語のファイル型への準拠。
- オプションダイアログ[1454] (「ツール | オプション」) がアプリケーションオプションをよりよく確認し、ユーザーフレンドリー性を向上するために編成されました。
- ネットワークプロキシ設定[1490] を使用することのできる複数のオプションを使用してカスタマイズできるようになりました。
- XMLSpy はXBRL International によりXBRL 認定を受けました。XBRL 証明書に関する説明は、XBRL ソフトウェア証明書を参照してください。XMLSpy 内でのXBRL との作業に関しての詳細は、ドキュメントのXBRL セクション[721] を参照してください。

バージョン 2018

XMLSpy バージョン 2018 にて新たに追加された機能は以下のとおりです。

- HTTP リクエストを作成し、ターゲット URL に送信し、レスポンスを受け取りチェックすることのできる新規 HTTP 出力ウィンドウ[711]。
- ファイル比較機能[1423] が強化され、3つのファイルを同時に単一のビューで比較できるようになりました。
- オプションダイアログ[1454] が再編成され、オプションの異なるグループがツリー構造に整理されました。これにより、異なるグループの概要がわかりやすくなりました。
- XBRL バージョン管理 1.0 のサポート
- XML 署名 1.1[370] のサポート
- Sybase 16、PostgreSQL 9.6、MySQL 5.7 のサポート
- Eclipse 4.7 との統合
- FOP 2.2 のサポート

1.1.4 バージョン 2017

バージョン 2017 リリース 3

XMLSpy バージョン 2017r3 にて新たに追加された機能は以下のとおりです。

- バックマッピング[1285] 機能が有効化されていると、XSLT 変換とXQuery 実行が行われ、結果ドキュメントが元の XSLT+XML またはXQuery+XML ドキュメントにバックマッピングされます。結果ドキュメント内のノードをクリックすると、XSLT 命令 とその特定の結果フラグメントを生成したXML ソースデータ また、XSLT 命令、または、XML データノードをクリックすると、他の2 つのドキュメント内の対応するノードがハイライトされます。
- JSON5 のサポート[587]: この機能には自動補完、JSON スキーマに対する検証、XML からの変換、XML への変換が含まれます。
- JSON スキーマのサポートの拡張[632]:標準のJSON コンテナーにより名前がつけられた定義に加え、`resourceDefinitions` という名前のコンテナーからの定義が追加されました。
- テキストビュー: (XML、JSON、などの) 異なるドキュメントの種類の 検証エラーは使用できる場合、番号付けられたマージン、スクロールバーにより示されており、エラーを素早くフィックスします。
- テキストビュー: ナビゲーション[137] と検索と置換[140] 関数が強化されました。

バージョン 2017

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

- テキストビュー: テキストのハイライト[137] は、現在選択されているテキストの一致を、一致に関するメタデータと共に表示します。
- テキストビュー: スクロールバー[139] は、ドキュメントのサイズ、ドキュメントのサイズに相対した現在の位置に関する情報を含んでいます。
- テキストビュー: 検索と置き換え関数[140] への追加関数と簡単な使用方法
- Avro データシリアル化[675]: Avro データ構造とAvro スキーマのためのインテリジェントな編集のサポートです。JSON とAvro スキーマ内のAvro データ構造の検証を行います。
- Avro ビュー[681]: 検索することのできるグリッドフォーマットのAvro バイナリのデータブロックを表示するAvro バイナリ専用のグリッドビューです。
- インラインのXBRL ドキュメントを[1410] XBRL インスタンスドキュメントに変換することが可能になりました。
- XMLSpy と共にダウンロードすることができるXBRL タクソノミパッケージ[738] によりXMLSpy は、パッケージのオフラインリソースを検索し使用できるようになりました。
- RaptorXML Server[34] の追加機能

1.2 Windows ファイルパス

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

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

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

Windows 7/8/10	C:\Users\<username>\Documents

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

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

メモ プラットフォーム更新済みの Windows Server 2008 R2 SP1 または以降 上でも XMLSpy はサポートされています。

1.3 RaptorXML Server について

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

RaptorXML には2つのエディションがあります:

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

RaptorXML をコマンドラインから実行することができ、COM、Java、.NET, および Python のためのインターフェイスがあります。XMLSpy インターフェイス内から Raptor サーバーを実行することもできます。

Altova W eb サイト: XML 検証サーバー、XML バリデーター

2 XMLSpy チュートリアル

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

このチュートリアルは以下のように構成されます:

- XMLSpy インターフェイス[36]。アプリケーションのグラフィカルユーザーインターフェイス(GUI) を詳しく理解することができます。
- XML スキーマの作成[46]。XMLSpy の直感的なスキーマビューにて XML スキーマを作成する方法、ドラッグアンドドロップにより複雑なコンテンツモデルを作成する方法、そしてスキーマの設定方法を学習します。
- スキーマビューの機能を使用[61]。複合型ならびに単純型、グローバルな要素参照、そして属性の列挙の作成方法を学習します。
- スキーマビューを使った スキーマのナビゲーション方法[73]、スキーマのナビゲーション方法[73]を学習します。
- XML ドキュメントの作成[79]。スキーマを XML ドキュメントに割り当て、グリッドビューやテキストビューを使ってXML ドキュメントを編集する方法、そして XMLSpy に組み込まれているバリデーターを使用して、XML ドキュメントを検証する方法を学習します。
- XSLT スタイルシートを使用して XML ファイルを変換する[99]。XSLT ファイルを割り当て、XMLSpy に組み込まれたXSLT エンジンを使用することで変換を行います。
- XMLSpy プロジェクトにて作業する[103]ことにより、XML ドキュメントを簡単に整理することができます。

インストールと設定

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

チュートリアルファイル

チュートリアル ファイルはアプリケーション フォルダにあります:

C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples¥Tutorial

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

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

2.1 XMLSpy インターフェイス

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

XMLSpy の開始

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

インターフェイスの概要

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

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

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

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

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

以下のビューを使用することができます: テキストビュー[130]、グリッドビュー[149]、スキーマビュー[159]、W SDL ビュー[231]、XBRL ビュー[243]、Authentic View- ビュー[526]、アーカイブビュー[834]、およびブラウザービュー[256]。

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

2.1.1 表示

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

ドキュメントビューの切り替え

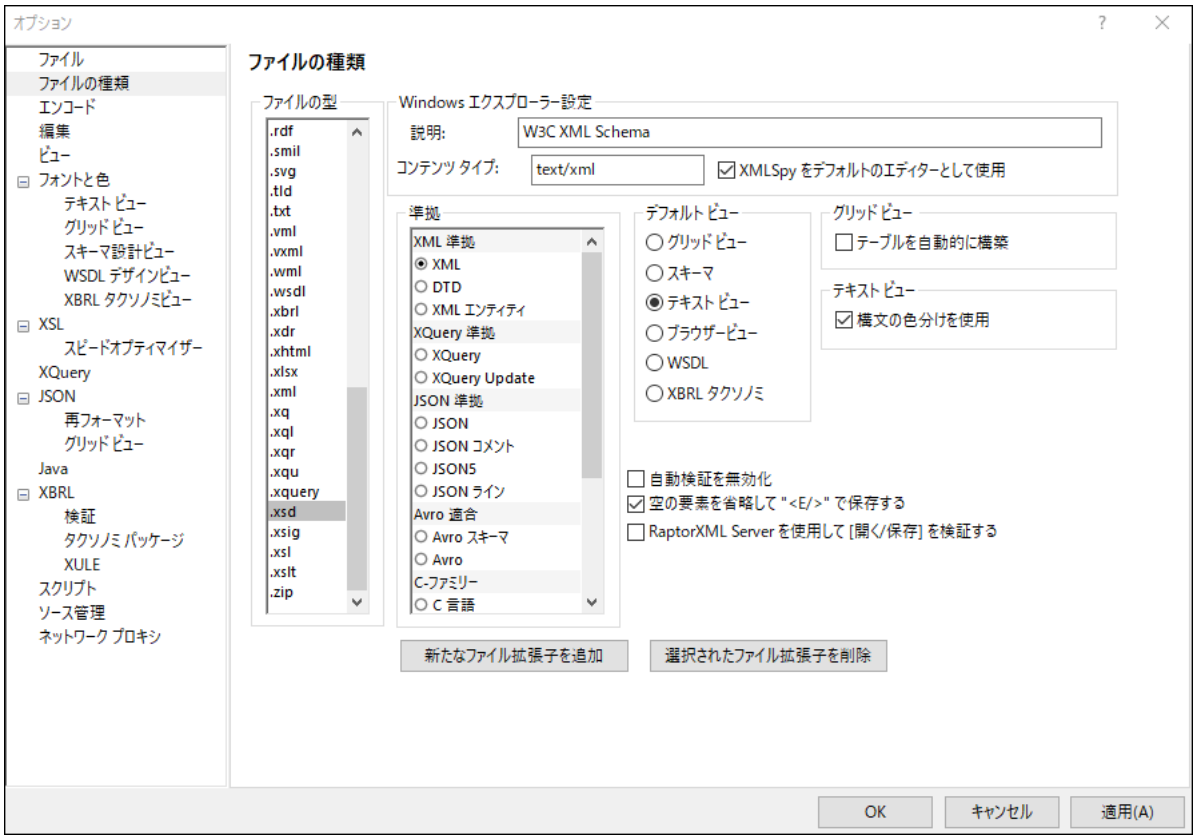
ドキュメントを開くと、そのドキュメントの型のために設定されたデフォルトのビューで開かれます。ドキュメントを以下の方法で開きます:

1. 「ファイル | 開く」コマンドをクリックします。
2. C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021\Examples\Tutorial フォルダーにあるファイル Address.xsd を参照し選択します、「開く」をクリックするとファイルがスキーマビューで開かれます。
3. メインウィンドウの下にあるビュータブをクリックして多種のビュー（テキストビュー、グリッドビュー、など）を切り替えることができます。XML スキーマドキュメントはテキストビュー、グリッドビュー、スキーマビューおよびブラウザービューで表示されます。
4. 「表示」メニューのオプションからビューのオプションを選択してビューを変更することができます。「表示」メニューコマンドを使用して AddressFirst.xsd ドキュメントのビューを切り替えてください。
5. 「ファイル | 閉じる」を使用してドキュメントを閉じてください。

ドキュメントの型のデフォルトのビューの変更

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

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



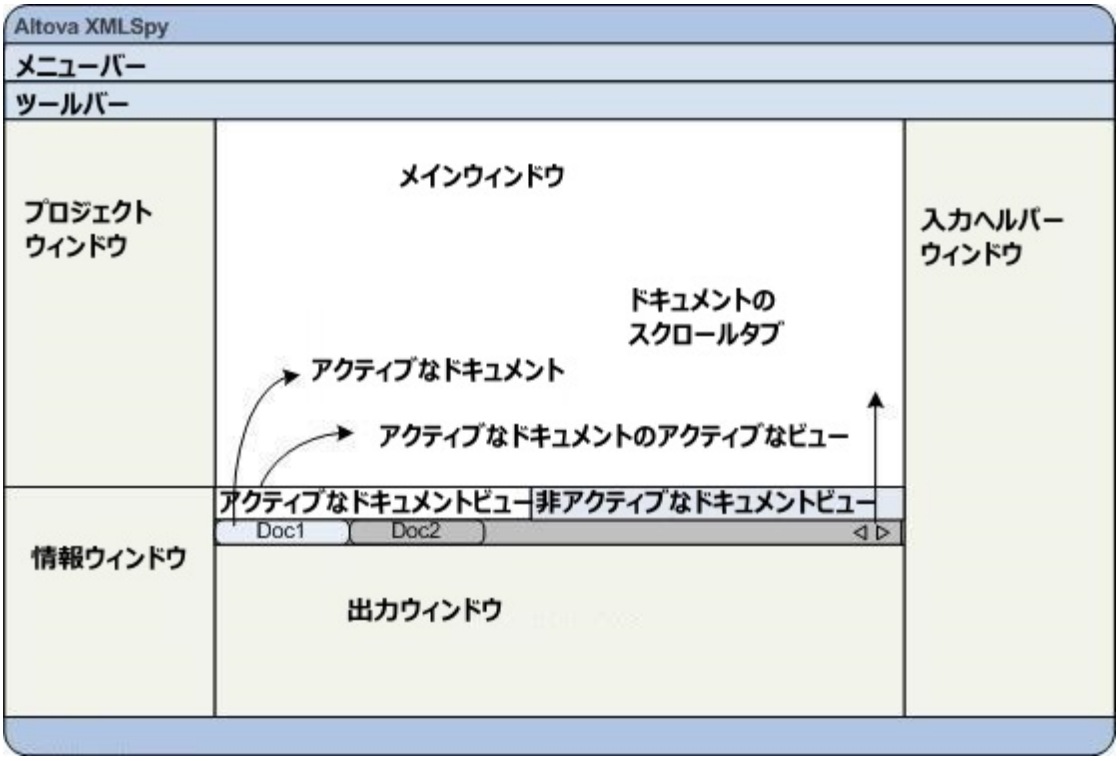
4. 「OK」をクリックします。
5. 「ファイル | 開く」コマンドをクリックして、`AddressFirst.xsd` ファイルを開きます。ファイルはテキストビューで開かれます。
6. スキーマビューに切り替えて、ファイルをこのビューで表示し、ファイルを「ファイル | 閉じる」を使用して閉じます。
7. 「ツール | オプション」を使用して、オプションダイアログに戻り、ファイル型のセクションで `.xsd` ファイルのデフォルトのビューをスキーマビューに戻します。

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

2.1.2 ウィンドウ

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

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



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

ウィンドウグループの表示のオンオフの切り替え

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

C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021\Examples\Tutorial フォルダー内のXML ファイルを開き、基本のコマンドの使用を練習し、コマンドの作動方法に慣れ親しんでください。ウィンドウグループの表示と非表示に関しての詳細は「XMLSpy インターフェイス」108 のセクションを参照してください。

保存された状態とバックアップされた状態

デフォルトではXMLSpy は5秒の間隔で保存されていないドキュメントをバックアップします。メインウィンドウの下の各ファイルタブはファイルの保存/未保存の状態とバックアップの状態を表示するシンボルを使用して情報を提供します。作業中に定期的にこれらのインジケーターが表示されるためその意味を確認してください。これらのインジケーターの情報に関する情報に関してはファイルの自動的なバックアップ128 を参照してください。

スクリーン上でウィンドウを移動する

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

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

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

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

スキーマビュー内で、C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント\Altova\XMLSpy2021\Examples\Tutorial フォルダーから XML スキーマファイルを移動させる練習をするには、上記の方法でインターフェイス内でのウィンドウの移動に慣れ親しんでください。

2.1.3 メニューとツールバー

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

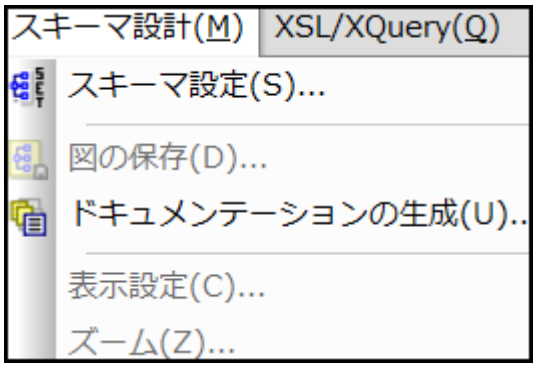
メニュー

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

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

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

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

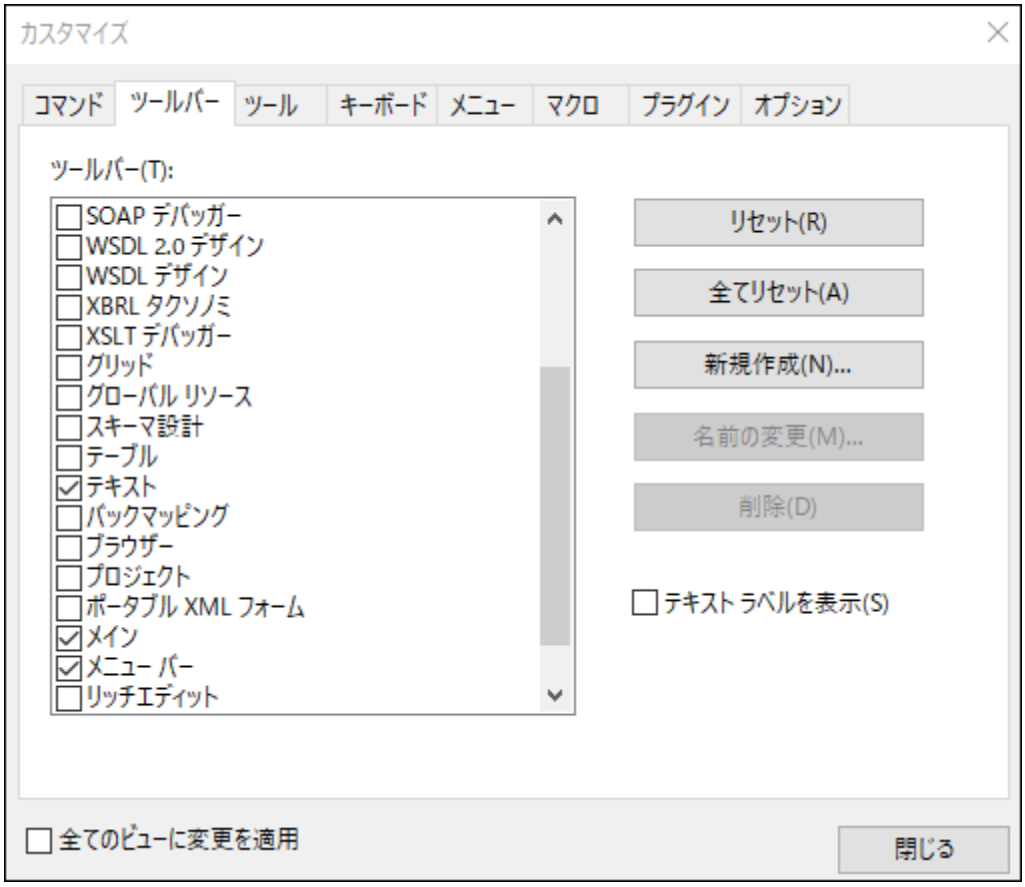


2. メニューの外側の空白をクリックして、メニューを消します。要素コンポーネントの左横にある**図の表示** アイコン をクリックします。これによりスキーマビューのコンポーネントモデルビューに移動します（スキーマビューの2 番目のビュー; 1 番目のビューはスキーマ概要です）。スキーマデザインメニューを確認すると、「**図の保存**」、「**表示設定**」、および「**ズーム**」コマンドが有効化されていることが確認できます。これらのコマンドはスキーマビューのコンテンツモデルビュー内のみで有効化されますが、スキーマビューのスキーマ概要またはその他のビューでは有効化されません。 XML スキーマ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

全てのメニューコマンドの詳細に関しては、ユーザーマニュアルの「ユーザーレファレンス」のセクション[1127]を参照してください。

ツールバー

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



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

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

終了すると `AddressFirst.xsd` ファイルを閉じます。

2.1.4 テキストビューの設定

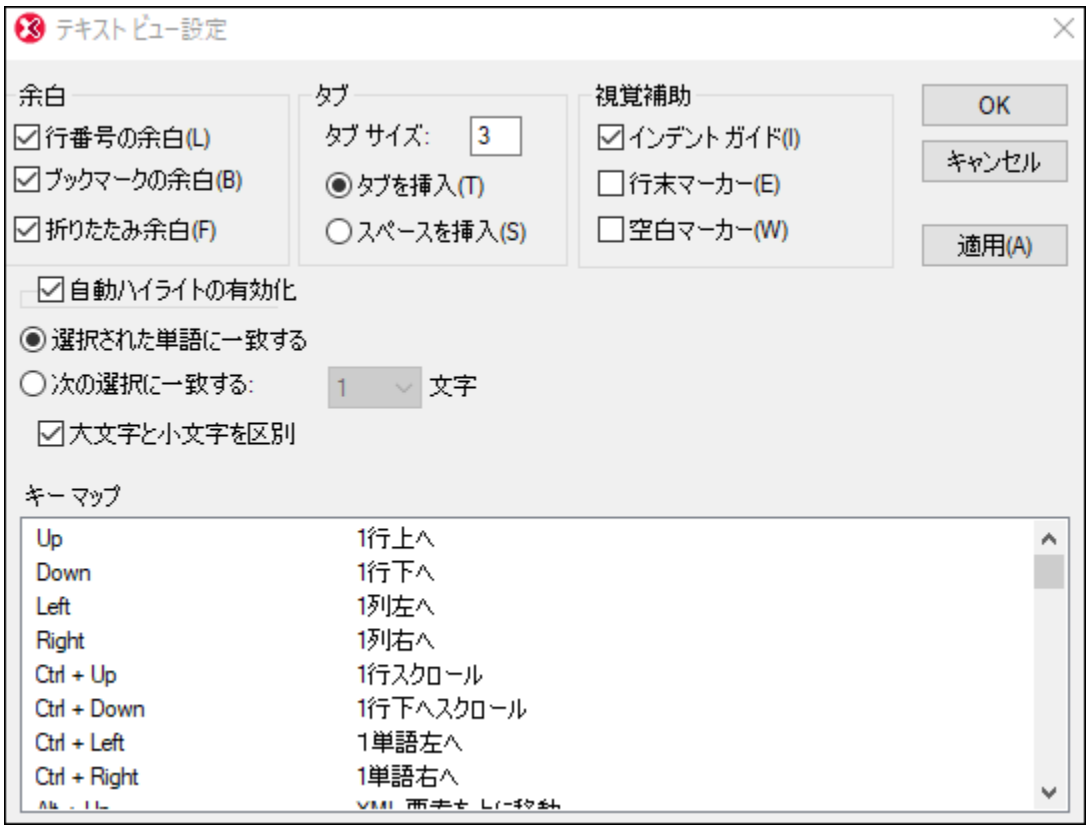
このセクションでは、再フォーマットされたドキュメントの設定および編集中のブックマークの使用方法を学習します。ドキュメントが再フォーマットされると、テキストビューで表示され、XML 階層の各下のレベルが前のレベルに比較してインデントされて表示されます (*下のスクリーンショット参照*)。ブックマークにより、ドキュメントの位置をマークし、戻る場所を素早く検索することができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<Company>
    <Address xsi:type="US-Address">
        <Name>US dependency</Name>
        <Street>Noble Ave.</Street>
        <City>Dallas</City>
        <Zip>04812</Zip>
        <State>Texas</State>
    </Address>
    <Person Manager="true" Degree="BA" Programmer="false"
        <First>Fred</First>
        <Last>Smith</Last>
        <PhoneExt>22</PhoneExt>
        <Email>Smith@work.com</Email>
    </Person>
</Company>
```

整形出力

再フォーマットには2 つのステップがあります: (i) 整形出力の設定 (ii) 整形出力 を適用します。

1. C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント\Altova\XMLSpy2021\Examples\Tutorial フォルダー内のファイルCompanyFirst.xml を開きます。
2. テキストビューがXML ドキュメントのデフォルトの開始ビューではない場合テキストビューに切り替えます。
3. メニューコマンド [表示 | テキストビュー設定] を選択してテキストビュー設定ダイアログを開きます (*下のスクリーンショット*)。



4. タブペイン内でタブのサイズを 3 にまで減らします。*タブの挿入*ラジオボタンのデフォルトの選択をそのままにしておいてください。これにより整形出力のインデントがスペースではなく タブになるように設定されます、各タブは、3 つの空白の幅になります。終了すると **[OK]** をクリックします。
5. メニューコマンド [**編集 | 整形出力**]をクリックします。整形出力が適用されます。ドキュメントの表示は新しいタブの値共にリセットされます。
6. テキストビュー設定 ダイアログ（[**表示 | テキストビューの設定**]）を開いて、*視覚補助*ペイン内の行末マーカーを解除します。
7. テキストビュー内で、行末に移動して、行末マーカーを削除することにより、次のラインが上に移動します。
8. グリッドビューに切り替えて、再度 テキストビューに切り替えます。
9. メニューコマンド[**編集 | 整形出力**] をクリックします。ドキュメントは、整形出力され、削除された行末マーカーが回復されます。

メモ オプションダイアログの整形出力セクション内で（「ツール | オプション | 整形出力」1462 ）、*インデントの使用* のチェックボックスのチェックを解除すると、すべての行はインデント無しに再フォーマットされます。

ブックマーク

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

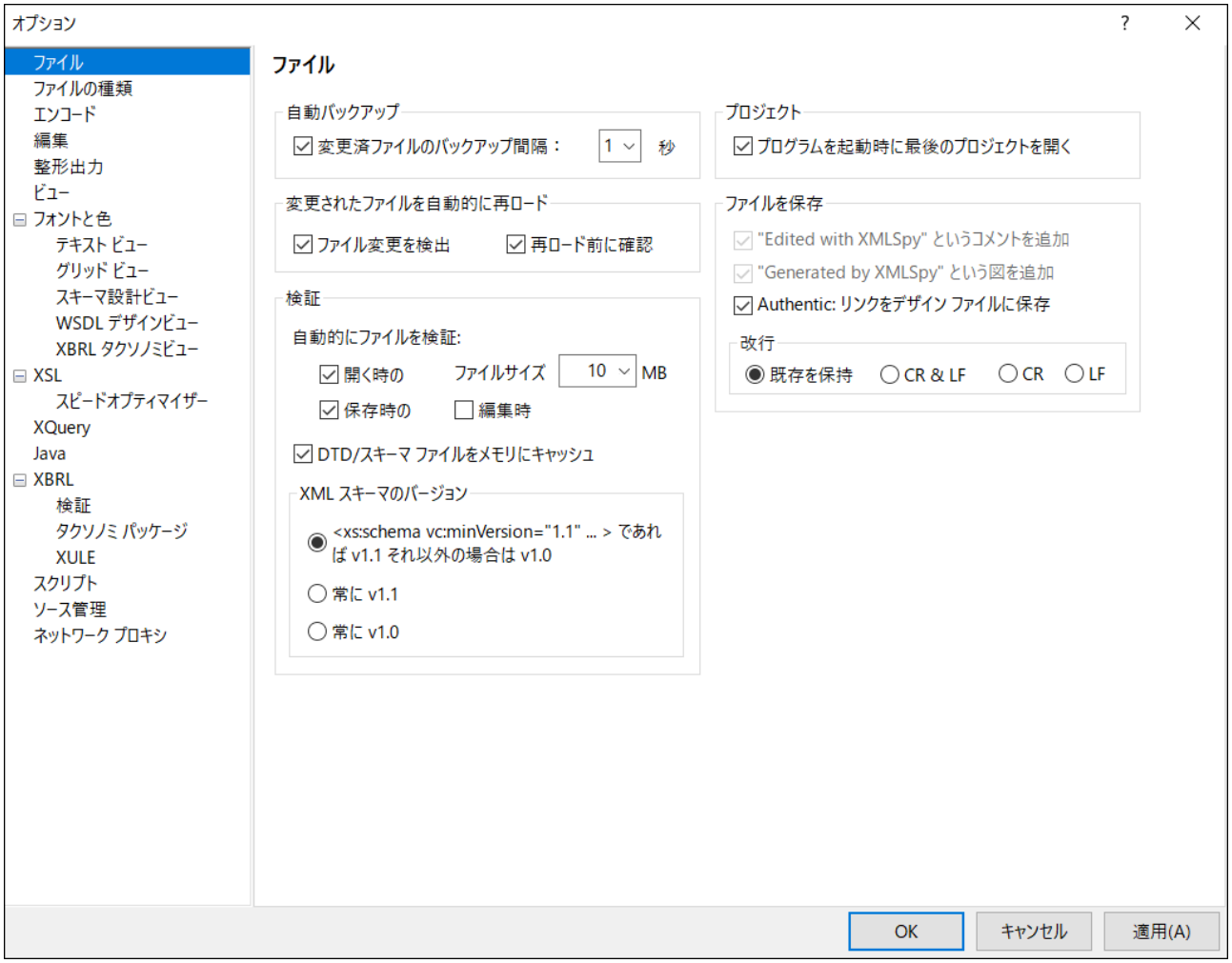
1. テキストビューの設定（「**表示 | テキストビューの設定**」*上のスクリーンショット*）で、*ブックマークマージン*のオプションがマージンペインで選択されていることを確認します。完了すると「OK」をクリックします。
2. ファイル `CompanyFirst.xml` のテキストビューで、*ブックマーク*するラインにカーソルをポイントし、メニューコマンド「**編集 | ブックマークの挿入/削除**」を選択します。ラインは*ブックマーク*され、*ブックマークマージン*内に青い*ブックマーク*で表示されます（*下のスクリーンショット参照*）。
3. *ステップ 2* と同様に、他のラインの*ブックマーク*を作成します。

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Company>
3     <Address xsi:type="US-Address">
4        <Name>US dependency</Name>
5        <Street>Noble Ave.</Street>
6        <City>Dallas</City>
7        <Zip>04812</Zip>
8        <State>Texas</State>
9     </Address>
10    <Person Manager="true" Degree="BA" Programmer="false">
11       <First>Fred</First>
12       <Last>Smith</Last>
13       <PhoneExt>22</PhoneExt>
14       <Email>Smith@work.com</Email>
15    </Person>
16 </Company>
17
```

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

2.1.5 アプリケーションオプション

XMLSpy は機能が豊富なため、作業の多くのアスペクトを大幅に影響する多種のオプションを設定することができます。作業の必要条件と作業スタイルに適するようにアプリケーションの設定を変更し理解することは長期的にとても有利です。これらの設定はメニューコマンドで「ツール | オプション」1454 を使用しオプションダイアログ内でアクセスすることができます（*下のスクリーンショット*）。



設定はセクションごとに整理されており、左側にリストされています。各セクションを確認してどの設定が使用可能かを確かめてください。下のリスト内では、一般的に使用される機能に影響を与える設定の一部に着目してみました。全ての設定についての詳細に関しては[オプションダイアログのドキュメンテーション](1454)に移動してください。設定の変更後「OK」クリックしてレジストリに変更を保存し、ダイアログを閉じます。「適用」ボタンにより現在開かれているドキュメントへの変更を表示することができます。

ファイル
(編集中のファイルの) *自動バックアップ*をオン/オフに切り替えることができます。*編集時の検証*は整形式チェックと検証チェックを入力中に行います。この機能が不便な場合、この機能をオフに切り替え、整形式チェックを実行し、検証チェックを準備ができた時点で行うことができます。

ファイル型
デフォルトのビューをファイル型に設定してください。ドキュメントの各型に応じて最も使用しやすいビューを選択してください。

表示
XMLSpy タイトルバー内ではファイル名またはファイルパス全体が表示されます。メインウィンドウの下のファイルタブ内にマウスをポイントするとファイルパスを確認できることに注意してください。

フォントと色
フォント、サイズ、およびテキストと他のコンポーネントの色をセットアップすることができます。これらは各ビューにより異なる設定です。

2.2 XML スキーマ: 基本

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

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

目的

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

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

基本的なスキーマが完成したら、%XMLSPY%> によるXML スキーマのさらに高度な機能について次のセクション[61]で学習します。その次には%XMLSPY%> のスキーマナビゲーションとドキュメンテーション[73]に関するセクションが続きます。

このセクションで使用されるコマンド

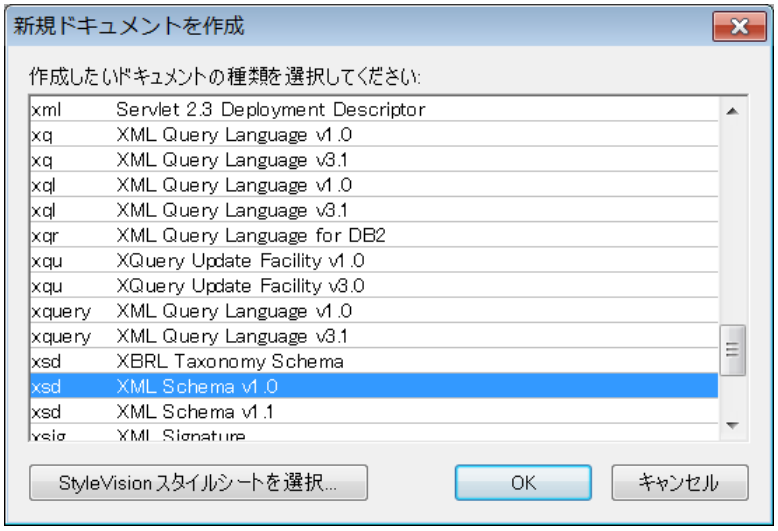
このセクションではスキーマビューだけを使用します。以下のコマンドが使われます:

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

2.2.1 新しい XML スキーマファイルの作成

新しいXML スキーマファイルの作成:

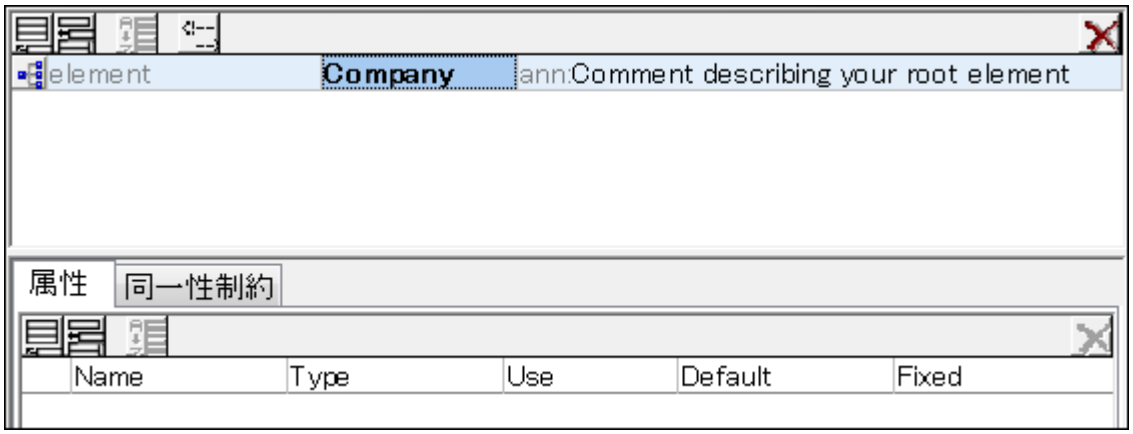
1. メニューオプションから「**ファイル | 新規作成**」を選択します。「**新規ドキュメント作成**」ダイアログが開きます。



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

4. ルート要素を入力するよう求められます。

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



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

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

ファイルタブ内の色付けされたサークルのシンボルはファイルのバックアップの状態を示しています。これらのインジケーターの詳細に関してはファイルの自動的なバックアップ 128 を参照してください。

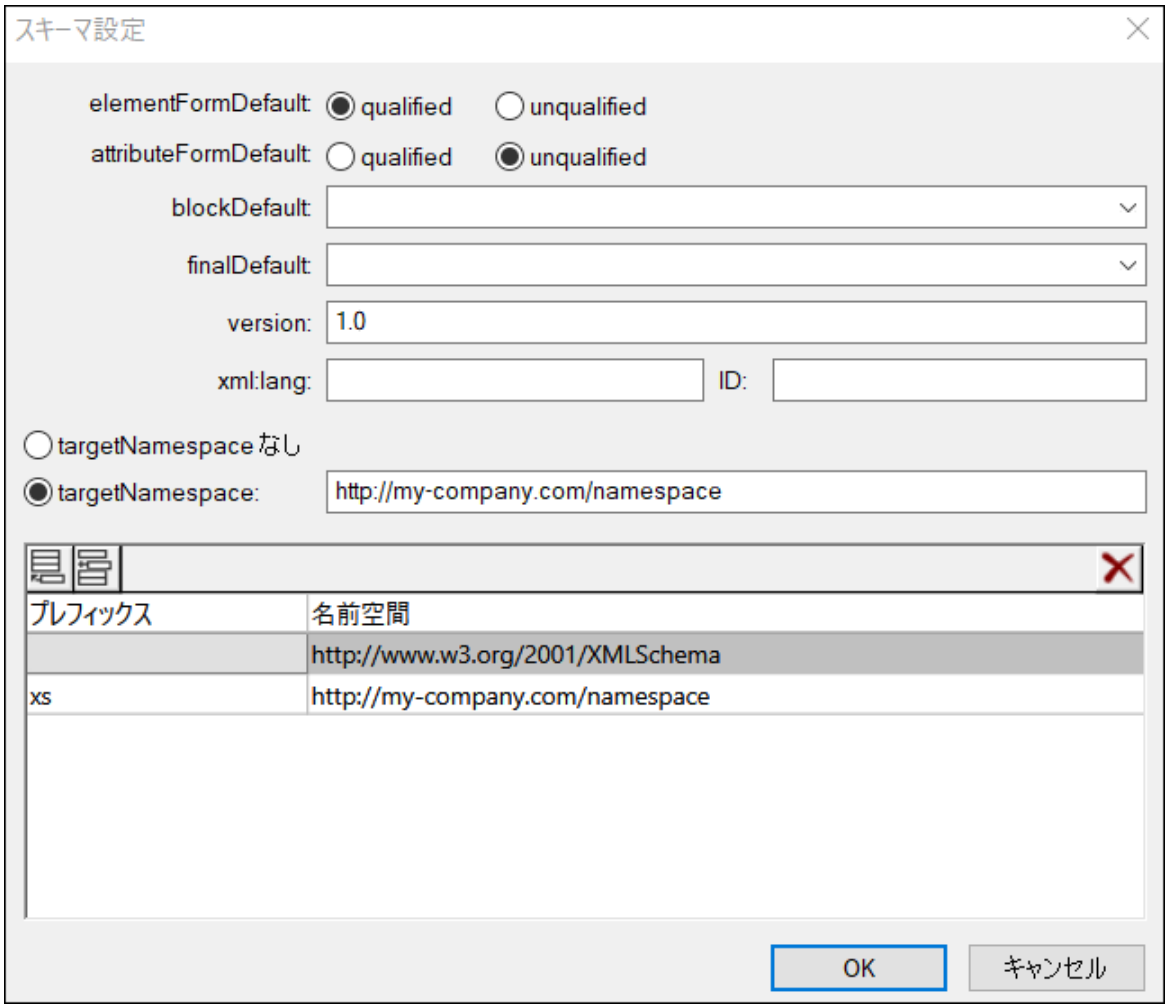
2.2.2 名前空間の定義

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

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

ターゲット名前空間の作成方法:

1. 「**スキーマ設計 | スキーマ設定**」をメニューオプションから選択します。これによりスキーマ設定ダイアログ (*下のスクリーンショット*) が開かれます。



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

注意点:

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

2.2.3 コンテンツモデルの定義

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

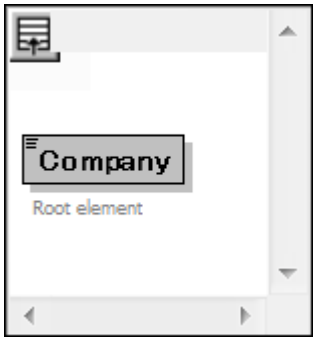
XMLSpy において、グローバルコンポーネントのコンテンツモデルは、スキーマビューのコンテンツモデルビュー内に表示されます。グローバルコンポーネントのコンテンツモデルを編集するには、グローバルコンポーネントの左にあるダイアグラム表示アイコン をクリックします。

このセクションでは、**Company** 要素のコンテンツモデルを作成します。

基本的なコンテンツモデルの作成

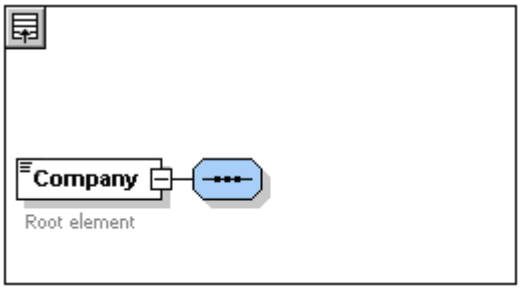
Company 要素のコンテンツモデルを作成するには:

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

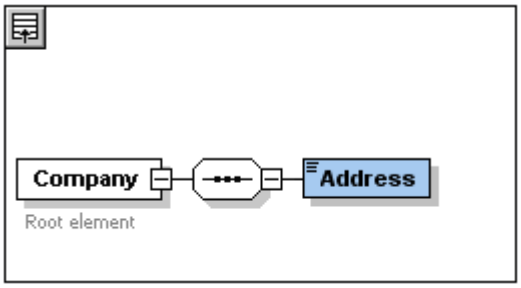


2. コンテンツモデルはコンポジッターとコンポーネントにより構成されます。コンポジッターは2つのコンポーネントの関係を指定します。Company コンテンツモデルでは、子要素を加えるために子コンポジッターを追加する必要があります。コンポジッターを追加するには、Company 要素を右クリックし、コンテキストメニューから「子を追加 | シーケンス」を選択します。（「シーケンス」、「選択」、そして「全て」は、コンテンツモデルで使用される3つのコンポジッターです）

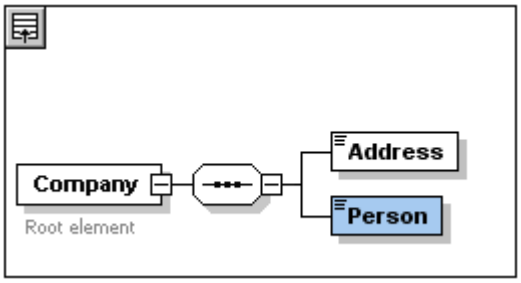
この操作によりシーケンスが指定され、そこから表示されるコンポーネントを定義するシーケンスコンポジッターが挿入されます。



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

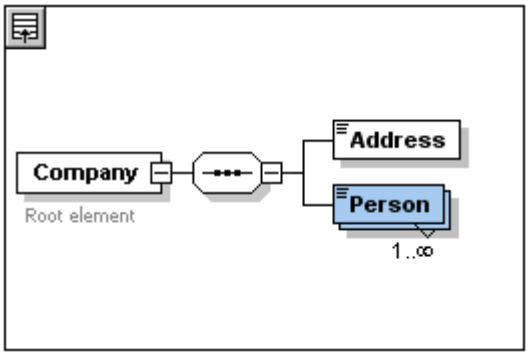


5. シーケンスコンポジッターを再度右クリックして、「子を追加 | 要素」を選択します。新たに作成された要素コンポジットの名前を Person にします。



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

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



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

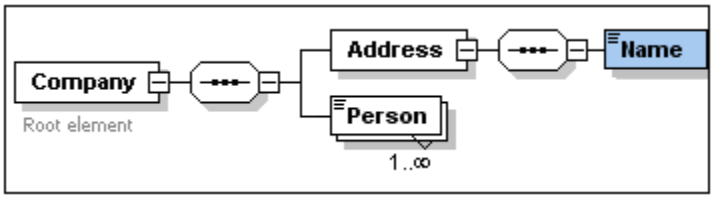
コンテンツモデル構造に新たな階層を追加

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

やり方:

1. Address 要素を右クリックしコンテキストメニューを開き、「子を追加 | シーケンス」を選択します。これでシーケンスコンポジッターが追加されます。

2. シーケンスコンポジッターを右クリックし、「子を追加 | 要素」を選択します。新たに作成された要素コンポーネントの名前を Name に変更します。



複合型、単純型、スキーマデータ型

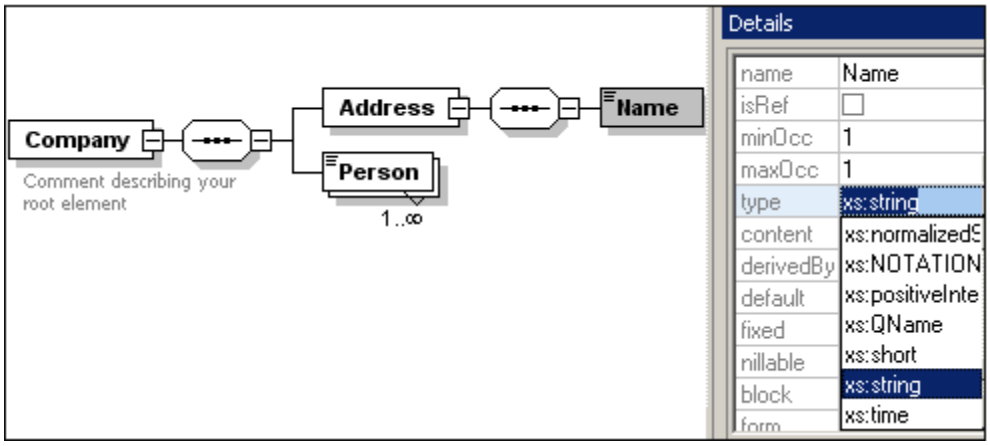
ここまで、要素の型を明示的に定義はしませんでした。テキストタブをクリックして、スキーマをテキストビューに表示させてください（下記参照）。シーケンスコンポジッターが挿入された場所には xs:complexType 内に xs:sequence 要素が挿入されていることが分かると思います。簡単に説明すると、Company そして Address 要素は子要素を持っているため複合型になります。複合型要素は属性または要素を持ちます。

```
<xs:element name="Company">
  <xs:annotation>
    <xs:ドキュメント>Root element</xs:ドキュメント>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Address">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Name"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Person"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

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

Name 要素のデータ型を定義するには:

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



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

Name 要素のテキスト表記は以下のようになります:

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

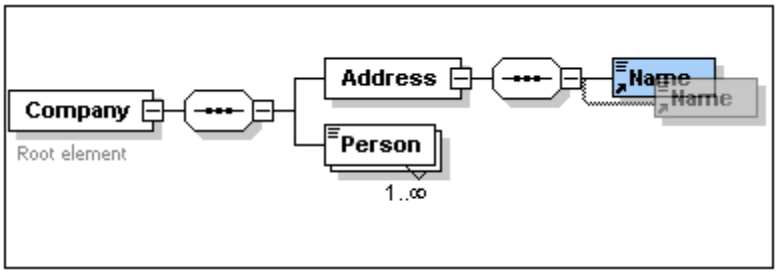
注意: 単純型要素は幾つかあるXMLスキーマデータ型のどれでも取ることができます。全ての場合において、要素ボックスにはテキストコンテンツであることを示すアイコンが表示されます。

2.2.4 ドラッグアンドドロップによる要素の追加

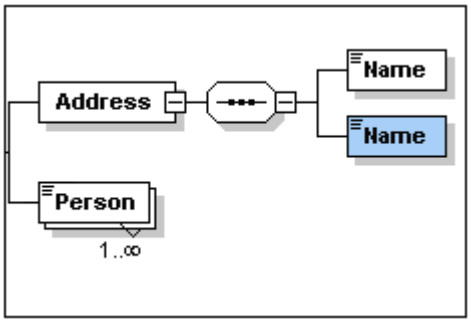
ここまでは、要素やコンポジッターを右クリックし、コンテキストメニューを使って要素を追加してきました。ドラッグアンドドロップを使うとコンテキストメニューより先に素早く要素を追加することができます。このセクションでは、ドラッグアンドドロップを使って Address 要素にさらに子要素を追加して定義を完成させます。

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

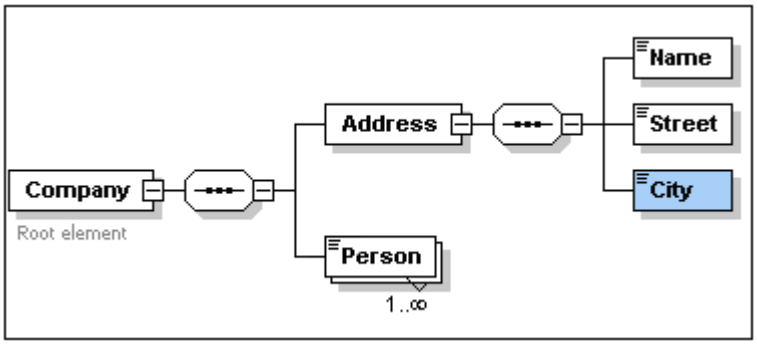
1. Address 要素の Name 要素をクリックして、「Ctrl」キーを押下したまま要素ボックスをマウスでドラッグします。要素ボックスに表示される小さな「+」アイコンはコピーが行われようとしていることを示しています。要素のコピーが接続線と一緒に表示され、要素がどこに作成されるのかを示しています。



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



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



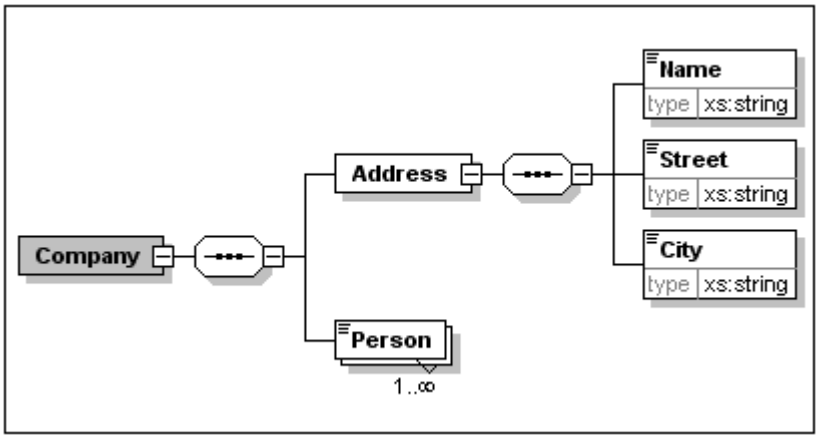
要素は `Name`、`Street`、そして `City` 要素からなるシーケンスを持つことになります。

2.2.5 コンテンツモデルビューの構成

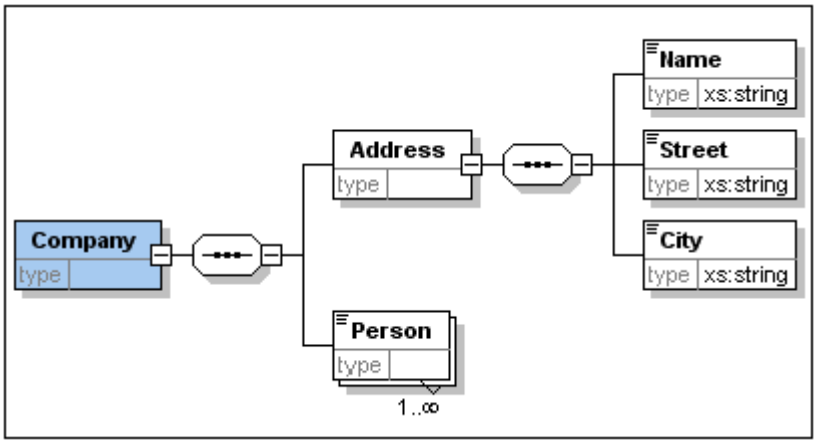
そろそろコンテンツモデルビューの設定を行っても良いでしょう。要素の型が要素ボックス内に表示されるようにビューを構成します。以下の手順を行ってください：

1. （コンテンツモデルビューアイコンをクリックして）コンポーネントのコンテンツモデルビューを選択します。
2. コンテンツモデルビュー内の場合メニューコマンド **スキーマ設計 | ビューの構成** が有効化されています。スキーマ表示構成ダイアログを選択します（*下のスクリーンショット*）。

3. *要素* タブ（*上のスクリーンショット参照*）で **付加** アイコンをクリックして *型（スクリーンショット）* 要素ボックスにこのプロパティ記述子を追加します。
4. *単一ラインの設定* ペイン内で、*値がない場合非表示*を選択します。（例えば、要素が複合型の場合）要素がデータ型を持たない場合これは要素ボックス内のデータ型の説明を非表示にします下の*スクリーンショット*で、型 xs:string の複合型ではなく単純型である Name、Street、および City 要素のために説明行が表示されることご注意してください。これは値の無い行を隠すトグルが選択されているためです。



5. *単一ライン設定* ペインで「*常に行を表示*」ラジオボタンを選選択します。
6. 変更を確認するために「OK」をクリックします。



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

以下の点に注意してください：

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

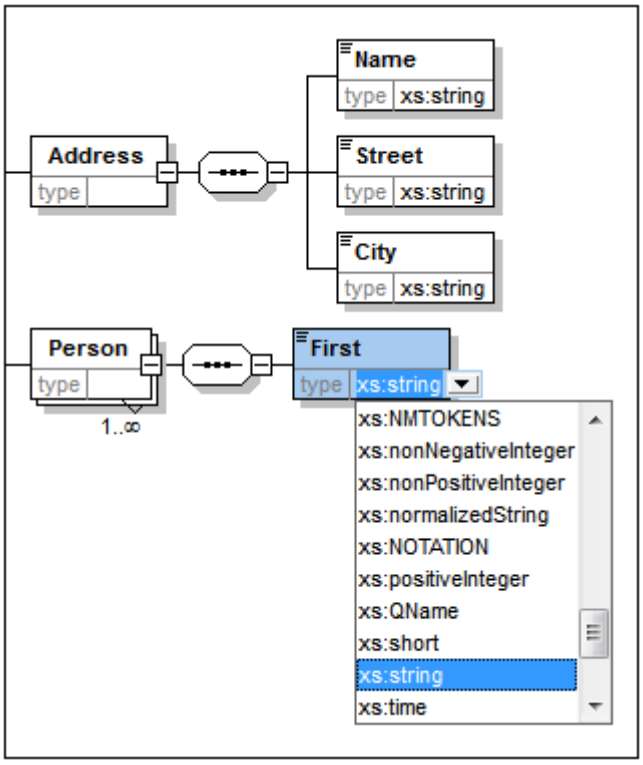
2.2.6 基本的なスキーマの完成

`Address` 要素のコンテンツが定義されています。以下の simpleType 子要素を含む必要のある `Person` 要素のコンテンツを定義する必要があります: `First`、`Last`、`Title`、`PhoneExt`、および `Email`。これらの要素は `Title` 以外は必須で指定された順序で発生する必要があります。型 `xs:integer` であり、2桁の数値に制限されている `PhoneExt` 以外の型 `xs:string` である必要があります。

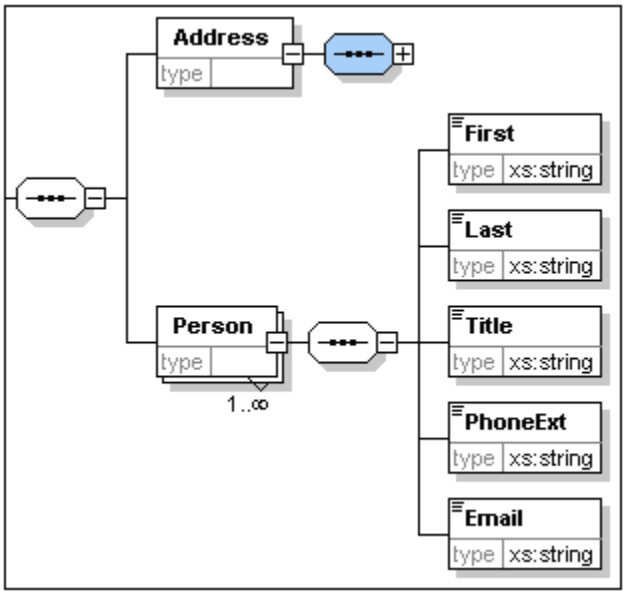
`Person` のコンテンツモデルを作成するには以下を行います:

1. `Person` 要素を右クリックし、コンテキストメニューを表示し「**子を追加 | シーケンス**」を選択します。これによりシーケンスコンポジッターが挿入されます。
2. シーケンス要素を右クリックして「**子の追加 | 要素**」を選択します。

3. **First** を要素名として入力し、「タブ」キーを押します。。これでカーソルがtype フィールドに移動します



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

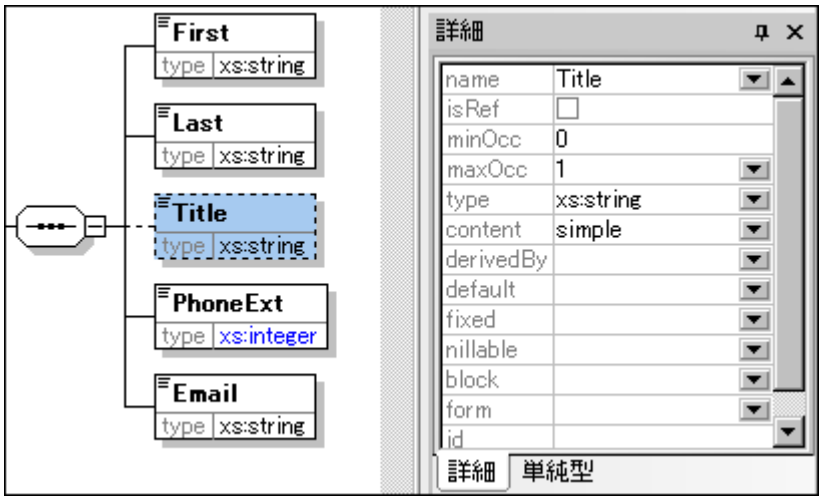


メモ 「Ctrl」キーを押しながら必要な要素をクリックすることで複数の要素を選択することができます。この方法で、例えば複数の要素を一度にコピーすることができます。

(C) 2015-2021 Altova GmbH

要素を省略可能にする

Title要素を右クリックしコンテキストメニューから**省略可能** を選択します。要素ボックスの枠組みが実線から点線に変更されます。これは要素が省略可能であることを示しています。

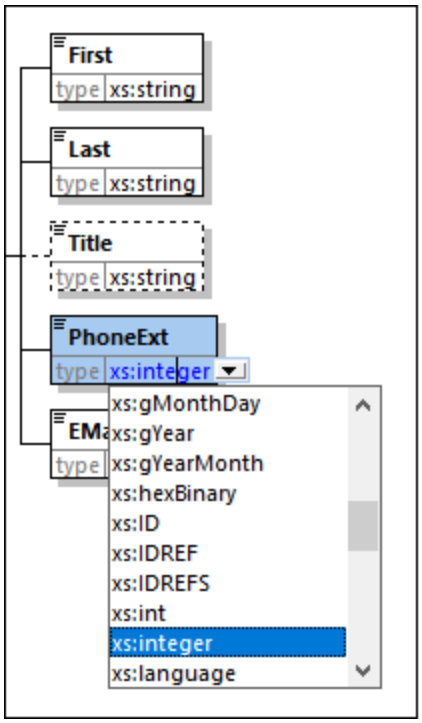


詳細入力ヘルパーにて、**minOcc=0** と**maxOcc=1** と表示され、要素が省略可能であることを示します。コンテキストメニューを使う方法以外にも、**minOcc=0** をセットして要素を省略可能にすることができます。

要素の内容を制限する

PhoneExt 要素が**xs:integer** 型で、最高2桁の数値だけを持てるようにするには以下を行います:

1. **PhoneExt** 要素の型フィールドをダブルクリックして、ドロップダウンリストから**xs:integer** を選択(または入力)します(*下のスクリーンショット参照*)。



2. この時点でファセット入力ヘルパーのアイテムが変化します。ファセット入力ヘルパーにて、**maxIncl** フィールドをダブルクリックして、**99** を入力します。「**Enter**」で確定します。この定義により、**0** から **99** までの内線番号（Phone Extension）の値を持つことができます。
3. メニューオプション「**ファイル | 保存**」を選択し、スキーマに対する変更を保存します。

以下の点に注意してください

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

(C) 2015-2021 Altova GmbH

2.3 XML スキーマ: 高度な機能

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

目的

このセクションでは、以下のことを学習します:

- スキーマの要素としても使用できる複合型ならびに単純型[61]の扱い方
- グローバル要素[68]を作成し、他の要素からの参照を行う
- 属性[70]とその列挙値を含むプロパティーを作成する

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

このセクションで使用されるコマンド

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

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

2.3.1 複合型と単純型を扱う

要素のコンテンツモデルの定義を行ったので、それをスキーマのどこか別の場所で再利用することにします。これは、例えば、米国と英国内のアドレスのために2つのコンテンツモデルを定義する場合に発生する場合があります。2つのアドレスのフォーマットの一部のコンポーネント、例えば、ストリートと都市のコンポーネント、は共通です。しかしながら他のコンポーネントは異なります。この理由のために適切な戦略は各アドレスのコンポーネントモデル(US とUK) 内の共通のコンポーネントを再利用し、それぞれに(US 内のZIP やUK 内の郵便番号などの) 固有のコンポーネントと共に各コンテンツモデルを完了することです。これを行うには、グローバル複合型として(または、各共通のコンポーネントをグローバル要素として) 共通のコンポーネントを作成し、グローバル複合型(またはグローバル要素) を各アドレス型のコンテンツモデル内で再利用します。

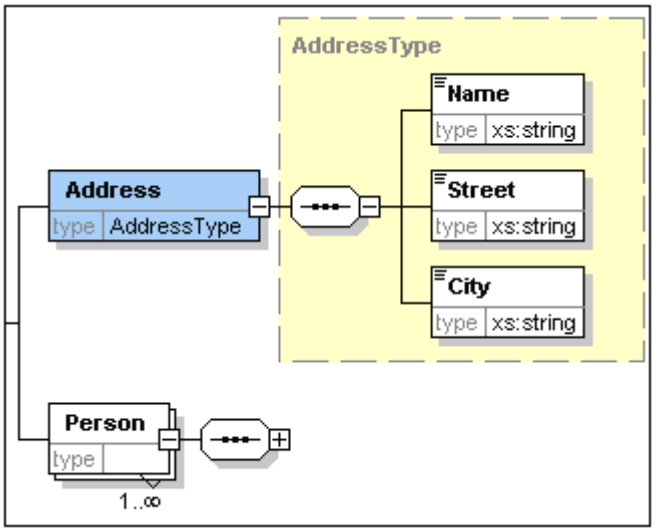
このセクションでは、グローバル複合型と作業します。複合型は単純に他の要素および/または属性を含む要素の型定義です。最初にグローバルレベルで複合型を作成し、コンテンツモデルにインポートし、拡張します。このチュートリアルで後にグローバル要素について学ぶことができます。

グローバル複合型の作成

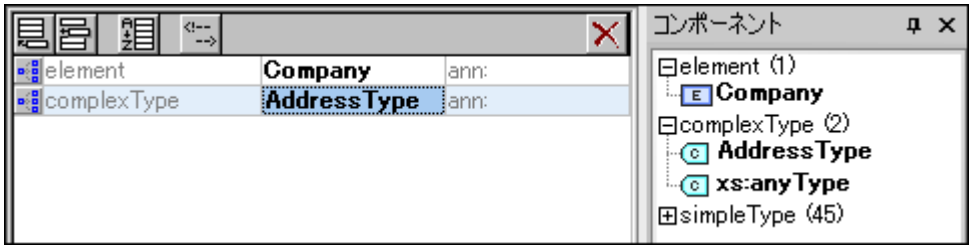
既に定義した(`Name`、`Street`、 `City` 要素を含む) 基本的な `Address` 要素は住所の書式として色々な場所で利用できます。そこで、これら要素の再利用をすることができる複合型を定義することにします。

グローバル複合型を作成するには、以下を行います:

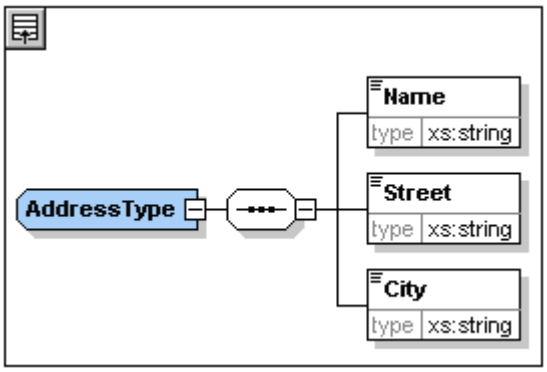
1. コンテンツモデルビューにて **Address** 要素を右クリックする。
2. 現れたコンテキストメニューで「グローバル化 | 複合型」を選択します。**AddressType** という名前のグローバル複合型が作成され、**Company** コンテンツモデルの **Address** 要素がこの型に割り当てられます。**Address** 要素の内容は、黄色のボックスで表示されている AddressType のコンテンツモデルになります。**Address** 要素のデータ型が **AddressType** になっていることに注目してください。



3. 全てのグローバルを表示アイコン をクリックします。メインウィンドウの表示が、スキーマ内の全てのグローバルコンポーネントを表示するスキーマ概要に移動します。
4. 対応するスキーマの構成が表示されるように*展開*アイコンをコンポーネント入力ヘルパー内の **element** と **complexType** エントリのためにクリックします。スキーマ概要は2つのグローバルコンポーネント（**Company** 要素と複合型の **AddressType**）を表示しています。コンポーネント入力ヘルパーにも **AddressType** 複合型は表示されます。



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



(C) 2015-2021 Altova GmbH

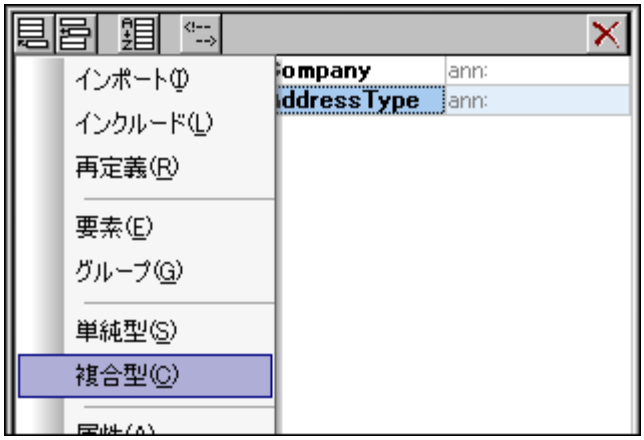
6. 全てのグローバルを表示アイコン をクリックして、スキーマ概要に戻ります。

複合型の定義の拡張

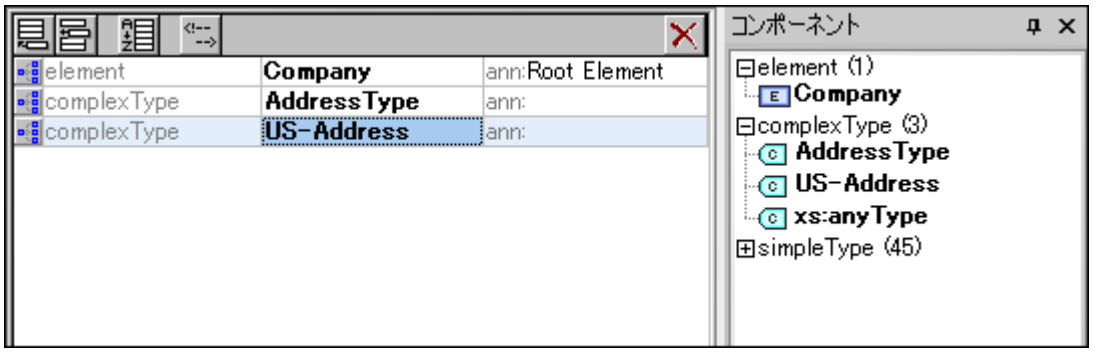
それでは、2つの国に特化した住所を表す為に、グローバル **AddressType** コンポーネントを使用したいと思います。このために、**AddressType** をベースにした新たな複合型を作成し、その定義を拡張します。

以下を行います:

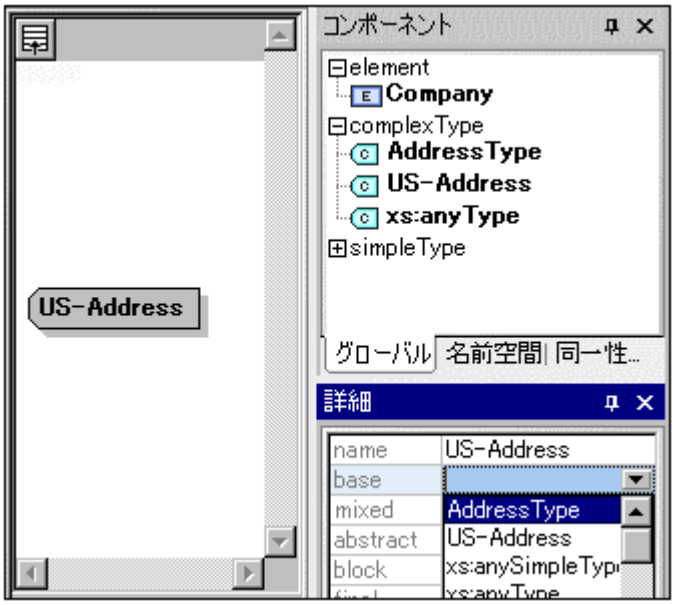
1. スキーマ概要に移動します(コンテンツモデルビューにいる場合、全てのグローバルを表示アイコン をクリックします)。
2. コンポーネントウィンドウの左上にある追加アイコン をクリックします。以下のメニューが開きます:



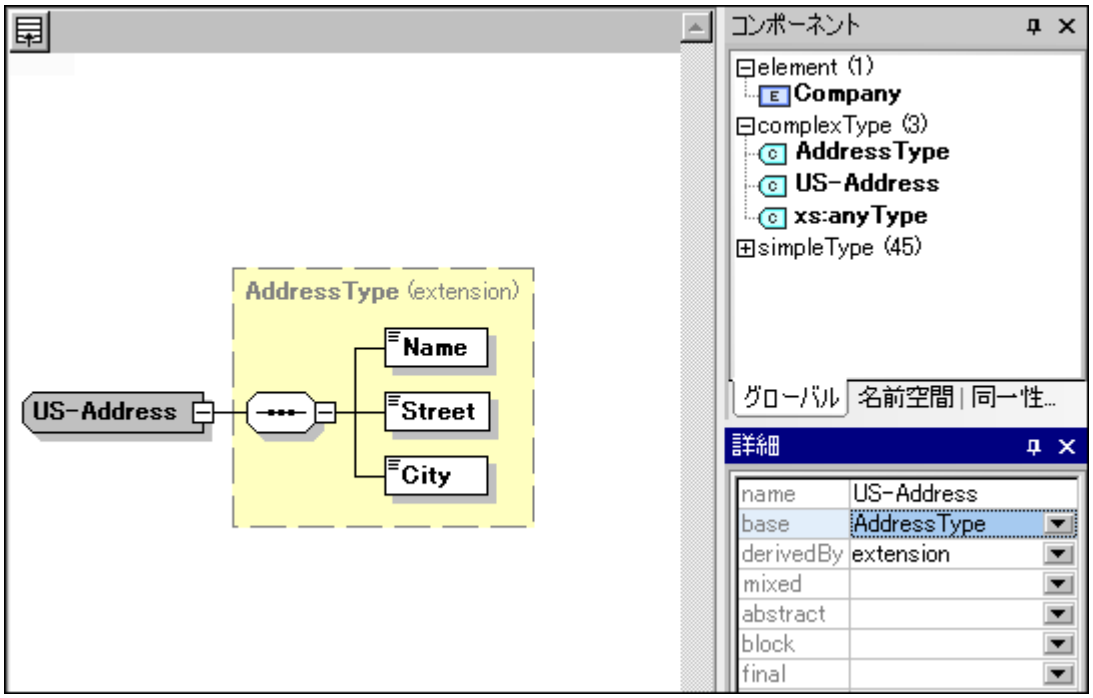
3. メニューから複合型を選択します。コンポーネントリストに新しいレコードが現れ、コンポーネントの名前を入力するようにカーソルがセットされます。
4. **US-Address** と入力し、「Enter」で確定します。(ハイフン(-)を入力する代わりに空白(スペース)を入力した場合、要素名が赤くなり、不正な文字であることが示されます)



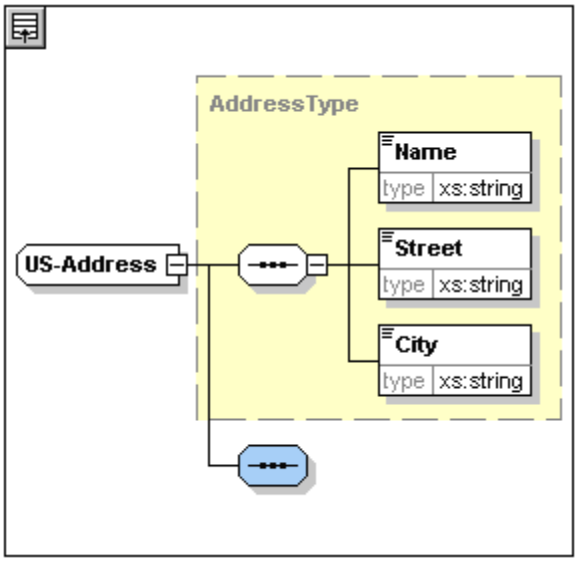
5. **US-Address** のコンテンツモデルビューアイコン をクリックし、新しい複合型のコンテンツモデルを表示します。コンテンツモデルは(以下のスクリーンショットにあるように)空です。
6. 詳細入力ヘルパーにて、base コンボボックスをクリックし、**AddressType** エントリを選択します。



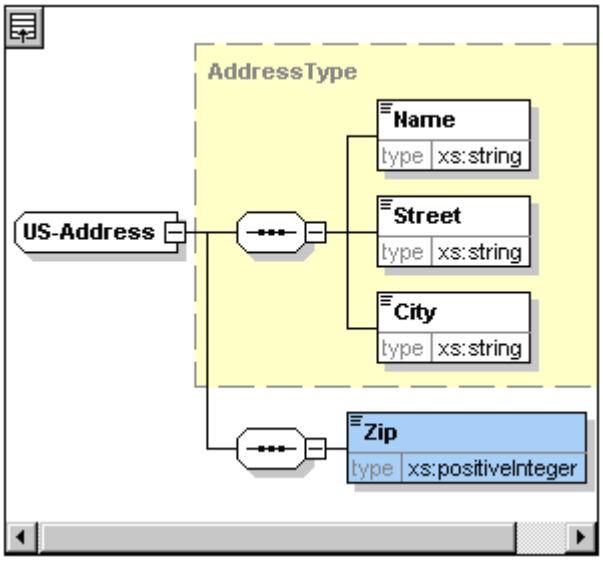
コンテンツモデルビューには **US-Address** のコンテンツモデルとして **AddressType** が表示されます。(*下のスクリーンショット*)。



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



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



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

グローバル単純型

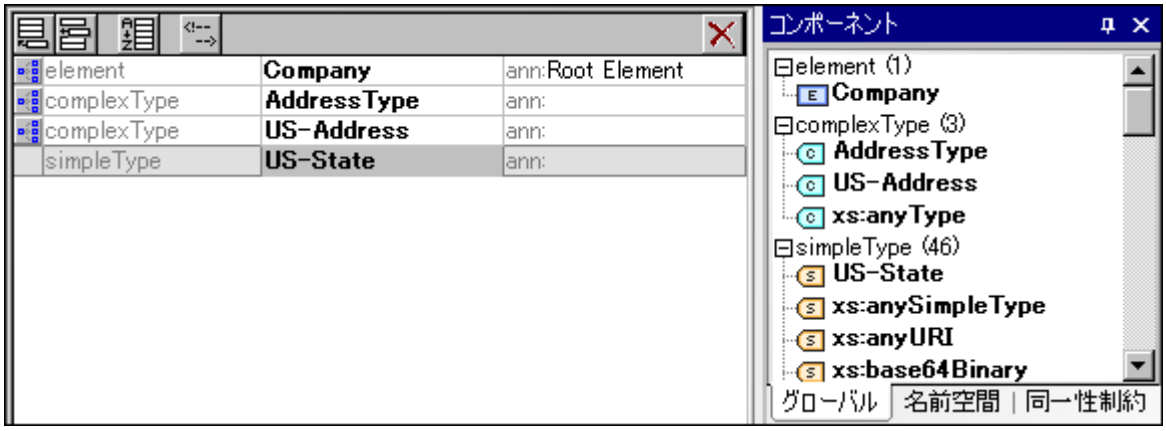
複合型の`US-Address` が複合型の`AddressType` をベースにしたように、単純型をベースにすることもできます。その利点はグローバル複合型の場合と同じで、単純型が再利用されることにあります。単純型を再利用するためには、その単純型をグローバルに定義しなければなりません。このチュートリアルではアメリカの州のコンテンツモデルを単純型として定義することにします。この単純型は他の要素のベースとして使用されることになります。

グローバル単純型の作成

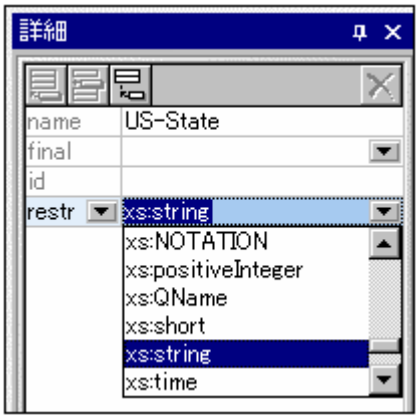
グローバル単純型の作成は、グローバルコンポーネントのリストに新しい単純型を追加し、名前を与え、データ型を定義することから構成されています。

グローバル単純型を作成するには以下をおこないます:

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



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



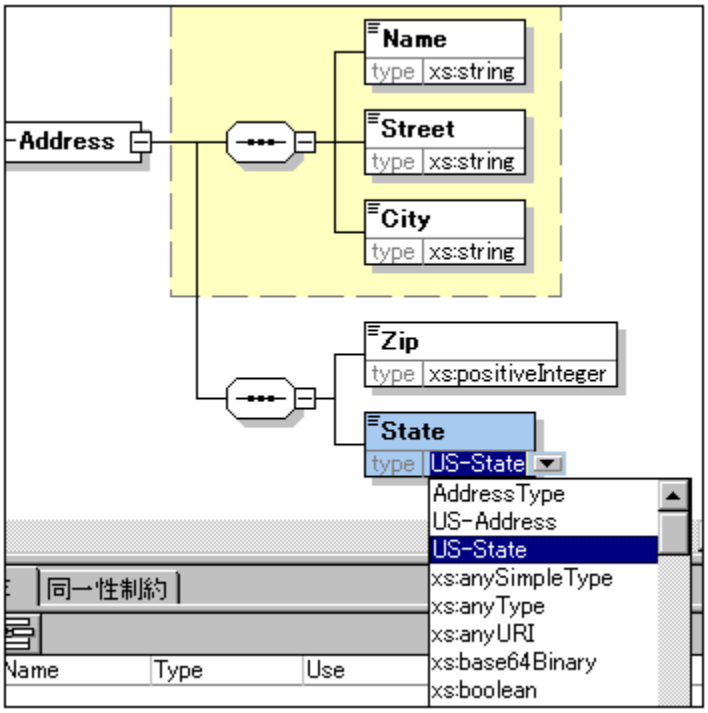
`xs:string` のデータ型である、`US-State` という名前の単純型が作成されます。このグローバルコンポーネントは、`US-Address` のコンテンツモデル内で使用することができます。

コンテンツモデル内でグローバル単純型を使用する

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

次を行います:

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



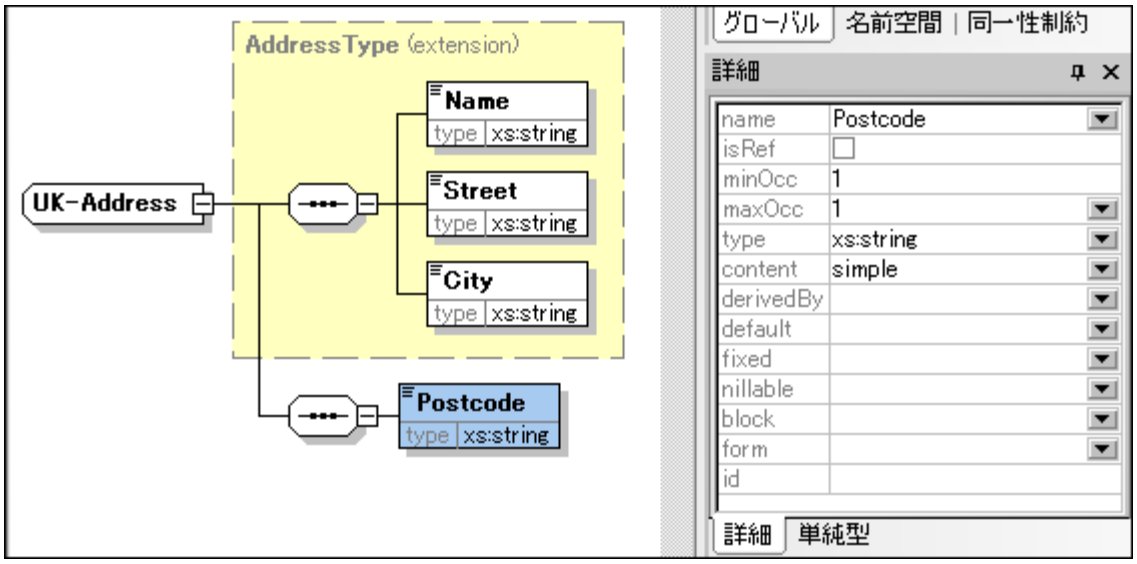
State 要素は、 **US-State** 単純型をベースにします。

第 2 の複合型を AddressType をベースにして作成する

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

次を行います:

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

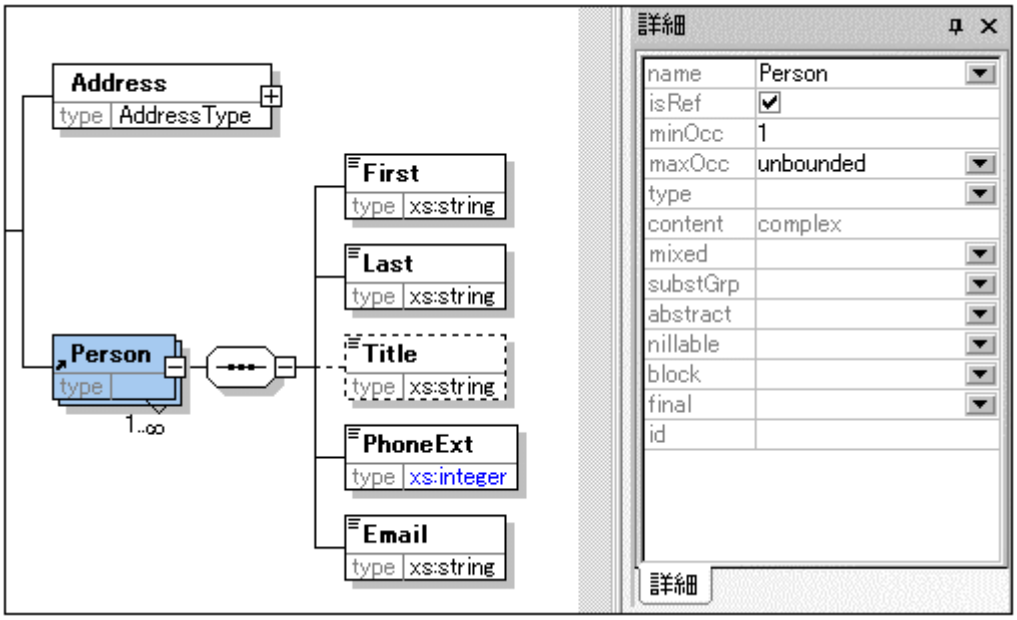


メモ このセクションでは、コンテンツモデル定義で使用されるグローバル単純型 および複合型を作成しました。グローバル型の利点は、複数の定義で再利用できることです。

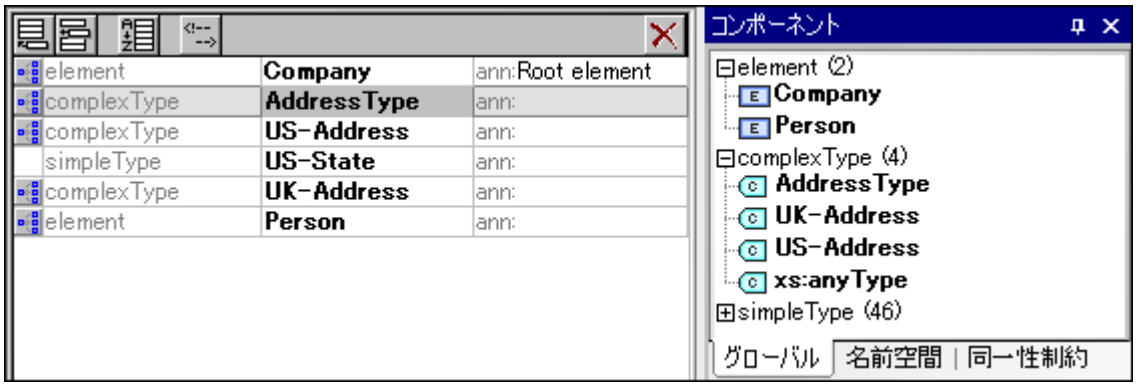
2.3.2 グローバル要素の参照

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

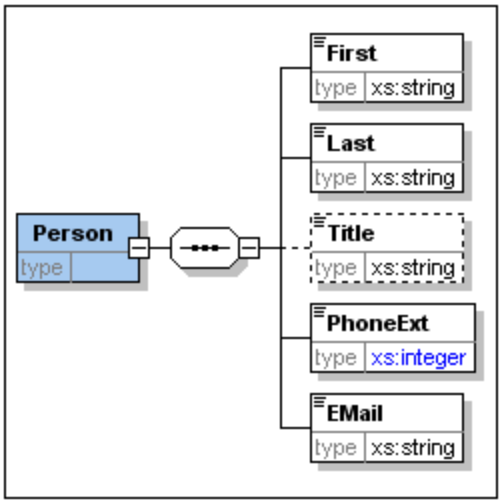
1. 全てのグローバルを表示アイコン をクリックして、スキーマ概要に移動します。
2. `Company` 要素のダイアグラム表示アイコン をクリックします。
3. `Person` 要素を右クリックし、「グローバル化 | 要素」を選択します。要素に小さな矢印アイコンが表示され、この要素がグローバルに宣言された `Person` 要素を参照していることを表します。詳細入力ヘルパーには `isRef` チェックボックスにチェックが入ります。



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



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



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

以下の点に注意してください:

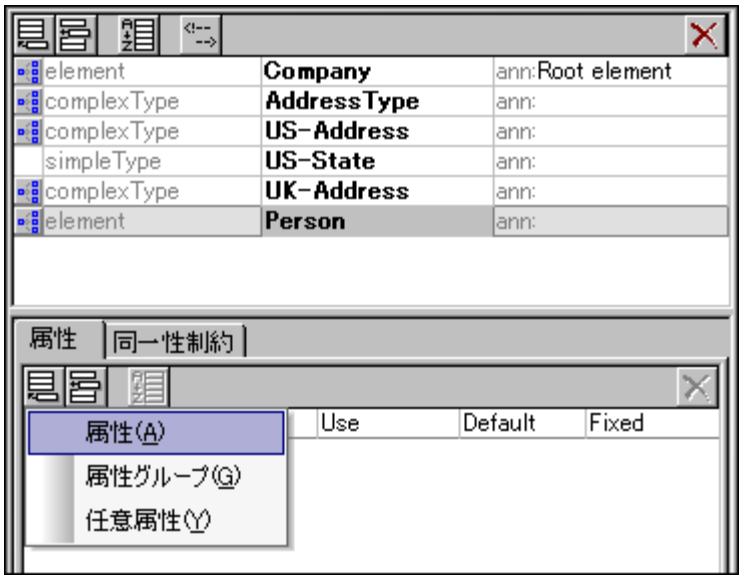
- グローバル要素を参照する要素は、参照するグローバル要素と同じ名前でなければなりません。
- グローバル定義はXMLドキュメントのどこでコンポーネントが使用されるかの記述は行わず、コンテンツモデルの記述だけを行います。グローバル定義が他のコンポーネントから参照された時に、XMLドキュメント内での場所が指定されます。
- グローバルに記述された要素は異なる場所で再利用することが可能です。グローバル複合型の場合とは違い、グローバル要素そのものを変更することなくコンテンツモデルの変更を行うことはできません。グローバル要素を参照する要素のコンテンツモデルを変更した場合、グローバル要素のコンテンツモデルも同様に変更され、そのグローバル要素を参照しているその他の要素のコンテンツモデルも変更されます。

2.3.3 属性と列挙型

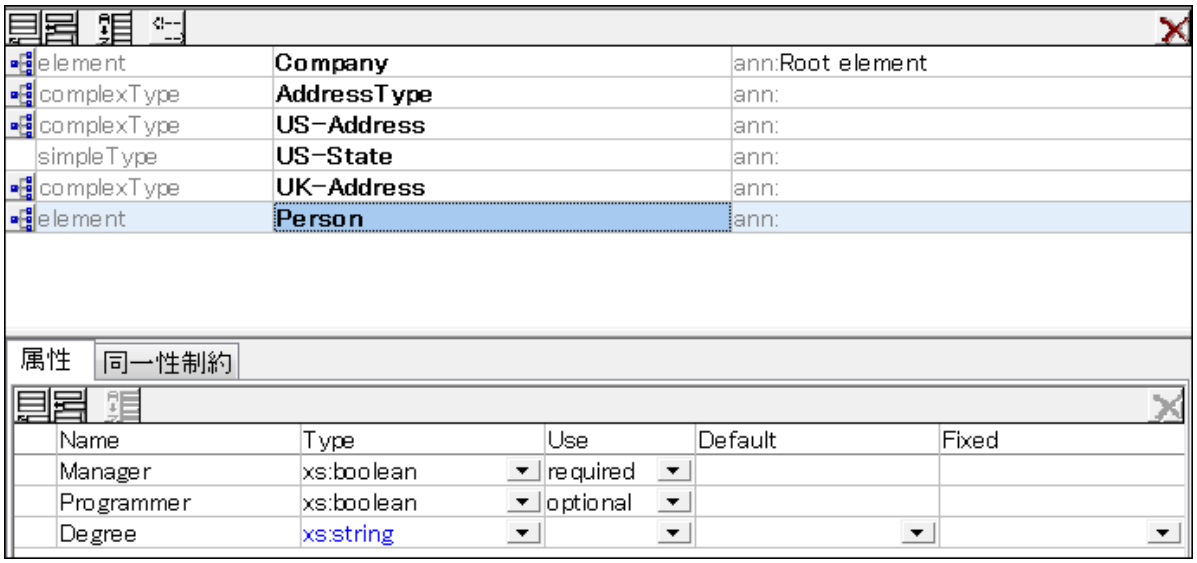
このセクションでは、属性と列挙型の属性の作成方法について学習します

要素属性の定義

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



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



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

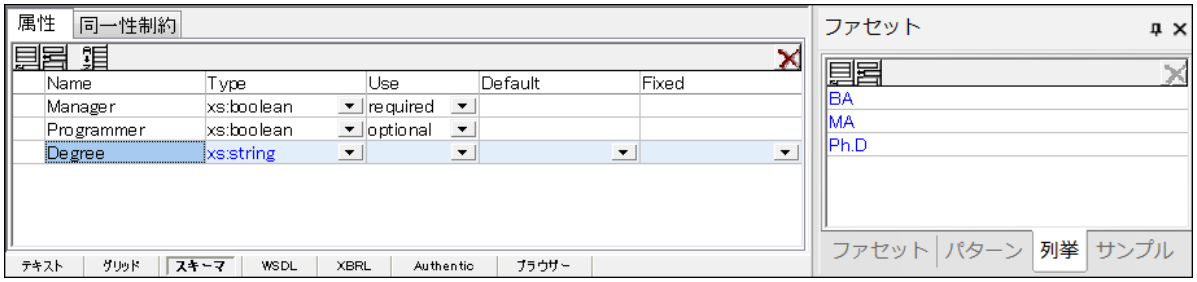
属性の列挙型の定義

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

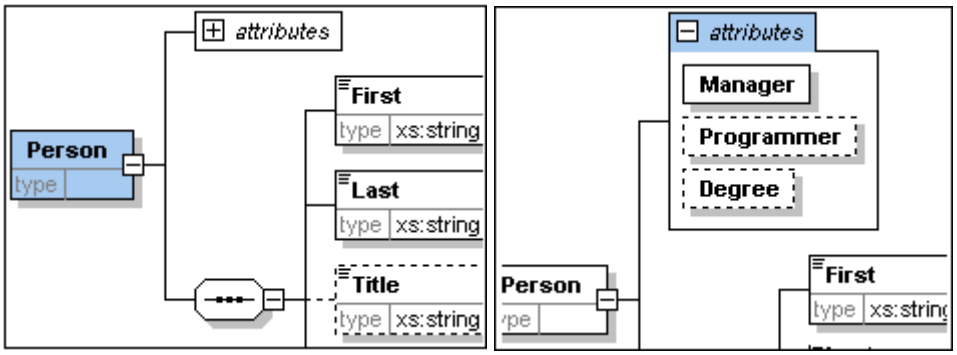
次を行います:

1. スキーマ概要にて、**Person** 要素をクリックしてアクティブにします。
2. 属性ウィンドウの左上にある追加アイコン をクリックして、「属性」エントリを選択します。

3. 属性の名前に **Degree** と入力し、**Type** に **xs:string** を選択します。
4. **Degree** 属性を選択した状態で、入力ヘルパー内の「**列挙**」タブをクリックします(スクリーンショットを参照)。



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



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

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

完成したスキーマの保存

XMLSpy によるスキーマファイルを保存する前に、既にある `AddressLast.xsd` ファイルの名前を変更し(例えば **AddressLast_original.xsd**)、上書きしないようにします。完成したスキーマファイルを好きな名前で保存します(「**ファイル | 名前を付けて保存**」)。チュートリアルの次のパートで使用されるスキーマファイルの名前は **AddressLast.xsd** であるため、ここでも **AddressLast.xsd** と名前をつけることが奨励されます。これはチュートリアルの次の部分で作成される XML ファイルは **AddressLast.xsd** スキーマをベースにするためです。

2.4 XML スキーマ：XMLSpy の機能

XMLスキーマを完成させた後は、ナビゲーションショートカット[73]とXMLSpy から生成することができるスキーマドキュメンテーション[75]の学習をしましょう。これらの機能についてはこのセクションのサブセクションで説明されています。

このセクションで使用されるコマンド

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

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

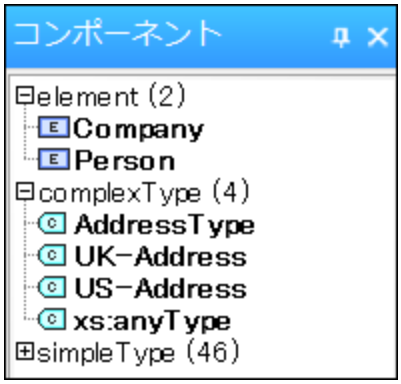
2.4.1 スキーマナビゲーション

このセクションでは効果的にスキーマビューをナビゲートする方法を学習します。

グローバルコンポーネントのコンテンツモデルを表示する

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

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

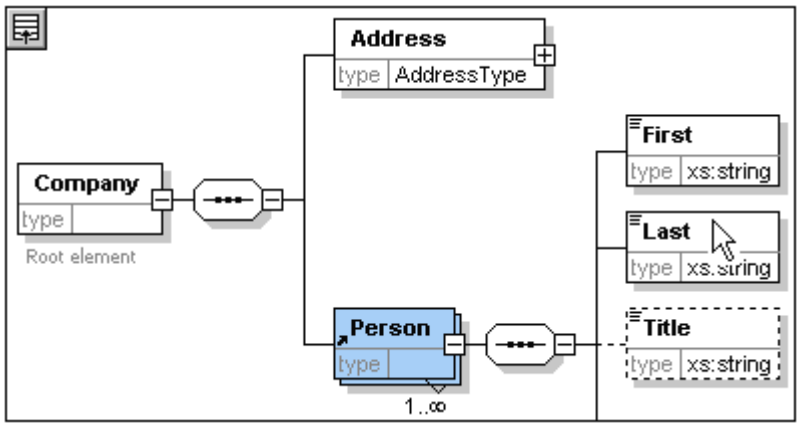


コンポーネント入力ヘルパーにあるその他のグローバルコンポーネント（単純型、属性、属性グループ）をダブルクリックすると、そのコンポーネントはスキーマ概要でハイライトされます（そのようなコンポーネントはコンテンツモデルを持たないため）。

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

参照している要素からグローバル要素に移動する

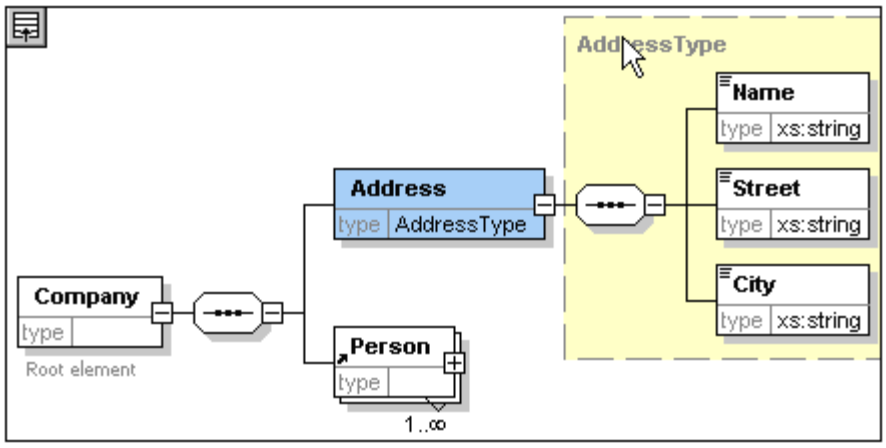
グローバル要素を参照する要素がコンテンツモデルに含まれている場合、「Ctrl」キーを押しながら必要な要素をダブルクリックすることで、そのグローバル要素またはその要素のコンポーネントに移動することができます。



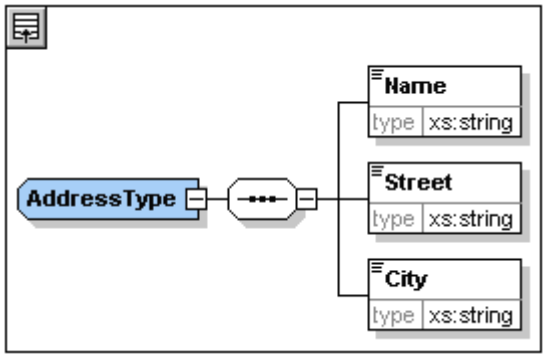
Last 要素がハイライトされたとき、関連する情報が入力ヘルパーならびに情報ウィンドウに表示されます。

複合型の定義に移動する

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



この操作で複合型のコンテンツモデルビューに移動することができます。



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

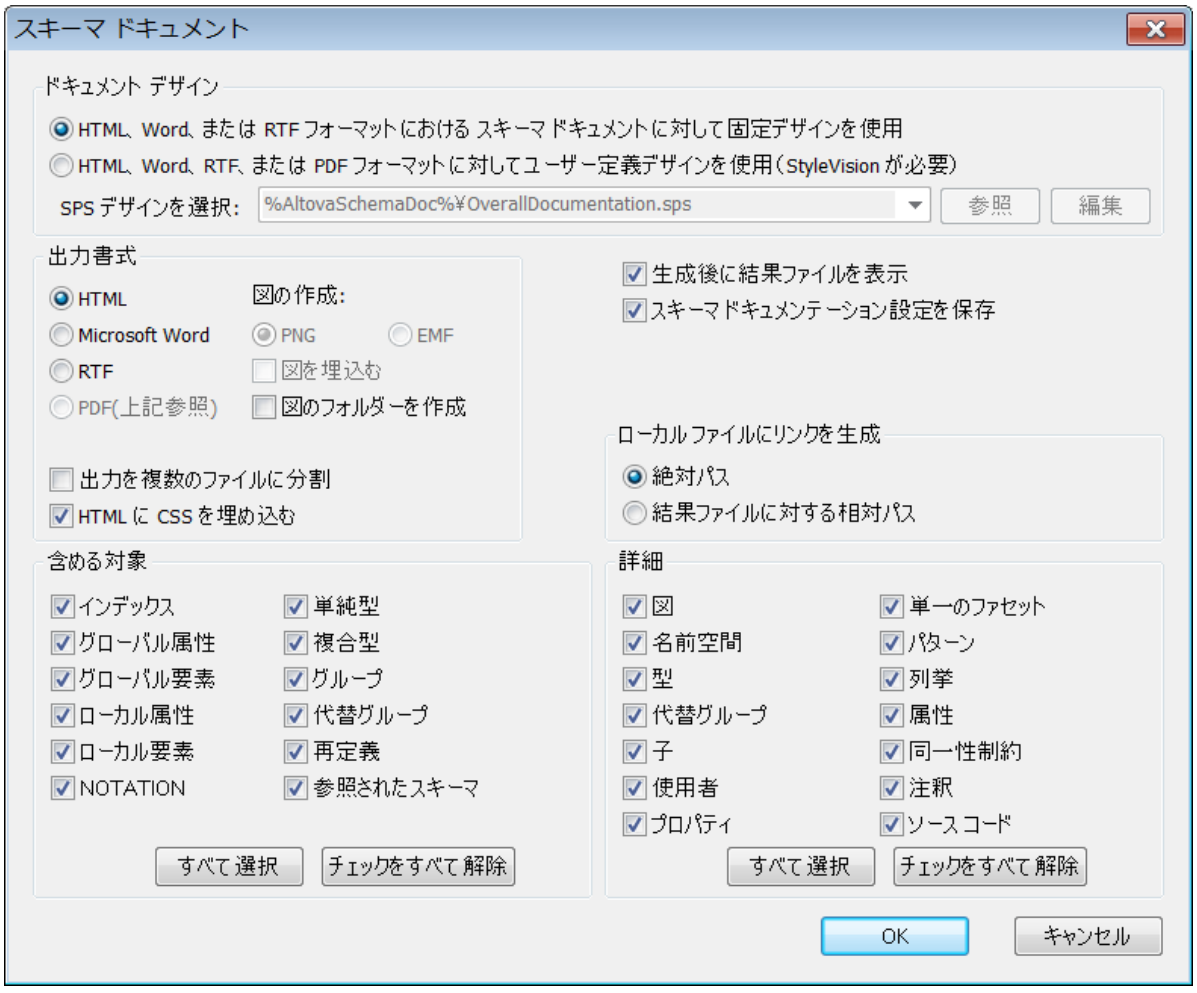
2.4.2　スキーマドキュメンテーション

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

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

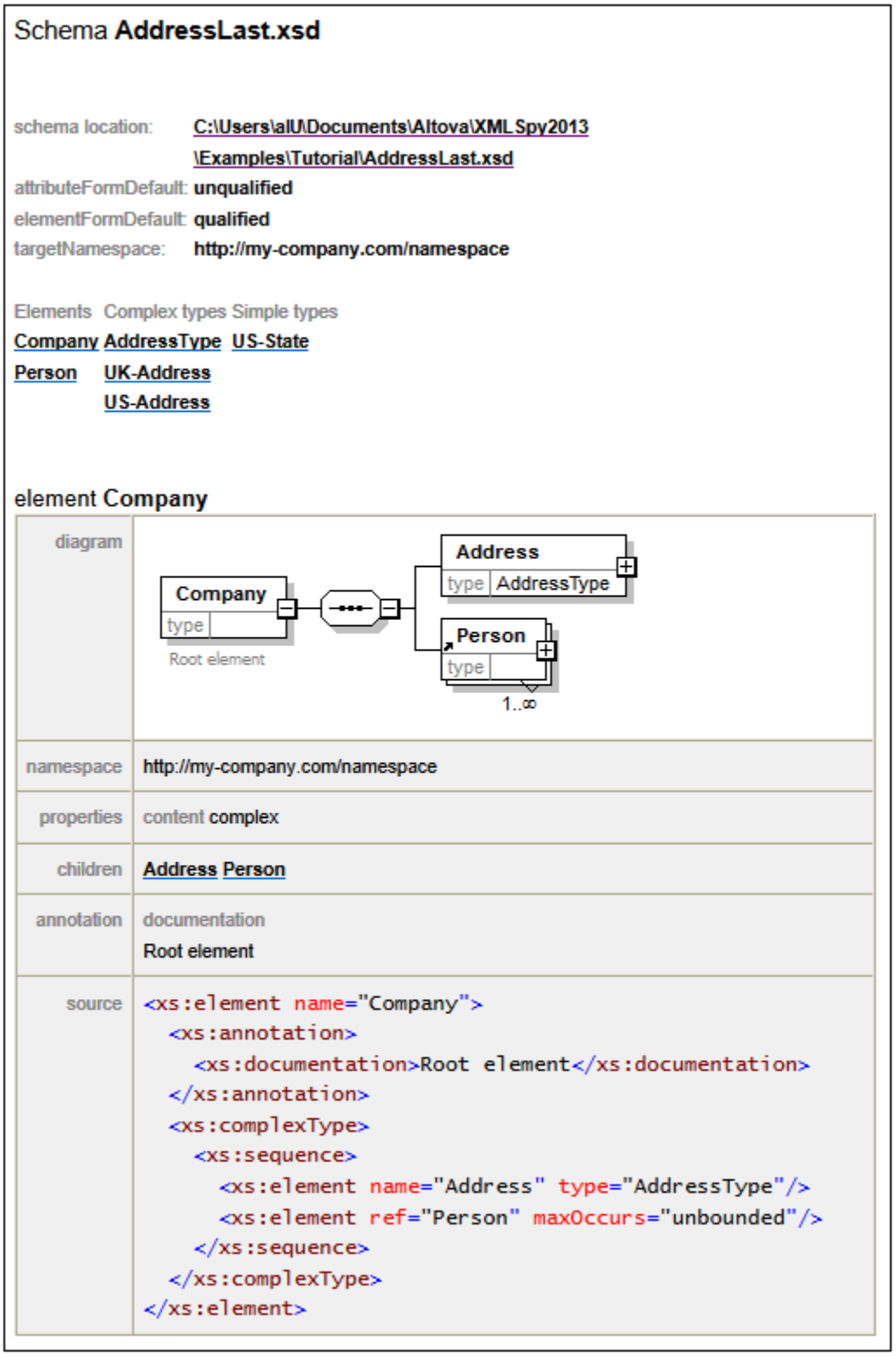
以下を行います:

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



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

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



Schema **AddressLast.xsd**

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

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

element **Company**

diagram

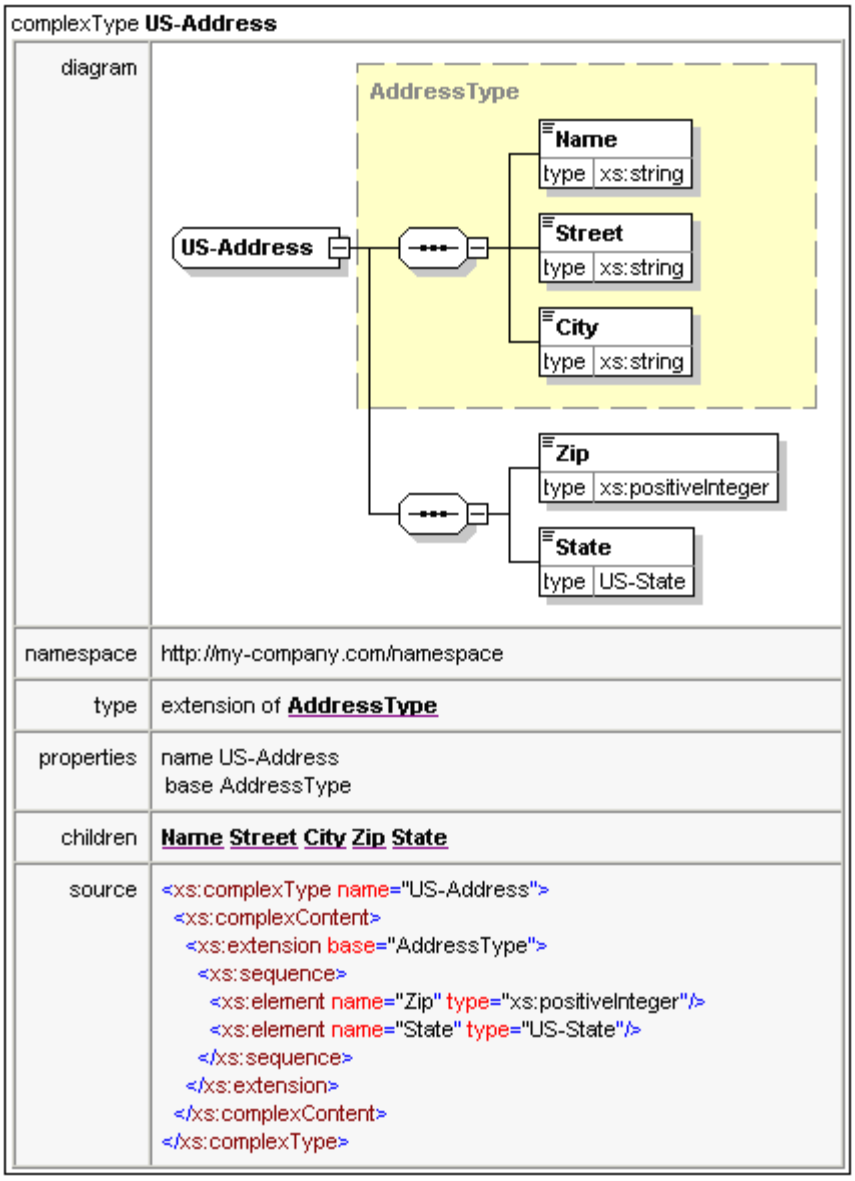
namespace	http://my-company.com/namespace
properties	content complex
children	**Address** **Person**
annotation	documentation Root element

source

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

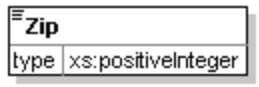

(C) 2015-2021 Altova GmbH

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

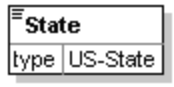


上の図は複合型がどのように表示されるか示しています。

element **US-Address/Zip**

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

element **US-Address/State**

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

simpleType **US-State**

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

上の図は要素や単純型がどのように表示されるか示しています。

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

(C) 2015-2021 Altova GmbH

2.5 XML ドキュメント

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

目的

このセクションでは以下のことを学習します:

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

このセクションで使用されるコマンド

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

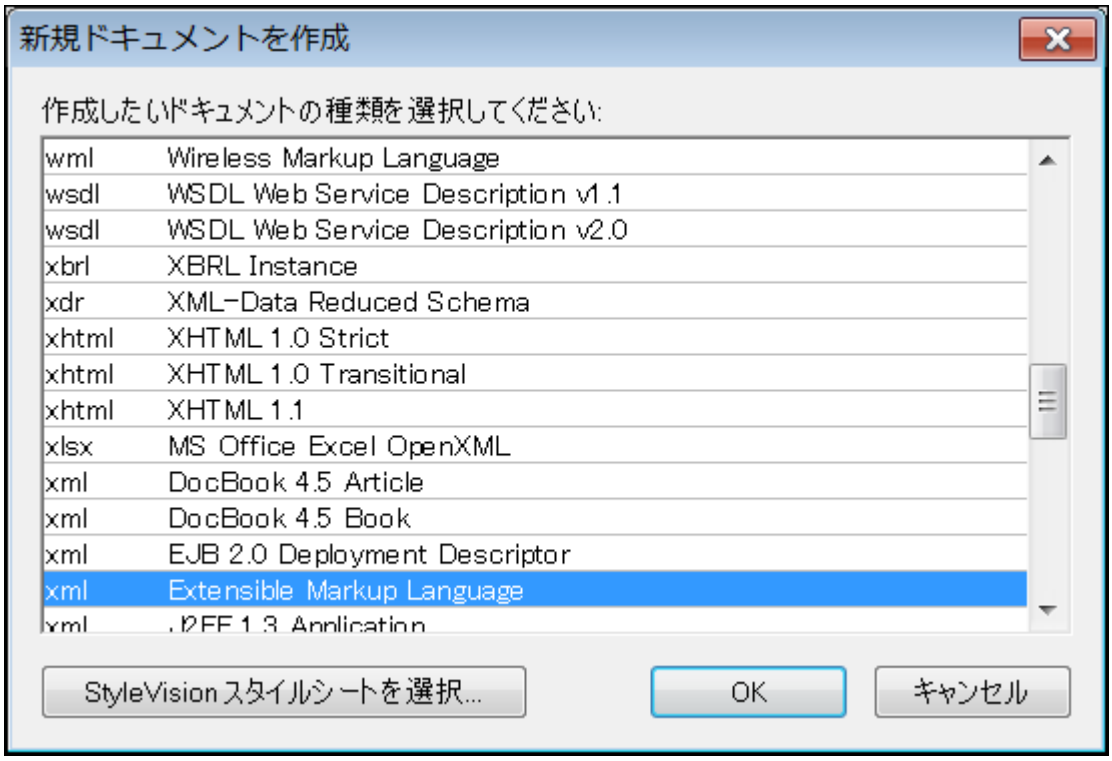
	**ファイル	新規作成** 新しい種類のXMLファイルを作成します。
	**表示	テキストビュー** テキストビューに切り替えます。
	**表示	グリッドビュー** 拡張グリッドビューに切り替えます。
	XML	テーブル	テーブルとして表示 1つの要素型の複数の出現をフラットなテーブルで表示します。このビューはデータベーステーブルビュー(または単純にテーブルビュー)といいます。アイコンはグリッドビューで一般的なビューとテーブルビューを切り替えるために使用されます。
	F7。整形式のチェックを行います。
	F8。割り当てられたDTD またはスキーマに対してXMLドキュメントの検証を行います。
	割り当てられたDTD またはスキーマファイルを開きます。

2.5.1 新規 XML ファイルの作成

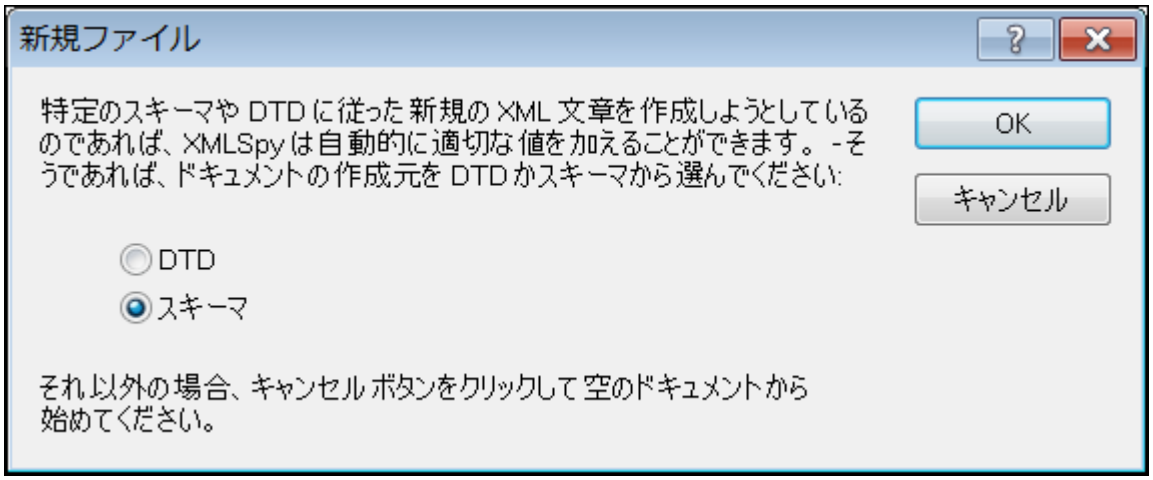
XMLSpy にて新しいXML ファイルを作成する際、スキーマ(DTD またはXML スキーマ)をベースにしたものにするかどうか決定することができます。このセクションでは、既に作成した **AddressLast.xsd** スキーマをベースにした新しいファイルを作成します。

新しいXML ファイルを作成する方法:

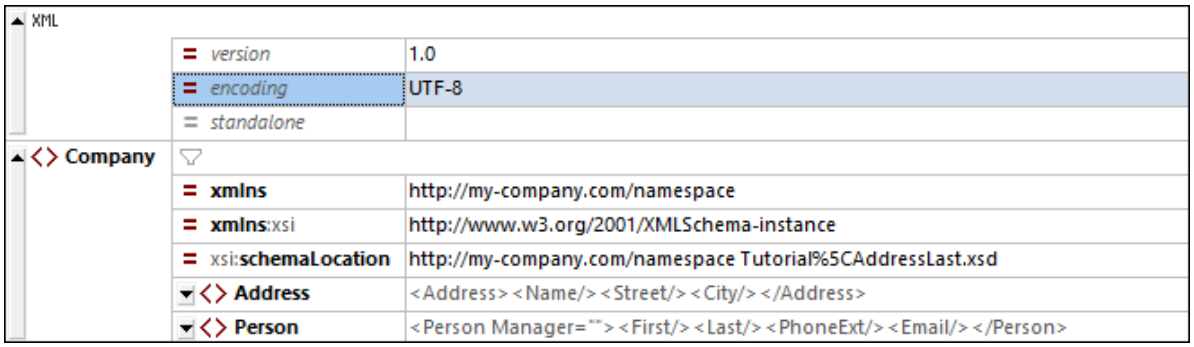
1. メニューオプション[**ファイル | 新規作成**]を選択します。*新規ドキュメント*を作成ダイアログが開かれます。



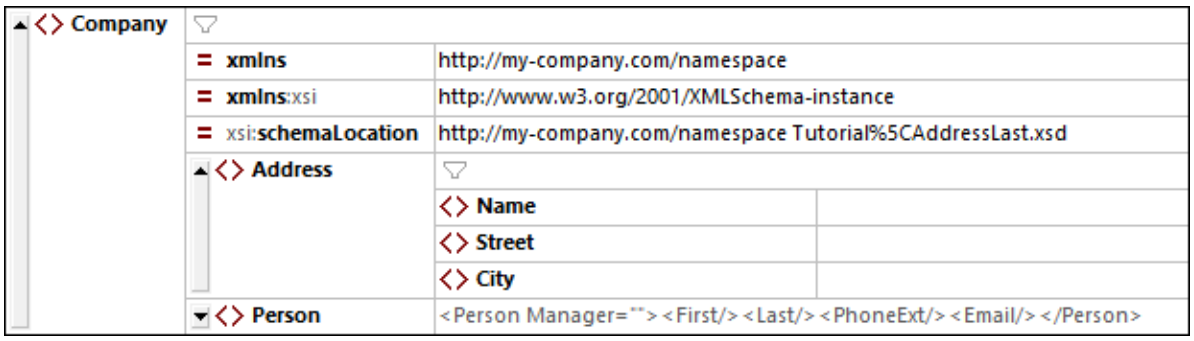
2. ダイアログから *Extensible Markup 言語* エントリ(ジェネリックな XML ドキュメントエントリ) を選択し、「OK」で確定します。ポップアップが表示され、XML ドキュメントのベースに DTD または スキーマを使用するか尋ねられます。



3. スキーマラジオボタンをクリックし、「OK」で確定します。次のダイアログが表示され、XML ドキュメントのベースとなるスキーマファイルを選ぶように促されます。
4. 参照もしくはウィンドウボタンを使用して、スキーマファイルを見つけます。ウィンドウボタンを使うことで XMLSpy にある全てのファイルとプロジェクトを閲覧することができます。`AddressLast.xsd` を選択し(チュートリアルのイントロダクション 35 を参照ください)、「OK」により確定します。スキーマで定義された要素を含む XML ドキュメントがメインウィンドウで開かれます。
5. グリッドタブをクリックしてグリッドビューを選択します。
6. グリッドビューに表示されるドキュメントの構造に注目してください。要素をクリックして選択範囲を狭めてください。ドキュメントは以下のように表示されます:



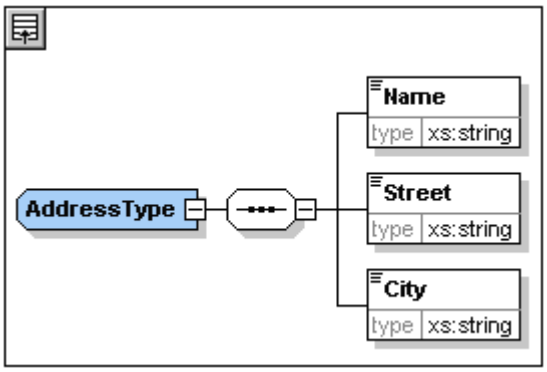
7. **Address** の横にある アイコンをクリックして、**Address** の子要素を表示します。ドキュメントは以下のようになるはずです。ドキュメントは以下のように表示されます:



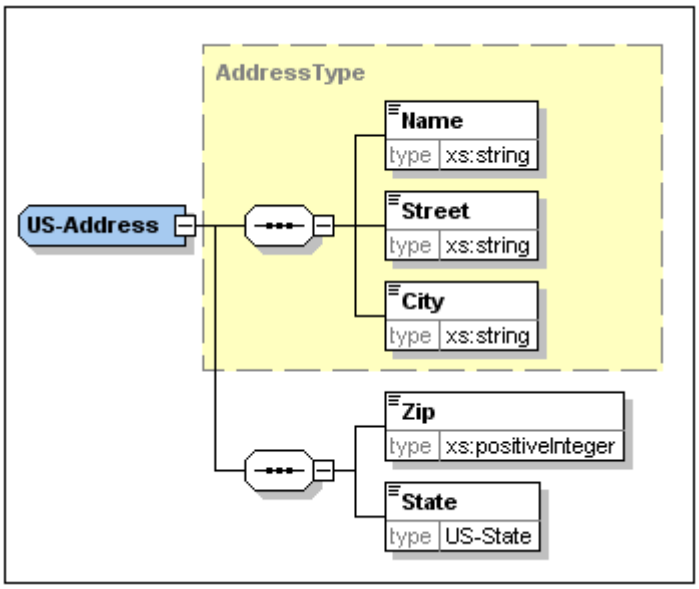
8. メニューオプション [ファイル | 保存] を選択して **Tutorial** フォルダー内に保存します。XML ドキュメントに適切な名前を与えます（例えば、**CompanyFirst.xml**）。**Tutorial** フォルダー内にも既に **CompanyFirst.xml** ファイルがあることに注意してください。作成したファイルを保存する前に、そちらのファイルの名前を変更する必要があるかも知れません。

2.5.2 要素の型の指定

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



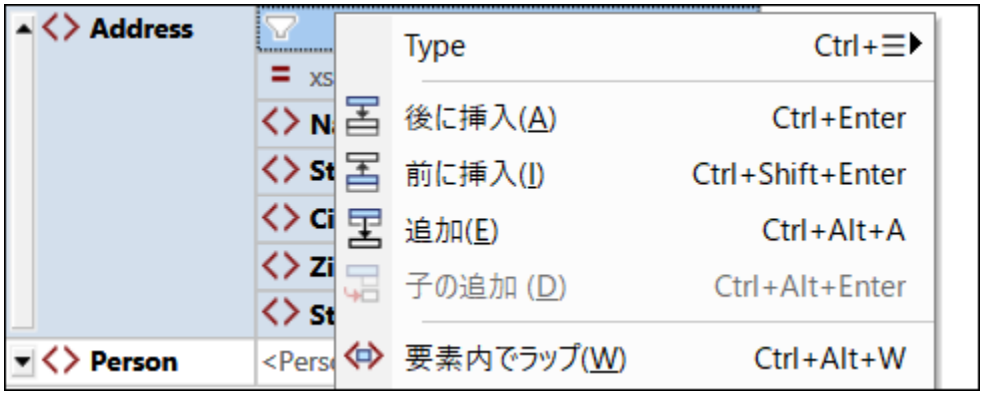
しかしここでは一般的な住所の型よりも、US または UK の住所を使用します。`AddressLast.xsd` スキーマで `US-Address` と `UK-Address` というグローバル複合型を作成したのを覚えているでしょうか？`US-Address` のコンテンツモデルを以下に示します。



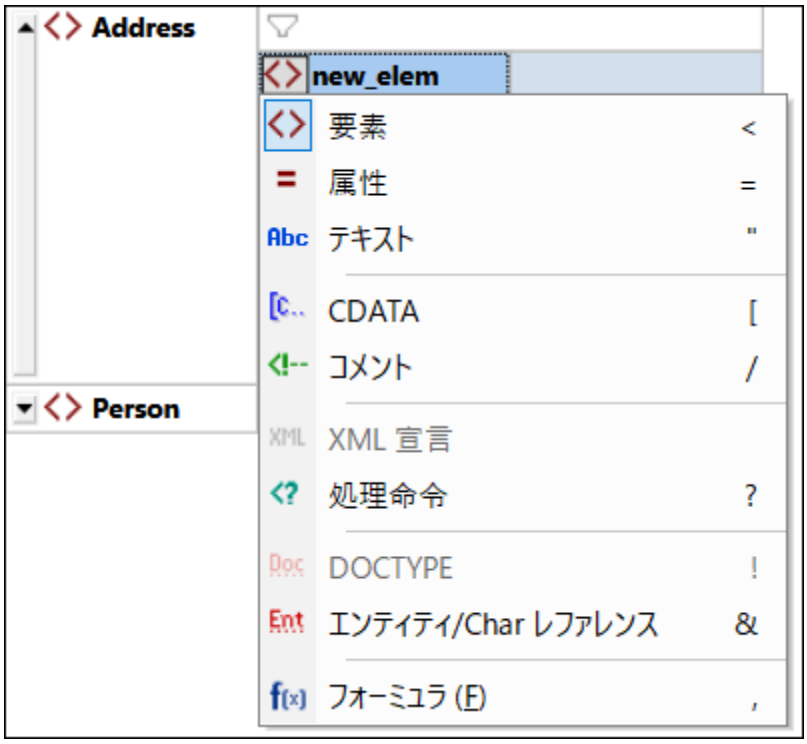
XML ドキュメントにて Address 要素が(一般的な AddressType 型ではなく)拡張された Address 型(US-Address か UK-Address)の要件を満たすためには、必要な複合型を Address 要素の属性として指定しなければなりません。

Address 要素のこの属性を次のように追加します:follows:

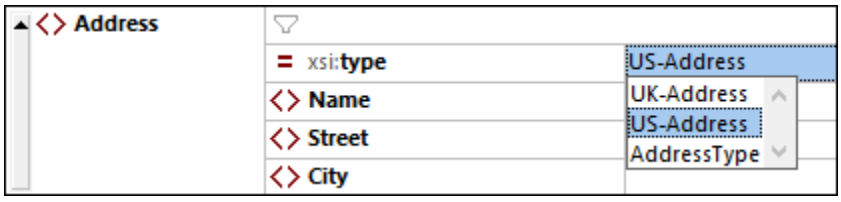
1. XML ドキュメント内で Name 要素を右クリックして[**前に挿入**]をコンテキストメニューから選択します(*下のスクリーンショット参照*)。



2. **new_elem** という名前の新規の要素ノードが**Name** 要素の上に追加されます(*下のスクリーンショットを参照してください*)。ノード名の左側の要素の型をクリックすると表示されるメニューから(*下のスクリーンショット*) *Attribute* ノード型を選択します。ノード型は*Attribute* ノード型に変更されますが名前は**new_elem** のままです。



3. ノード名をダブルクリックして、表示される入力ヘルパーから **xsi:type** を選択します。
4. 属性の値フィールドに移動するために **Tab** キーを押します。使用可能な **xsi:type** 値を含むポップアップが表示されます（*下のスクリーンショット参照*）。これらの値はスキーマ内の **Address** 要素のために定義された複合型です。



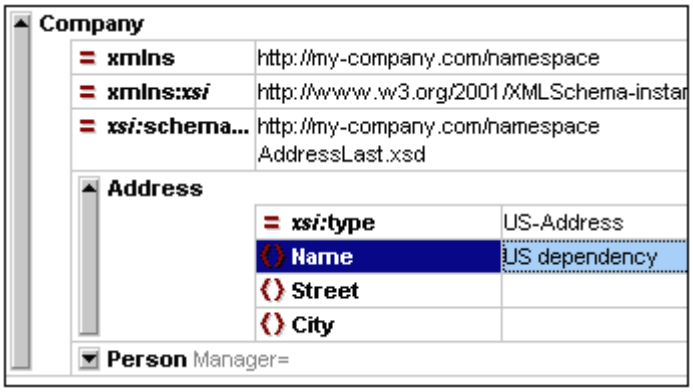
5. **US-Address** を **xsi:type** 属性の値として選択します。

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

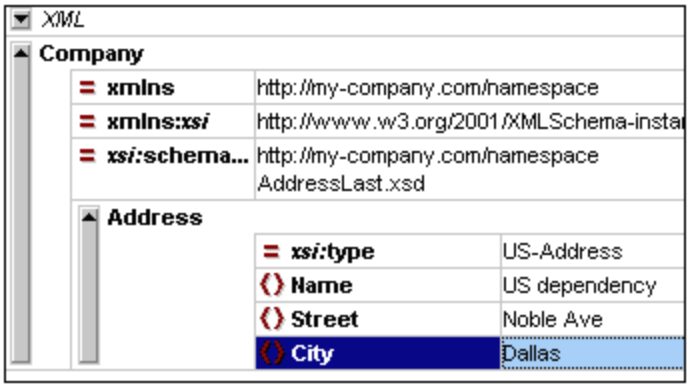
2.5.3 グリッドビューを使ったデータの入力

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

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



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



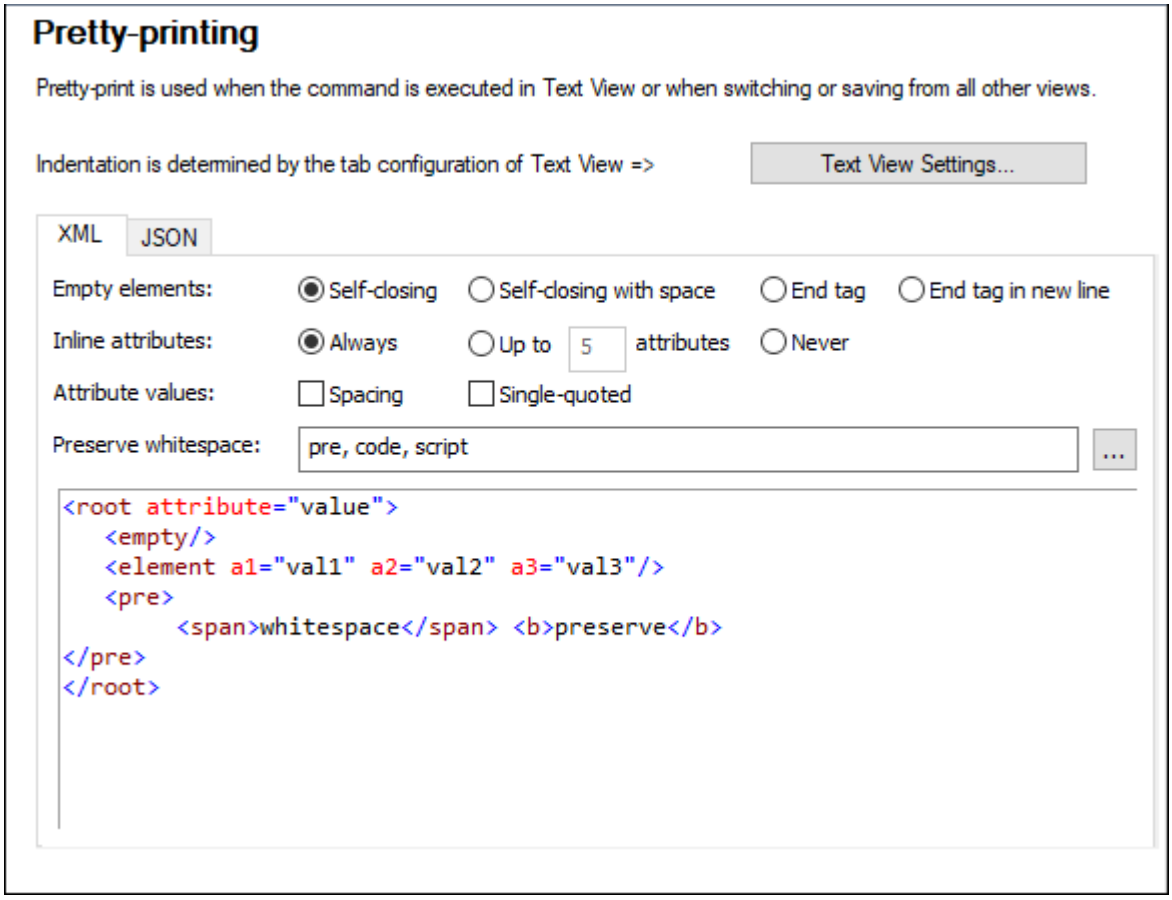
2.5.4 テキストビューを使ったデータの入力

テキストビューにはXML ファイルの実際のデータとマークアップを簡単にフォローできる構造的なレイアウトとスキーマに関連したインテリジェントな編集機能が搭載されています。

ドキュメントのレイアウト

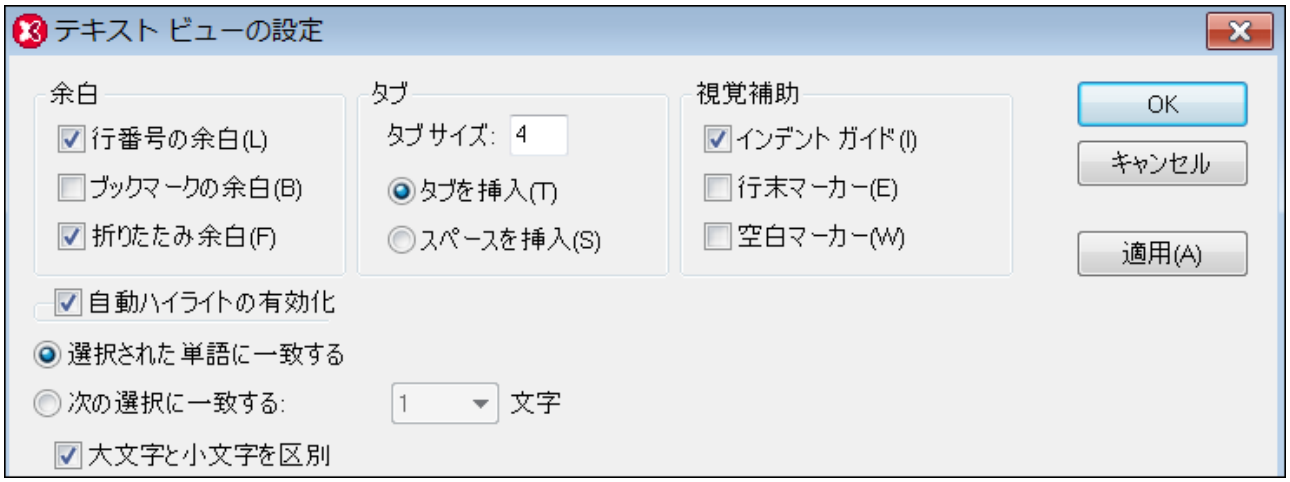
テキストビューのドキュメントのレイアウトは2つの場所で定義されています:

整形出力オプション
これらの設定はオプションダイアログの整形出力設定内にあります(*下のスクリーンショット*)。オプションを設定すると、下のプレビューペインですぐに効果を確認することができます。整形出力を必要に応じてセットアップします。テキストビュー内で編集中に特にテキストのブロックをコピーして貼り付けるとドキュメントのレイアウトが非構造化されてきていることに気が付きます。整頓され構造されているレイアウトが必要な場合[**編集 | 整形出力**] コマンドをクリックします。

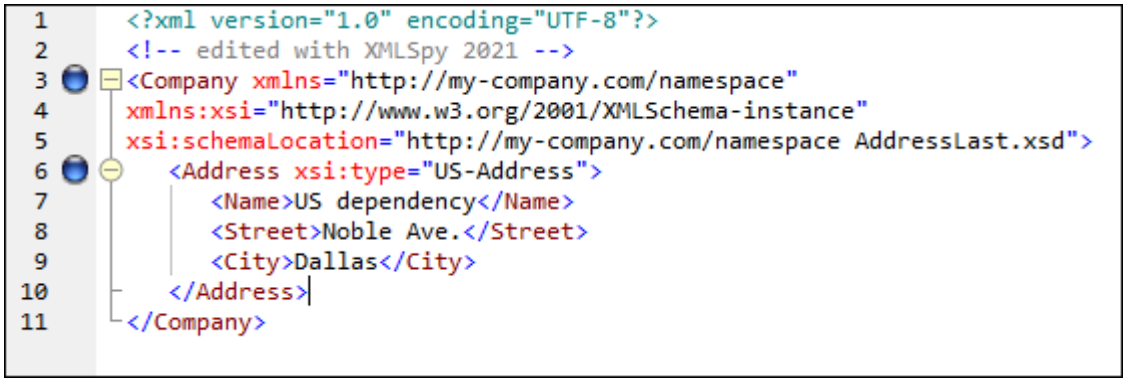


テキストビューの設定

テキストビュー設定ダイアログ（*下のスクリーンショット*）は追加レイアウトオプションだけでなく行番号やマージンの折りたたみなどテキストビュー機能のオン/オフを切り替える機能を提供します。[**表示 | テキストビュー設定**] コマンドを使用してテキストビュー設定ダイアログにアクセスします。



下のスクリーンショットは上のダイアログ内の設定に従いオンに切り替えられているテキストビュー内の現在のXML ファイルを表示しています。



左側には3つのマージンが存在します: (i) 行番号マージン (ii) (2つの青いブックマークが含まれている) ブックマークマージン、および (iii) (XML 要素の表示を展開し折りたたむことを許可する) ソースの折りたたみマージン。インデントガイドは同じ階層レベルでタグのインデントを表示する薄い灰色の垂直線です。更に資格補助はオンとオフに切り替え可能な行末マーカーと空白マーカーです *視覚補助* ペイン (*上のスクリーンショット参照*)。

メモ テキストビューに関連した印刷およびブックマークの機能はこのチュートリアルのテキストビューの設定 42 のセクションで説明されています。

テキストビューでの編集

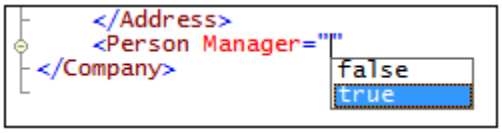
テキストビューに表示される以下の余白 (マージン) の有効/無効を設定することができます。

メモ *編集中の検証* 機能はデフォルトでオンに切り替えられるため、編集中に生成された検証エラーはメッセージウィンドウにエラーメッセージとしてすぐにフラグされます。これらのエラーとメッセージを今の時点では無視してください。バックグラウンドでの検証を必要としない場合、*編集中の検証* をオプションダイアログの検証設定 1455 内でオフに切り替えてください。これを行う場合、ドキュメントをいつでも検証できることに注意してください (このチュートリアルの次のセクション 88 で説明されています)。

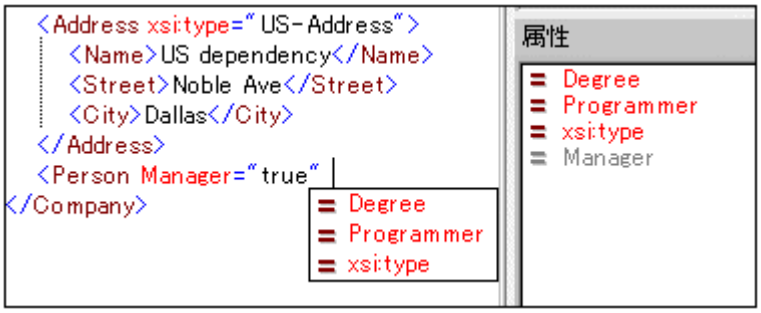
以下を行います:

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

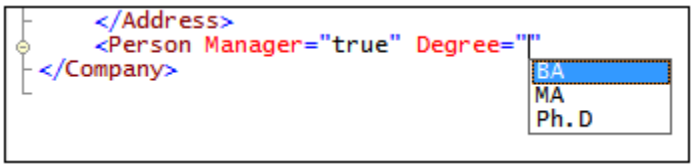
4. **Person** エントリを選択します。**Person** 要素が **Manager** 属性と共に挿入され、**Manager** 属性の値フィールドの内部にカーソルが移動します。
5. **Manager** 属性のポップアップから **true** を選択します。



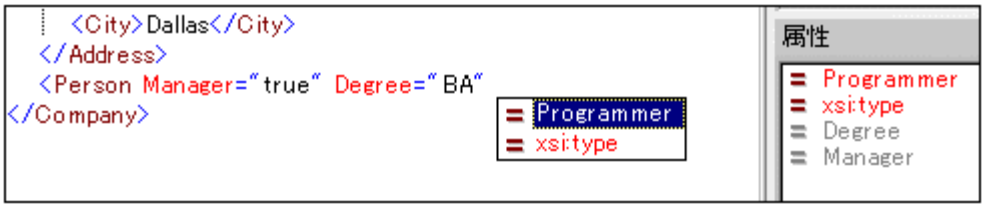
6. カーソルを行末に移動し（例えば「End」キーを使うことができます）、スペースバーを押します。新しいポップアップが開かれ、この位置で許されている属性のリストが表示されます。同様に、属性入力ヘルパーにて使用することのできる属性が赤色で表示されます。**Manager** 属性はグレーになっていますが、それはその属性が既に使われたためです。



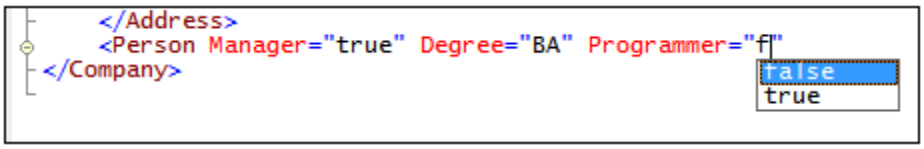
7. **[Degree]** を下向きの矢印を使用して選択し **[Enter]**を押します。これでまた別のポップアップが表示され、既に定義された列挙（**BA**、**MA**、または **PhD**）から選択を行うことができます。（列挙はXML スキーマにより許可されている値です。）



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



9. **Programmer** を下向きの矢印を使用して選択し **[Enter]**を押します。



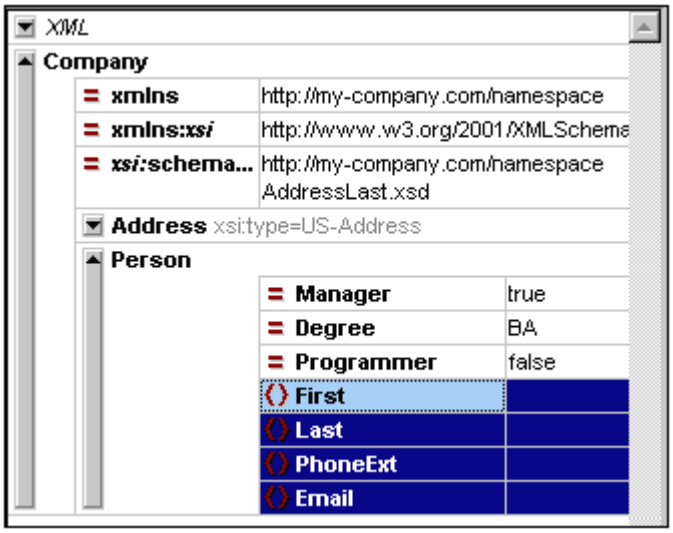
10. 文字 **"f"** を入力し、**[Enter]**を押します。
11. (**End** キーを使用して) カーソルを行末まで移動し、大なり記号 **>** を入力します。XMLSpy が**Person** に必要な子要素を自動的に挿入します（省略可能な **Title** 要素が挿入されていないことにご注目してください）。全ての要素には開始と終了タグがありますが、コンテンツはありません。

```
<Company xmlns="http://my-company.com/namespace" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://my-company.com/namespace
AddressLast.xsd">
    <Address xsi:type="US-Address">
        <Name>US dependency</Name>
        <Street>Noble Ave.</Street>
        <City>Dallas</City>
    </Address>
    <Person Manager="true" Degree="BA" Programmer="false">
        <First></First>
        <Last></Last>
        <PhoneExt></PhoneExt>
        <Email></Email>
    </Person>
</Company>
```

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

グリッドビューへ切り替え

グリッドビューへ変更するには、メニューオプションの「表示 | 拡張グリッドビュー」を選択するか、「グリッド」タグをクリックします。新たに加えられた `Person` の子要素がどのように表示されているか確認してください。



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

2.5.5 ドキュメントの検証

XMLSpy ではXMLドキュメントを2 つの方法でチェックできます:

- 整形式のチェック
- 検証のチェック

	整形式のチェック

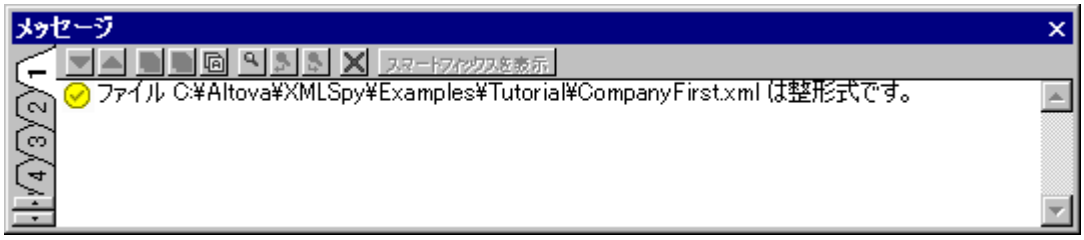
	XML 検証

編集中の検証 機能はデフォルトでオンに切り替えられるため、編集中に生成された検証エラーはメッセージウィンドウにエラーメッセージとしてすぐにフラグされます。バックグラウンドでの検証を必要としない場合、*編集中の検証* をオプションダイアログの検証設定[1455]内でオフに切り替えてください。これを行う場合 XML メニュー内で対応するコマンドを呼び出して整形式チェックと検証チェックを個別にいつでも行えることに注意してください。チュートリアルの一部でこれらのチェックを行う方法が説明されています。

整形式チェック

開始タグと終了タグがマッチし、要素が正しくネストされ、間違った文字があったり必要な文字（例えばセミコロンの無いエンティティなど）がなかったりしなければ、そのXML ドキュメントは整形式であると言います。全ての編集ビューで整形式をチェックすることができます。ドキュメントを以下のようにチェックします:

1. テキストビューを選択します。
2. メニューオプション [XML | **整形式のチェック**] または **F7** キーをチェックします。（またはツールバー内のコマンドアイコンをクリックすることができます。）メインウィンドウの下にあるメッセージウィンドウにドキュメントは整形式であるというメッセージが表示されます。



メッセージウィンドウの出力には9つのタブがあることに注目してください。評価された結果の出力はアクティブなタブに対して行われます。従って、タブ1で整形式のチェックをしておきながら、タブ2に移ることができます。タブを切り替えない場合、新しい結果がアクティブなタブの前の結果を上書きします。

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

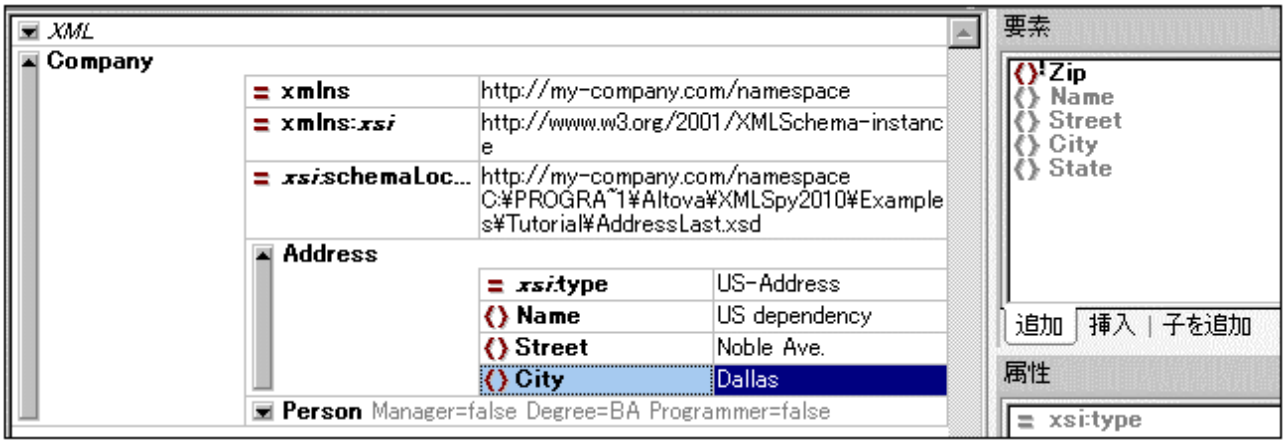
妥当性のチェック

ドキュメント構造とスキーマ内のドキュメントコンテンツに準拠するXML ドキュメントはスキーマに対して妥当です。編集ビューで妥当性をチェックすることができます。ドキュメントを以下のように検証します:

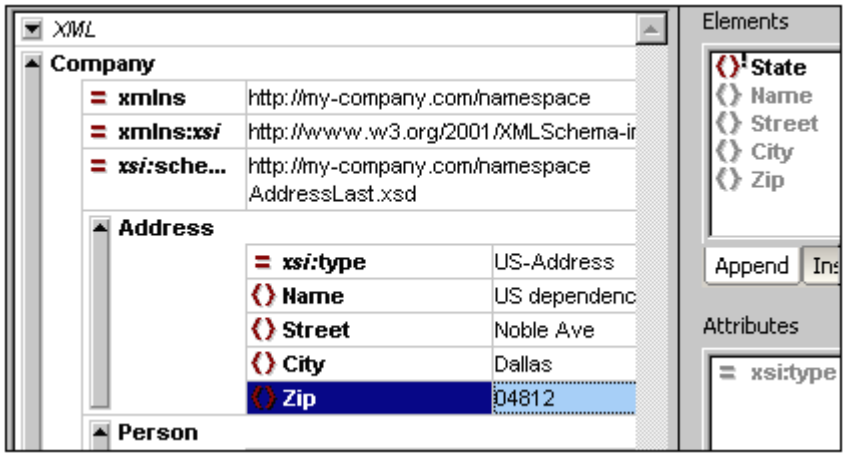
1. グリッドビューを選択します。
2. メニューオプション [XML | **検証**] または **F8** キーをチェックします。（またはツールバー内のコマンドアイコンをクリックすることができます。）メッセージウィンドウにファイルが妥当でないことを示すエラーメッセージが表示されます。`Address` 内の `City` 要素の後に必須要素が期待されます。（`xsi:type` 属性を介してこの `Address` 要素を設定する）スキーマをチェックすると `US-Address` 複合型 は `City` 要素に `Zip` 要素と `State` 要素が続く必要のあるコンテンツモデルを持ちます。

無効なドキュメントの修正

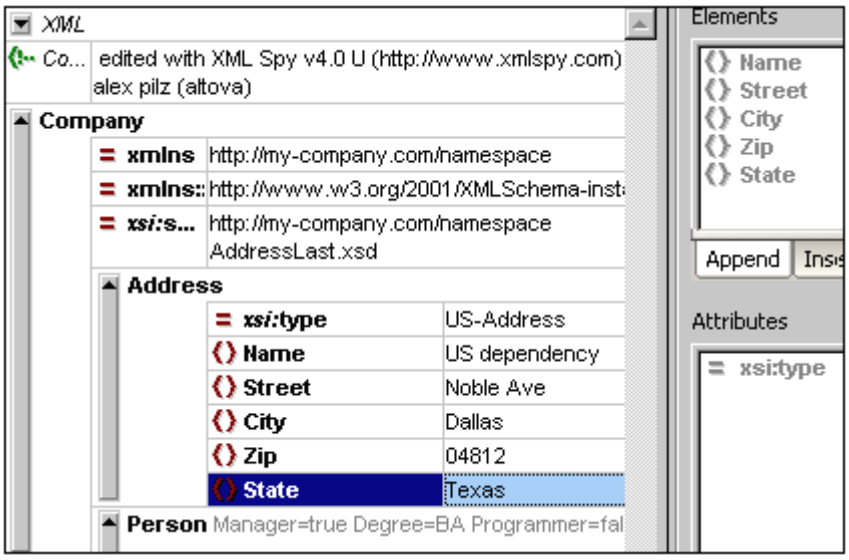
ドキュメントが無効になる個所がスマートフィックス[302]と共に赤色でハイライトされます。この場合の無効な要素は `Address` 要素です。スマートフィックスアイコンを使用するとポップアップが表示されます: *同じコンテンツを持つ不足する要素 <Zip> を追加します。*スキーマをチェックすると `Address/City` 要素が必須要素 `Zip` が後に続くことが必要なことに気が付きます。これをダブルクリックするには `City` 要素を選択して、兄弟入力ヘルパーを確認します。`Zip` 要素が現在のコンテキスト内で要素が必須なことを示す感嘆符と共にプレフィックスされていることに気が付きます。



*スマートフィックスをクリック*します（*上のスクリーンショット参照*）。`Zip` 要素が追加され、要素を有効にするサンプルコンテンツが含まれています。正確な `Zip` コードを入力します（ 例えばDallas のために `04812` を入力します）。兄弟入力ヘルパーを確認します。`State` 要素が必須であることが表示されています（感嘆符と共に表示されています）。`State` 要素を選択すると有効化されているため入力ヘルパーオプションが使用可能になります（ *下のスクリーンショット参照*）。これらは `State` 要素を（`City` である）メインウィンドウ内で現在選択されている要素の後に入力する、または `City` のすべての兄弟の後に `State` を付加するアクションです。



この場合、両方のアクションは同じ効果を持つため、2つのアクションのどちらかを選択します。`State` 要素が `City` の後に追加されます。`State` のコンテンツフィールドの内部をダブルクリックし州の名前 `Texas` を入力します（*下のスクリーンショット*）。兄弟入力ヘルパーにお追加する必須要素がもう存在しないことを示す灰色表示された要素のみが含まれていることにご注意してください。

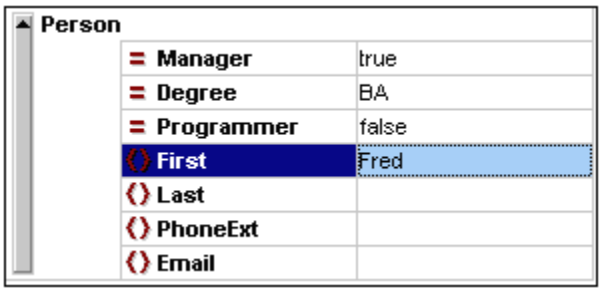


ドキュメントを完成させ、再度検証する

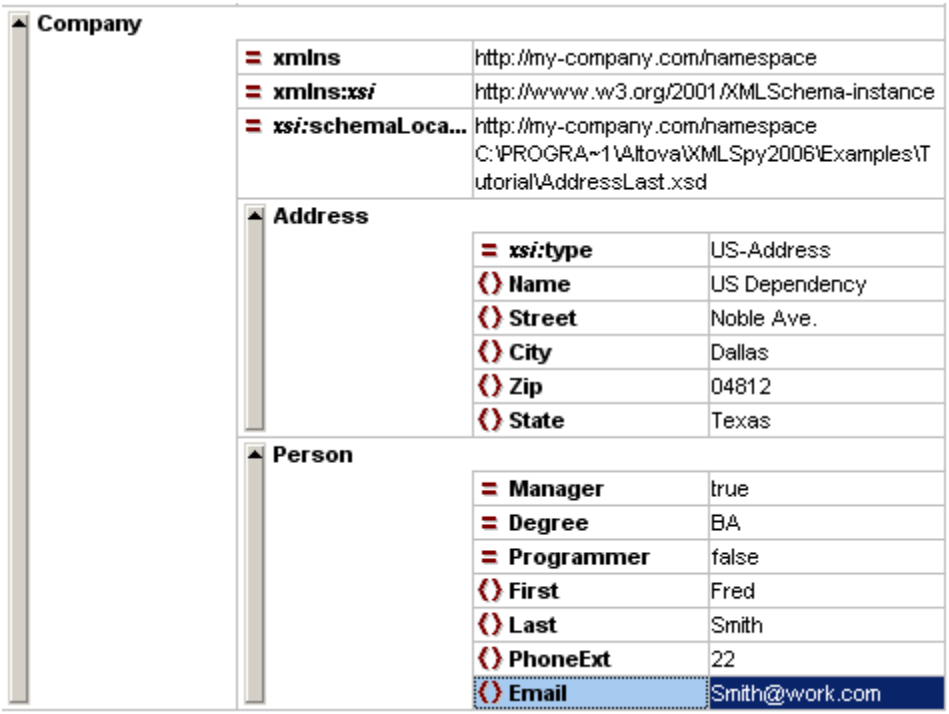
それでは再度検証を行う前に `Person` 要素の情報を入力し)ドキュメントを完成させましょう。

以下を行います:

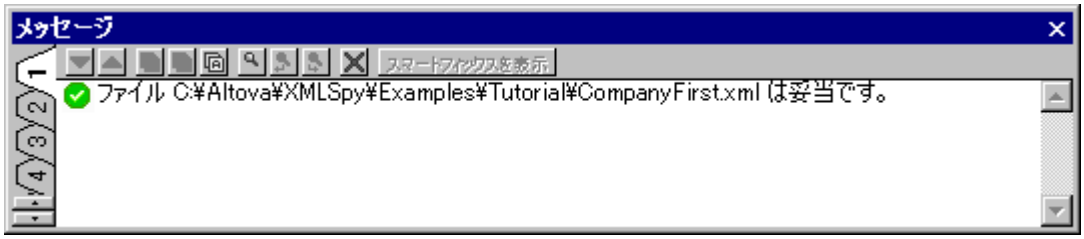
1. **`First`** 要素をクリックして、名前(例えば `Fred`)を入力し、「**Enter**」で確定します。



2. 同様の方法で **`Person`** の子要素全て(**`Last`**、**`PhoneExt`**、および **`Email`**)のデータを入力してください。Tab キーを使用してセルを前進むことができます。(スキーマ内で許可されている **`PhoneExt`** 値である) **`PhoneExt`** の値は**`99`** 以下の整数でなければならないことご注意してください。XML ドキュメントはグリッドビューで以下のようになっているはずです:



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



4. [ファイル | 保存]を使用してファイルを保存します。

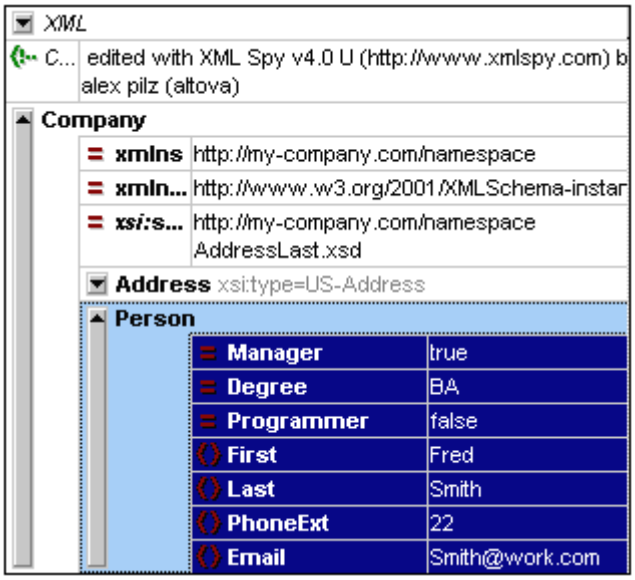
メモ: 保存する際にXMLドキュメントが妥当である必要はありません。不正なドキュメントを保存しようとすると、その旨の警告が表示されます。無効のままの状態で保存したいのであれば、**そのまま保存**を選択することができます。

2.5.6 要素と属性の追加

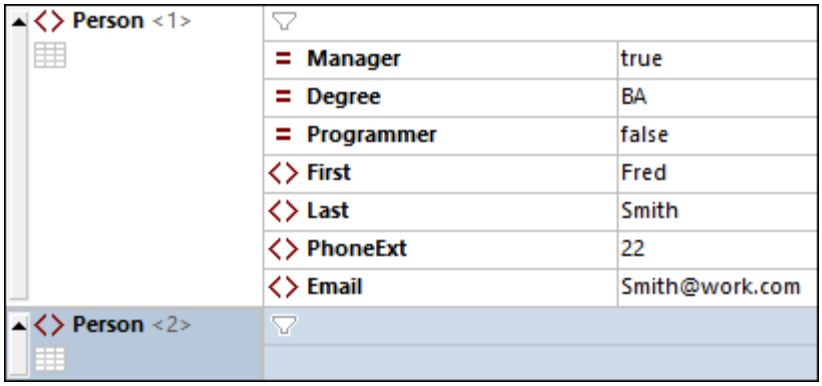
この段階ではドキュメント内に `Person` 要素は1つしかありません。

新しい `Person` 要素を追加するには以下を行います:

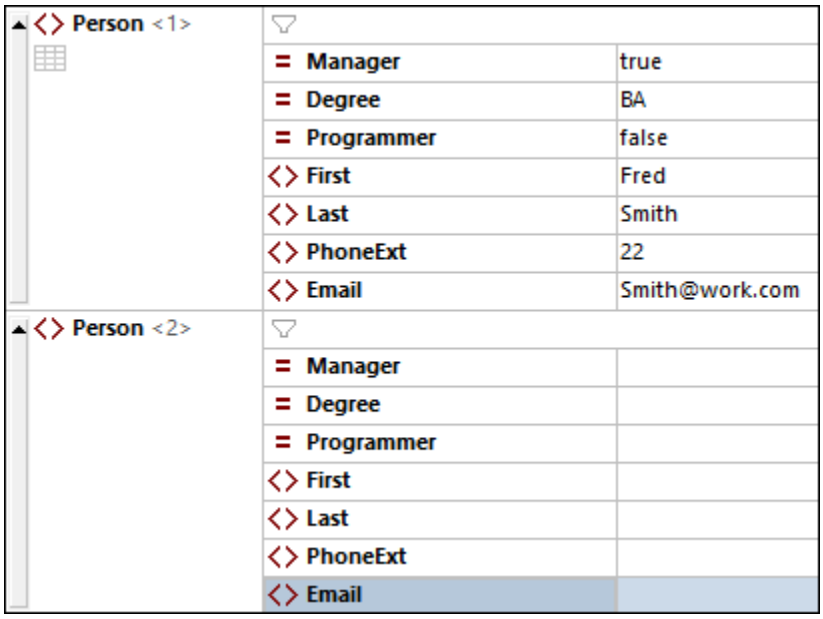
1. `Address` 要素の左にあるグレーのサイドバーをクリックして `Address` 要素を折りたたみます。これによりビュー内でのスペースを整頓することができます。
2. グリッドビューの `Person` 要素をクリックして `Person` 要素全体を選択します。要素入力ヘルパーの「追加」タブにて `Person` 要素が使えるようになっていることご注目してください。



3. 兄弟入力ヘルパー内の `Person` 要素を選択して **後に挿入** または **付加**をクリックします。新規の `Person` 要素が付加されます（*下のスクリーンショット*）。



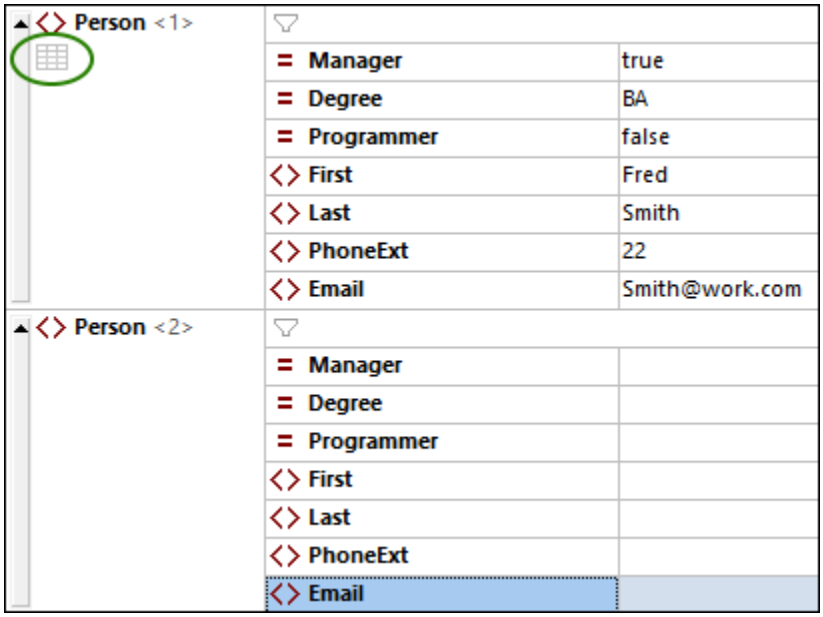
4. `Person` 要素が選択されていると、子入力ヘルパー内で、この要素で使用可能な子属性と要素を確認することができます。属性と要素をダブルクリックして最初の `Person` 要素として同じ子ノードを追加します。`Person` 要素から追加された子要素にメインウィンドウ内のフォーカスが変更されると `Person` 要素の追加の子を以下の2つの方法の1つを使用して追加することができます: (i) `Person` 要素（を選択し）子入力ヘルパーから新規の子を追加してフォーカスを切り替える（ii）追加された子要素のフォーカスを使用して、兄弟入力ヘルパーから兄弟子要素を追加する。両方の入力ヘルパーで既に追加済みの `Person` の子ノードが灰色表示されます。



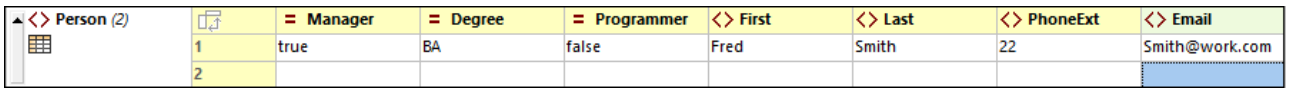
Person 要素の子ノードのために標準のグリッドビューでコンテンツを追加することができますが、Person などの複数の発生がある構造を編集するために適切なグリッドビューのテーブル表示に切り替えることにしましょう。

2.5.7 データベース/テーブルビューでの編集

グリッドビューの中には複数回出現する要素の編集に適したテーブルの表示という特別なビューが存在します。例えば Person 要素には複数の発生があり（*下のスクリーンショット*）これをテーブルとして表示することができます。このような要素をテーブルとして表示するためには要素の最初の発生の**テーブルの表示** アイコンをクリックします。例えば、下のスクリーンショット内では Person 要素の**テーブルの表示** アイコンは緑色の丸枠で表示されています。（または、メニューコマンド XML | **テーブルとして表示** または[グリッドビューツールバー](#) 268 のコマンドのツールバーアイコンをクリックします。）



テーブルの表示アイコンをクリックすると Person 要素がテーブルとして表示されます。テーブルの表示では、要素(とその属性と要素) の子ノードが列として表示されていますが、それぞれの Person 要素は行として表示されています (*下のスクリーンショットを参照*) 。



テーブルの表示の利点
テーブルの表示には以下の利点があります:

- それぞれに対応する列全体にポジションを変更するために列ヘッダーをドラッグアンドドロップすることができます。実際の XML ドキュメント内では、これは(テーブルの行に対応する) すべての要素の発生の子ノードの対応する位置の変更に反映されます。
- テーブルとこれに関連した要素の発生は列のコンテンツに従い (昇順および降順に) 並べ替えることができます。メニューコマンド [XML | 昇順に並べ替え] または [降順に並べ替え] を使用します。
- 追加行 (要素の発生) を追加することが可能になります。または **XML** メニュー内のコマンドを使用して素早く挿入することもできます。この利点は新規の要素が追加されるだけでなく、テーブルの列により示されるその子全てが追加されます。
- Microsoft Excel などのサードパーティ製品へまたはサードパーティ製品から *構造化されたデータ* コピーして貼り付けることが可能になります。
- XMLSpy のインテリジェントな編集機能をテーブルの表示内で使用することができます。

複数の発生を持つ要素をテーブルとして表示する

Person 要素をテーブルとして表示するには、以下を行います:

1. Person 要素の最初の発生の **テーブルの表示** アイコンを上記の通りクリックします。
2. メニューオプション [XML | **最適な幅**] を選択して、または[グリッドビューツールバー](#) 268 内の **最適な幅** アイコンをクリックします。

メモ テーブル表示は複数の発生を持つすべての要素のためにオン/オフに切り替えることができます。しかしながら、テーブルとして表示されている子要素はテーブルとして表示され続けます。

テーブルの表示内にコンテンツを入力する方法

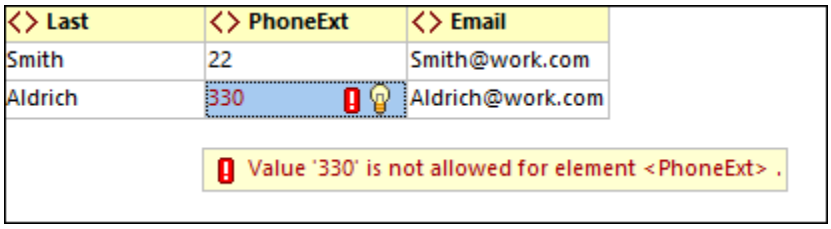
2番目の Person 要素にコンテンツを挿入するには、2番目の行内のテーブルセルのそれぞれをダブルクリックして、データを入力します。(例えば、ブール値コンテンツと Degree 属性の列挙)などのドロップダウンリストからのオプションを選択できるように、テーブルのセル内でもインテリジェントな機能は有効化されています。

	Manager	Degree	Programmer	First	Last	PhoneExt	Email
1	true	BA	false	Fred	Smith	22	Smith@work.com
2	false	MA	true	Alfred	Aldrich	33	Aldrich@work.com

動的な検証

スキーマ内で定義済みの通り PhoneExt はファイルが妥当であるために 0 から 99 の整数である必要があります。XMLSpy の関数をオンに切り替えて編集中に検証することができます。オンに切り替えられると、ファイルはフォーカスが新しいノードに切り替えられる度に検証されます。動的な検証を以下のように試します:

1. メニューコマンド [XML | 編集時に検証]をオンに切り替えます。
2. (99 より大きな値)無効な PhoneExt 値をスクリーンショットで示されるように入力します。
3. 「タブ」キーを押します。PhoneExt セル内にエラーアイコンとスマートフィックスアイコンが表示されます(*下のスクリーンショット参照*)。
4. エラーアイコンにマウスをポイントして検証エラーメッセージを確認します(*下のスクリーンショット*)。

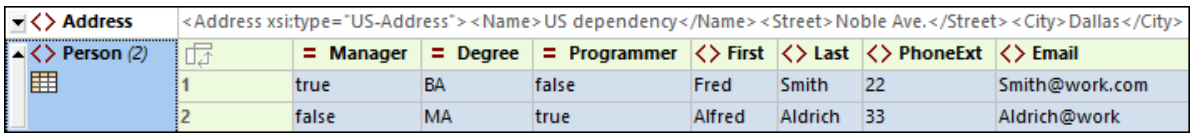


5. スマートフィックスアイコンをクリックするとスマートフィックスオプションが表示されます。無効な値が有効な値と置き換えられ、エラーフラグが消えます。

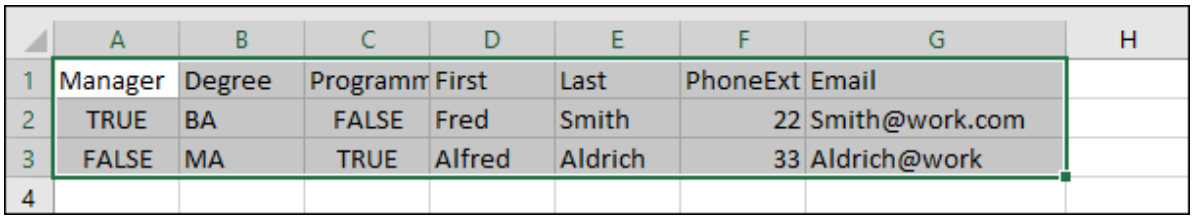
XML データをスプレッドシートアプリケーションへ、およびスプレッドシートからコピーする。

テーブルの表示の場合、MS Excel などのスプレッドシートアプリケーションと交換するためにデータをタブ区切りテキストとしてコピーすることができます。XML ファイルからデータをコピーする場合、以下を行います:

1. Person 要素をクリックします(*下のスクリーンショット参照*)。これにより列ヘッダーとテーブルの行が選択されます。

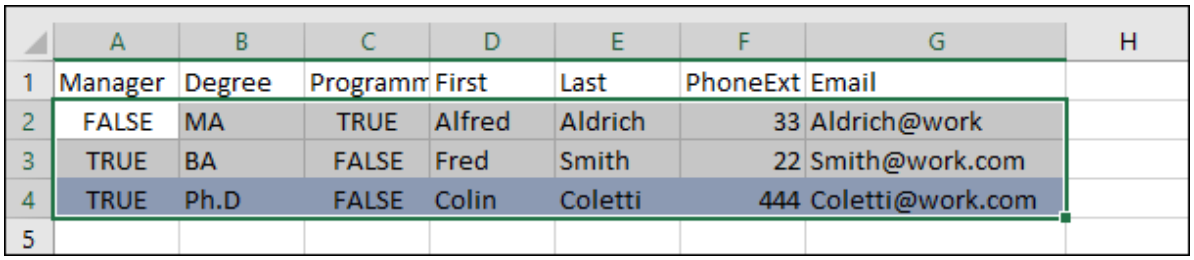


2. 選択範囲内を右クリックして、表示されるコンテキストメニュー内でメニューコマンド [**コピー | タブ区切りテキストとしてコピー**]を選択します。または **[Ctrl+C]** を押します。
3. Excel ワークシートに切り替えセル A1 を選択して XML データを張り付け(**Ctrl+V**) ます。データはテーブル表示内のテーブル構造に対応する行として入力されます(*下のスクリーンショット参照*)。



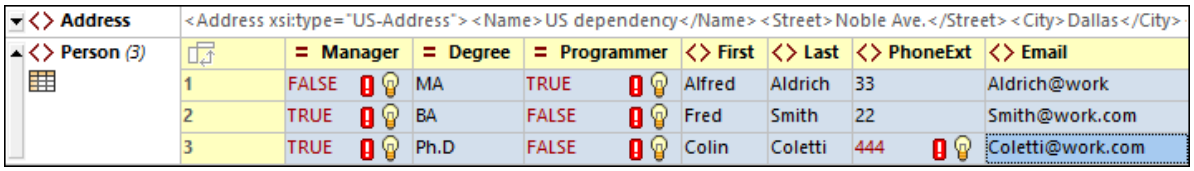
	A	B	C	D	E	F	G	H
1	Manager	Degree	Programm	First	Last	PhoneExt	Email	
2	TRUE	BA	FALSE	Fred	Smith	22	Smith@work.com	
3	FALSE	MA	TRUE	Alfred	Aldrich	33	Aldrich@work	
4								

4. 下のスクリーンショットに示されるようにExcel 内に新規の行を入力します。`PhoneExt` 要素のために（例えば、`444`）3桁の数値を入力します。
5. 列ヘッダーを除いて（下のスクリーンショット内で緑色の枠で表示されている）Excel 内のテーブルデータをマークし **[Ctrl+C]**を使用してコピーします。



	A	B	C	D	E	F	G	H
1	Manager	Degree	Programm	First	Last	PhoneExt	Email	
2	FALSE	MA	TRUE	Alfred	Aldrich	33	Aldrich@work	
3	TRUE	BA	FALSE	Fred	Smith	22	Smith@work.com	
4	TRUE	Ph.D	FALSE	Colin	Coletti	444	Coletti@work.com	
5								

6. XMLSpy 内で [XML | 編集中に検証] コマンドがオンに切り替えられていることを確認してください。
7. XMLSpy 内のXML ドキュメントのテーブルの表示で、最初の行の`Manager` セルを選択して、**[Ctrl+V]** を使用してクリップボードにコンテンツを張り付けます。テーブルは以下のスクリーンショットのようになります。



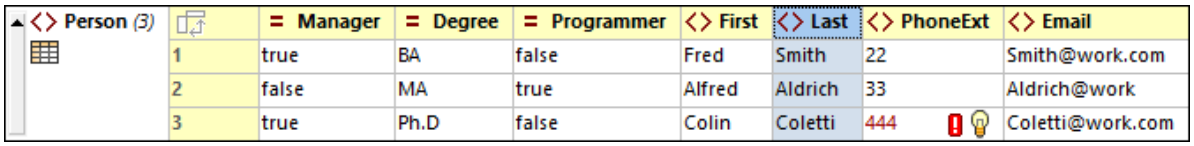
Address	<Address xsi:type="US-Address"><Name>US dependency</Name><Street>Noble Ave.</Street><City>Dallas</City>						
Person (3)	Manager	Degree	Programmer	First	Last	PhoneExt	Email
1	FALSE	MA	TRUE	Alfred	Aldrich	33	Aldrich@work
2	TRUE	BA	FALSE	Fred	Smith	22	Smith@work.com
3	TRUE	Ph.D	FALSE	Colin	Coletti	444	Coletti@work.com

8. ブール値のための検証エラーがXML とExcel 間の大文字/小文字の区別により引き起こされています。これを修正するには、対応するテーブルセルのスマートフィックスを適用してください。

列のコンテンツ上のテーブルを並べ替える方法

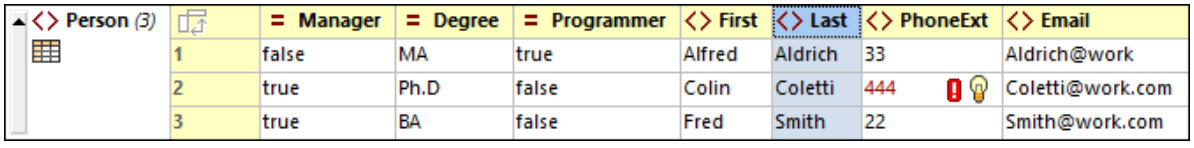
テーブルの表示内のテーブルを全ての列上で昇順または降順に並べ替えることができます。姓（last name）上の`Person` テーブルを並べ替えます。以下の手順を行ってください：

1. `Last` 列をヘッダーをクリックして選択します。



Person (3)	Manager	Degree	Programmer	First	Last	PhoneExt	Email
1	true	BA	false	Fred	Smith	22	Smith@work.com
2	false	MA	true	Alfred	Aldrich	33	Aldrich@work
3	true	Ph.D	false	Colin	Coletti	444	Coletti@work.com

2. メニューオプション [XML | 昇順に並べ替え] を選択して、またはグリッドビューツールバー[268]内の昇順に並べ替えアイコンをクリックします。列と列を含むテーブルがアルファベット順に並べ替えられます。列はハイライトされたままです。



Person (3)	Manager	Degree	Programmer	First	Last	PhoneExt	Email
1	false	MA	true	Alfred	Aldrich	33	Aldrich@work
2	true	Ph.D	false	Colin	Coletti	444	Coletti@work.com
3	true	BA	false	Fred	Smith	22	Smith@work.com

内線番号 444 は正しいですが、妥当ではないためXML スキーマを変更して数値が妥当になるようにします。これは次のセクションで行われます。

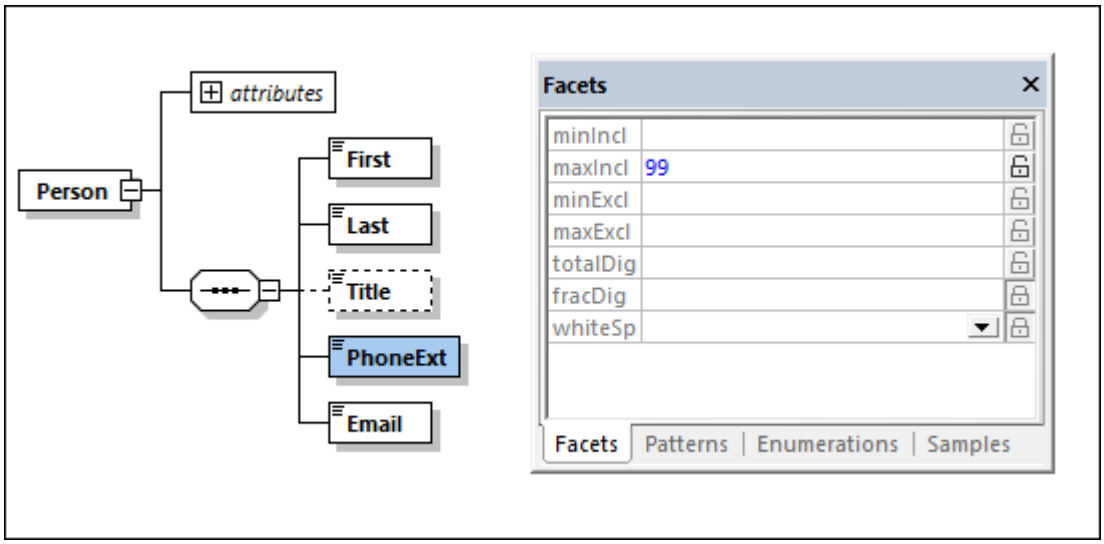
2.5.8 スキーマの修正

XML スキーマ内で定義されている範囲(0 から 99) の外部である内線番号が含まれているため、範囲を999 に広げます。方法は以下で説明されているとおりです。

1. グリッドビュー内で PhoneExt セルを選択します (*下のスクリーンショット*)。

Person (3)	Manager	Degree	Programmer	First	Last	PhoneExt	Email
1	false	MA	true	Alfred	Aldrich	33	Aldrich@work
2	true	Ph.D	false	Colin	Coletti	444	Coletti@work.com
3	true	BA	false	Fred	Smith	22	Smith@work.com

2. メニューオプション DTD/スキーマ | 定義に移動 を選択して[グリッドビューツールバー](#)[268] 内の 定義に移動 アイコンをクリックします。関連付けされたスキーマ、この場合は AddressLast.xsd が開かれ PhoneExt 定義がハイライトされます (*下のスクリーンショット*)。



3. 要素の maxIncl ファセットは 99 です(*スクリーンショット参照*)。この値を 999 に編集し、スキーマを保存します。
4. XML ドキュメントに戻り検証します。妥当なドキュメントになります。
5. ファイルを CompanyLast.xml として保存します。

メモ XMLSpy のチュートリアルフォルダーにはこのチュートリアルを完了した時点で保存されるファイルと同じデータを含む CompanyLast.xml という名前のファイルが含まれています。

2.6 XSLT 変換

目的

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

方法

変換を行うために必要な方法は以下の通りです:

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

このセクションで使用されるコマンド

以下の XMLSpy コマンドがこのセクションでは使用されます:

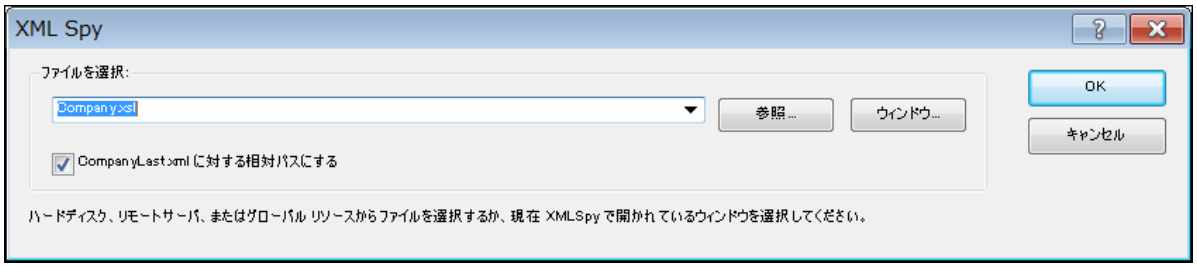
	「XSL/XQuery | XSLの割り当て」: アクティブな XML ドキュメントに XSL ファイルを割り当てます。
	「XSL/XQuery | XSLへ移動」: XML ドキュメントにより参照されている XSL ファイルを開きます。
	「XSL/XQuery | XSL変換 (F10)」、または対応するツールバーアイコン: アクティブな XML ドキュメントをその XML ファイルに割り当てられた XSL スタイルシートを使って変換します。XSL ファイルがまだ割り当てられていない場合、割り当てを行うプロンプトが表示されます。

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

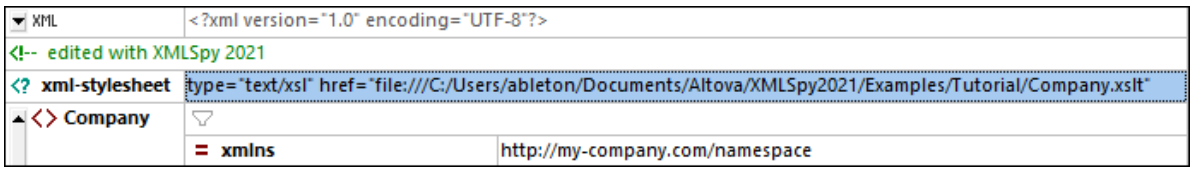
2.6.1 XSLT ファイルの割り当て

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

1. メインウィンドウの `CompanyLast.xml` タブをクリックして `CompanyLast.xml` をアクティブにし、テキストビューに切り替えます。
2. メニューオプションの「XSL/XQuery | XSLの割り当て」を選択します。
3. 「参照」ボタンをクリックして、Tutorial フォルダーから `Company.xsl` ファイルを選択します。XML ドキュメント内でパスを相対的にするためにダイアログ内で、オプション *CompanyLast.xml Mに対してパスを相対的にする* を有効化します。



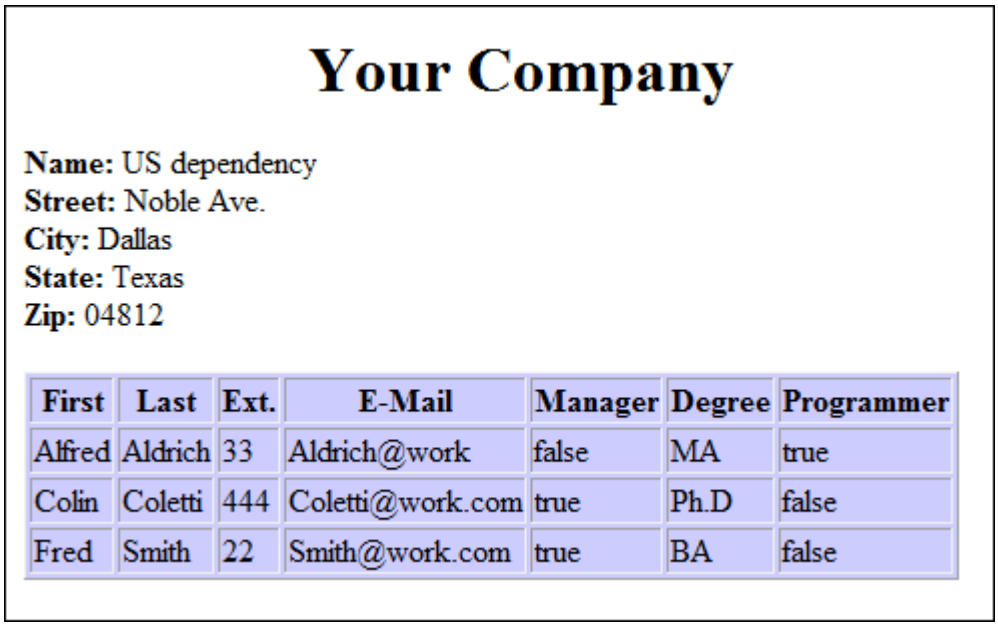
4. 「OK」をクリックしてXSL ファイルをXML ドキュメントに割り当てます。
5. グリッドビューに切り替えて割り当てを確認します(下のスクリーンショットを参照)。XSL ファイルを参照するXML-スタイルシート処理命令がXML ドキュメントに挿入されます。*CompanyLast.xml に対する相対パスにする* チェックボックスが有効化されている場合、パスは相対的になります。(スクリーンショットで表示されている通り)それ以外の場合はパスは絶対パスになります。



2.6.2 XMLファイルの変換

割り当てられたXSL ファイルを使用してXML ドキュメントを変換する方法:

1. XML ファイルがアクティブなドキュメントになっていることを確認します。
2. メニューオプションから「XSL/XQuery | XSL 変換 (F10)」を選択するかコマンドのアイコンをクリックします。この操作によりXML ドキュメントで参照されているXSLT スタイルシートを使用した変換が開始されます出力ドキュメントはブラウザービューに表示されます。このドキュメントには `XSL Output.html` という名前が付けられています。HTML ドキュメントは `Company/Address` データは左側で単一のブロックで、`Company/Person` データはテーブルの書式で表示されています。



Your Company

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

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

メモ `Company.xslt` ファイルはXSLT 3.0 ドキュメントであるためビルトイン Altova XSLT 3.0 Engine が変換のために自動的に選択されます。HTML 出力ファイルが生成されない場合 オプションダイアログ（「ツール | オプション）の*XSL*セクションで出力ファイルのデフォルトのファイル拡張子が `.html` に設定されていることを確認してください。これによりブラウザーが出力ドキュメントを正確にファイルと読み取ることを保証することができます。

2.6.3 XSLファイルの修正

XSL ドキュメントを修正することで、出力ファイルを変更することができます。例えば、HTML 出力のテーブルの背景色を `#ccccff` から `#99cc99` に変更してみましょう。以下の手順を行ってください：

1. `CompanyLast.xml` タブをクリックしてドキュメントをアクティブにします。
2. メニューオプションから「「XSL/XQuery | XSLへ移動」を選びます。このコマンドでXMLドキュメントにより参照されている `Company.xslt` ファイルが開かれます。
3. (*下のスクリーンショットでハイライトされている*) `table` 要素と要素の `bgcolor` 属性の開始タグを検索します。属性の値を `#ccccff` から `#99cc99` に変更します。

```
<xsl:template match="/">
   <html>
      <head><title>Your company</title></head>
         <body>
            <h1><center>Your Company</center></h1>
            <xsl:apply-templates select="//my:Address"/>
               <table border="1" bgcolor="#ccccff">
                  <thead align="center">
                     <td><strong>First</strong></td>
                     <td><strong>Last</strong></td>
                     <td><strong>Ext.</strong></td>
                     <td><strong>E-Mail</strong></td>
                     <td><strong>Manager</strong></td>
                     <td><strong>Degree</strong></td>
                     <td><strong>Programmer</strong></td>
                  </thead>
                  <xsl:apply-templates select="//my:Person"/>
               </table>
         </body>
   </html>
</xsl:template>
```

4. メニューオプション「ファイル | 保存」を選択して変更を保存します。
5. `CompanyLast.xml` タブをクリックして XML ファイルをアクティブにします。
6. メニューコマンド [XSL/XQuery | XSL 変換]を実行する。または、**[F10]**を押します。新しい背景色で表示される新しい `XSL Output.html` ファイルがのブラウザービューに表示されます(*下のスクリーンショット参照*)。

Your Company

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

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

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

(C) 2015-2021 Altova GmbH

2.7 プロジェクト管理

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

2.7.1 プロジェクトの利点

XML ファイルをプロジェクトにより管理すると以下のような利点があります。

- ファイルを共通の条件に基づきフォルダーにグループ分けすることができます。例えば、XML ファイルとXSD ファイルを個別のフォルダーにグループ分けすることができます。希望する階層を作成することができます。
- 各フォルダーには設定可能な特定のプロパティが存在します。例えば、XML ファイルのフォルダーを検証のためにスキーマに割り当てることができます。このプロジェクトフォルダー内のすべてのファイルをフォルダーのスキーマファイルに対してバッチで検証することができます。プロジェクトフォルダーのスキーマ割り当てを変更すると、新規のバッチ検証を素早く実行することができます。単一のXSLT を使用するバッチ変換のためのXSLT ファイルなど他の役に立つフォルダープロパティの一部を設定することができます。
- フォルダーまたはプロジェクト全体に対してバッチ処理を行うことができます。
- DTD またはXML スキーマを指定したフォルダーに割り当て、フォルダー内のファイルに対して検証を行うことができます。
- XSLT ファイルを指定したフォルダーに割り当て、割り当てたXSLT を使ってそのフォルダー内にあるXML ファイルの変換を行うことができます。
- フォルダー全体のためにXSL 変換の出力ファイルの変換先を指定することができます。

上のプロジェクト設定は、全てメニューオプションの「**プロジェクト | プロパティ**」にて行うことができます。プロジェクトコマンドはプロジェクトと個別のプロジェクトフォルダーのコンテキストメニュー内で使用することができます。次のセクションでは、**プロジェクト**メニューを使ってプロジェクトを作成します。

更に、次の高度なプロジェクト機能を使用することができます:

- XML ファイルを[**プロジェクト | ソース管理 | ソース管理に追加**]を使用してソースコントロール下で管理することができます。詳細に関してはソース管理セクション[1171]を参照してください。
- ネットワーク上の外部フォルダー[1186]でもweb フォルダー[1189]をプロジェクトに追加することができます。これにより検証と変換などプロジェクトフォルダーのすべての機能がネットワークまたはインターネット上のフォルダーに適用されることが許可されます。

2.7.2 プロジェクトの作成

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

新規のプロジェクトの作成

新規のプロジェクトを以下のように作成します:

1. メニューオプション[**プロジェクト | 新規のプロジェクト**]を選択します。新しいプロジェクト という名前の新しいプロジェクトがプロジェクトウィンドウに作成されます(*下のスクリーンショット参照*)。新しいプロジェクトには典型的なXML ファイルのカテゴリフォルダーが作成されます。

2. `CompanyLast.xml` タブをクリックし、メインウィンドウの `CompanyLast.xml` ファイルをアクティブにします。
3. メニューオプションから「**プロジェクト | プロジェクトにアクティブならびに関係するファイルを追加**」を選択します。2つのファイルがプロジェクトに追加されます: `CompanyLast.xml` と `AddressLast.xsd`。アクティブなファイルなため XML ファイルは *XML* サブフォルダーに追加されますXML ファイルに含まれ間連するファイルになるためスキーマファイルは *DTD/スキーマ*フォルダーに追加されます。処理命令で参照されるファイル(例えばXSLT ファイル)は関係するファイルとは見なされないことに注意してください。
4. メニューオプションから「**プロジェクト | プロジェクトを保存**」を選択します。プロジェクトを `Tutorial` という名前で保存します。

メモ フォルダー(プロジェクトではなく) フォルダーはそれぞれ *ファイル拡張子*という名前のプロパティを持ちます。これはセミコロンで区切られたファイルの拡張子のリストです(例えば、`xml;svg;wml`)。このリストはファイルがプロジェクトに追加される際どのファイルがどのプロジェクトに追加されるかを決定します。例えば、上で行われているようにアクティブで関連のあるファイルがプロジェクトに追加されると、フォルダーの *ファイル拡張子* プロパティがどのフォルダーに追加されたファイルが配置されるかを決定します。

プロジェクトとフォルダープロパティ

(検証のためのスキーマと変換のためのXSLT など) プロパティをプロジェクト全体だけでなく、個別のフォルダーのために設定することができます。プロジェクト全体または個別のフォルダーのために検証と変換などのアクションを実行することができます。アクションを実行するには、プロジェクトまたはフォルダーを右クリックして、表示されるコンテキストメニューから実行するアクションを選択します。

以下の点に注意してください:

- フォルダーで設定されたプロパティはプロジェクトの同じプロパティをオーバーライドします。
- プロパティがプロジェクトで設定されている場合、同じプロパティが設定されていないすべてのフォルダーに適用されます。
- プロジェクトでアクションが実行されると、プロジェクトの全てのフォルダー内の全ての適用可能なファイル型に適用されます。例えば、検証がプロジェクトで実行さえる場合、検証はプロジェクトの全てのフォルダー内の全てのXML ファイルで実行されます。この場合、プロジェクトのために設定されたスキーマは他のスキーマにスキーマ検証プロパティが設定されているフォルダー内のXML ファイル以外のすべての検証のために使用されます。

プロジェクトにファイルを追加する

プロジェクトには他のファイルを追加することもできます。以下の手順を行ってください:

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

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

その他の役に立つコマンド

その他にも良く使われる以下のプロジェクトコマンドがあります:

- 新しいフォルダーをプロジェクトに保存するには、「**プロジェクト | プロジェクトにプロジェクトフォルダーを追加**」を選択して、プロジェクトフォルダーの名前を入力します。
- プロジェクトのフォルダーを削除するには、フォルダーを右クリックして、コンテキストメニューから「**削除**」を選択します。
- プロジェクトのファイルを削除するには、ファイルを右クリックして「**削除**」キーを押下します。

2.8 おしまい

ここまでくれば、おめでとう。そしてありがとう！

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

3 インターフェイスと環境

このセクションでは以下の説明を行ないます:

- アプリケーション GUI 108
- アプリケーション環境 122

GUI のセクション 108 では、最初に GUI の概要を説明した後、各 GUI ウィンドウの詳細について解説していきます。ウィンドウならびに GUI のサイズ変更、移動、そして操作方法などについても説明します。

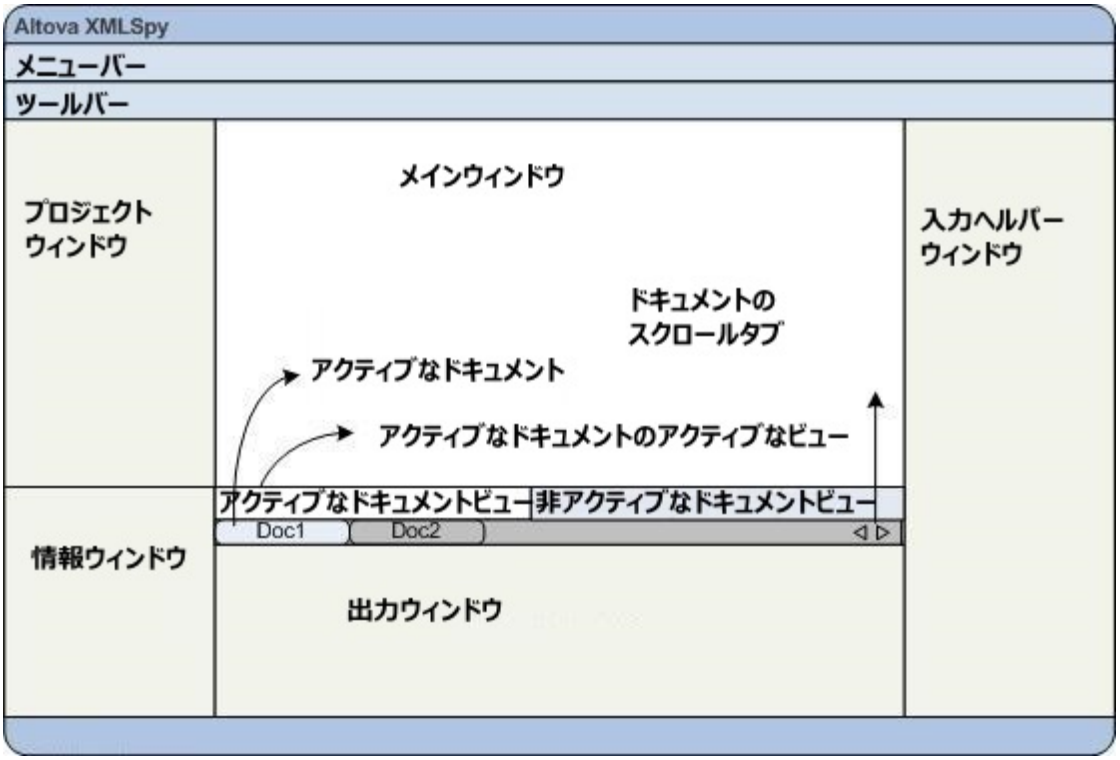
アプリケーション環境のセクション 122 では、ファイルの管理を行うために必要な、様々な設定の表示ならびに変更方法について説明します。アプリケーションのカスタマイズを行う場所と、その方法についても記述されます。このセクションでは、あなたのコンピューター内にインストールされている、重要なサンプルファイルやチュートリアルファイルの場所についても言及されています。セクションの最後には Altova W eb サイトへのリンクが掲載されており、アプリケーションが持つ機能の一覧や、様々な形式のユーザーマニュアル、利用可能なサポートオプション、そして Altova が提供するその他の製品に関する情報を参照することができます。

3.1 グラフィカルユーザインターフェース (GUI)

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

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

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



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

サイドバー表示の切り替え

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

出力ウィンドウの1つを閉じると、メニューコマンド「ウィンドウ | 出力ウィンドウ」をクリックして元に戻すことができます。

サイドバーのフロートとドッキング

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

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

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

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

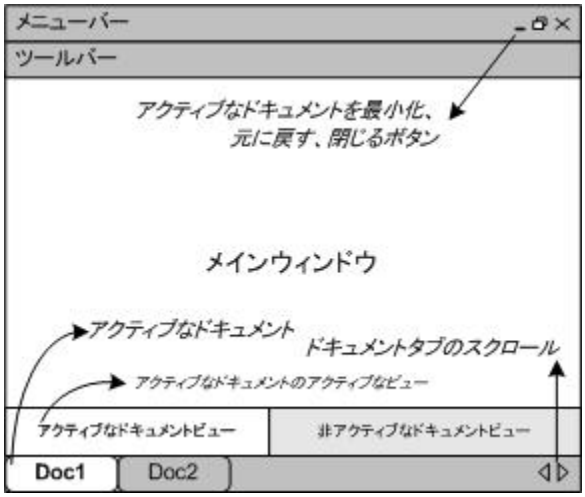
サイドバーを自動的に隠す

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

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

3.1.1 メイン ウィンドウ

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



メインウィンドウのファイル

- 開いたり編集を行うファイルの数に制限はありません。
- 開かれた各ドキュメントは、各自ウィンドウに表示され、その名前がメインウィンドウの下部にあるタブに表示されます。ドキュメントをアクティブにするには、そのドキュメントのタブをクリックしてください。

- 複数のファイルが開かれている場合、ドキュメントタブバーに収まりきらず、一部のドキュメントタブが表示されないことがあります。目的のドキュメントタブは、(i) ドキュメントタブバーの右にあるスクロールボタンを使うことで、または(ii) ウィンドウ[1493] メニューから目的のドキュメントを選択することでアクティブにすることができます。
- アクティブなドキュメントが最大化されると、「最小化」「元に戻す」、「閉じる」ボタンがメニューバーの右側に配置されます。ドキュメントが重ねて表示、または最小化されると、「最大化」、「元に戻す」、「閉じる」ボタンがドキュメントウィンドウのタイトルバーに配置されます。
- 1つのファイルを最大化すると、開かれている他のファイルも全て最大化されます
- ウィンドウ[1493] メニューのコマンドを使用することで、開かれているドキュメントを重ねて、または上下(左右)に並べて表示することができます。
- 開かれているファイルは、「**Ctrl+Tab**」または「**Ctrl+F6**」により、開かれた順序でアクティブに切り替えることができます。
- ドキュメントタブを右クリックすることで表示されるコンテキストメニューから、「印刷」や「閉じる」といったコマンドを実行することができます。

メインウィンドウ内のビュー

アクティブなドキュメントを表示、編集する方法は何通りかあります。使用することのできるビューは、(上の図にあるように)ドキュメントタブ上のバーに表示されており、アクティブなビューがハイライトされます。目的のビューボタンをクリック、または表示[1493] メニューのコマンドを使用することでビューの切り替えを行うことができます。

利用可能なビューは編集またはブラウザビューです:

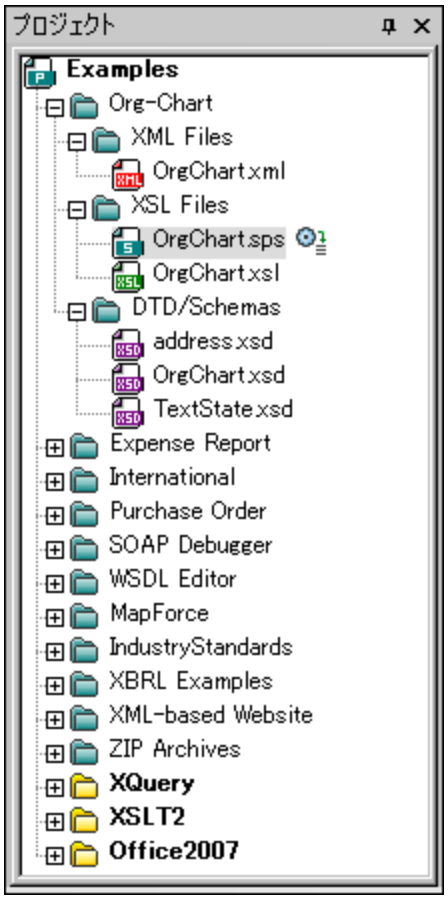
- テキストビュー[130]: 構文による色分けなどの機能を備えたソースコードレベルの編集を行うビュー
- グリッドビュー[149]: 構造化された編集を行うビュー。グラフィカルな編集を行うことのできる構造化されたグリッドの形式で、ドキュメントは表示されます。このビューにはテーブルビュー[151] が内蔵されており、繰り返し出現する要素をテーブル形式で表示することができます。
- スキーマビュー[126]: XML スキーマの表示や編集を行ないます。
- WSDL ビュー[126]: WSDL ドキュメントの表示と編集を行ないます。
- Authentic ビュー[540]: StyleVision Power Stylesheet をベースにした XML ドキュメントの編集をグラフィカルなインターフェイスで行ないます。
- ブラウザビュー[256]: CSS ならびに XSL スタイルシートをサポートする統合されたブラウザビューです。

メモ 各ファイル拡張子のデフォルトビューは、「ツール | オプション[1454]」ダイアログの、ファイルタブにあるデフォルトビューペインにてカスタマイズすることができます。

3.1.2 プロジェクト ウィンドウ

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

メモ サンプルプロジェクトはデフォルトでプロジェクトウィンドウ内で開かれます。開く必要がある場合、アプリケーションの*ドキュメントフォルダー*[33] 内の**Examples** フォルダーに移動してファイル**Examples.spp** をダブルクリックします。



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

プロジェクトの操作

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

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

- プロジェクトウィンドウでは、フォルダーを他のフォルダー、または、同じフォルダー内の他のロケーションにドラッグすることができます。ファイルは他のフォルダーにドラッグすることができますが(アルファベット順にファイルが並べられている)同じフォルダー内では移動することができません。また、Windows エクスプローラーなどからファイルやフォルダーをプロジェクトウィンドウにドラッグすることで、プロジェクトへの追加を行うこともできます。
- 各フォルダーには、フォルダーのプロパティダイアログにて定義されるプロパティがあります。これらのプロパティには、フォルダーのファイル拡張子、XML ファイルの検証を行うスキーマファイル、XML ファイルの変換を行う XSLT ファイルなどが記述されます。
- フォルダー内にあるファイルのバッチ処理を行うには、フォルダーを右クリックし、コンテキストメニューから選択することのできるコマンドを選択します(例: **XML の検証**または**整形式のチェック**)。

更に詳細なプロジェクトの使用方法に関しては、プロジェクト[950]のセクションを参照ください。

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

3.1.3 情報ウィンドウ

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

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

以下の点に注意してください:

- XSLT ファイルがアクティブな場合、情報ウィンドウの XSLT タブには XSLT 固有の情報やコマンドが表示されます(以下のスクリーンショットの XSLT タブを参照)。XSLT タブに表示される情報やコマンドの読み方については、XSLT と XQuery | XSLT | XSL アウトライン[437]のセクションを参照ください。
- JSON ドキュメント (*インスタンス*または*スキーマ*) がアクティブな場合、情報ウィンドウの JSON タブ内で返照とインテリジェントな編集のためのオプションを使用することができます。詳細については JSON ドキュメントの検証[662] を参照してください。
- XML スキーマドキュメント(.xsd ファイル)がアクティブな場合、情報 ウィンドウのスキーマタブにて、拡張スキーマ検証を有効にすることができます。拡張検証の設定方法に関しては、スキーマルール[391]のセクションを参照ください。
- XULE ドキュメントがアクティブなドキュメントの場合、情報ウィンドウの XULE タブが XBRL *インスタンス*の構造に関する情報のために使用する XBRL タクソノミを選択するオプションを提供します。タクソノミ情報は XULE エディター[821]の自動補完機能のために使用されます。

3.1.4 入力ヘルパー

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

利用可能な入力ヘルパーは以下の条件により変化します:

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

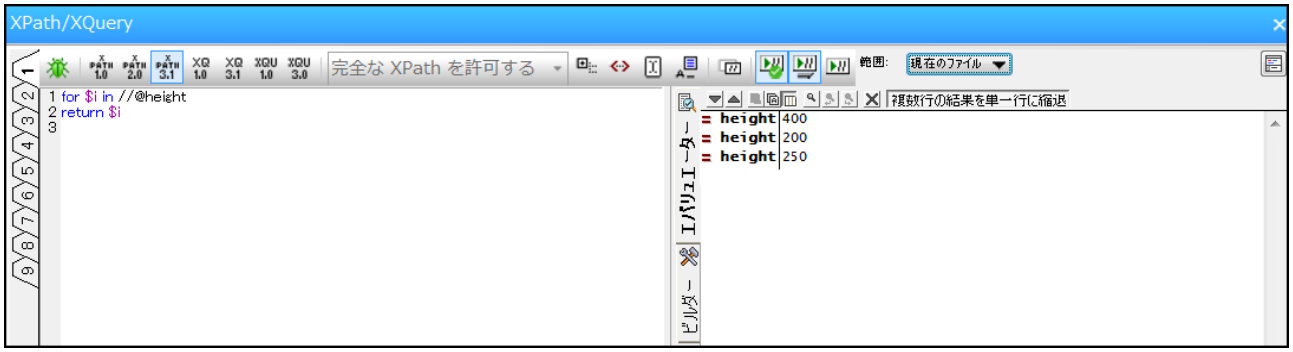
各ビューにおける入力ヘルパーの詳細については編集ビュー[126]のセクションを参照ください。異なるドキュメントの種類と入力ヘルパーの関係については、(例えば、XML 入力ヘルパー[301] や XQuery 入力ヘルパー[445] といった) 各ドキュメントの記述を参照ください。

以下の点に留意してください:

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

3.1.5 出力ウィンドウ: XPath/XQuery

XPath/XQuery ウィンドウ(*下のスクリーンショット*) により、XML または JSON ドキュメントに関連した XPath と XQuery 式を作成、評価、デバッグすることができます (JSON クエリを有効化する機能は XPath/XQuery 3.1. で導入されました)。



このセクションは XPath/XQuery ウィンドウの主要な機能の概要を説明しています。XPath/XQuery ウィンドウとの作業方法に関する詳細は、XPath/XQuery 式[506] セクションを参照してください。

主要な機能

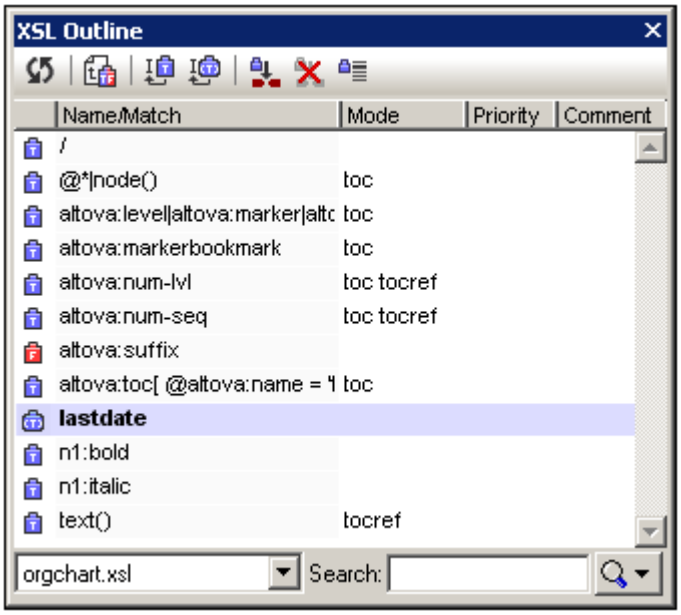
XPath/XQuery ウィンドウには以下の主要機能が搭載されています:

- *評価モード:* 評価モード内では、XPath または XQuery 式は1つまたは複数の XML/JSON ドキュメントに対応して評価されます。式は式ペインに入力され、横の結果ペイン内で結果が表示されます。結果内のノードをクリックして、XML または JSON ドキュメント内でそのノードに移動することができます。
- *デバッグモード:* デバッグモード内では、現在アクティブな XML ドキュメント内で XPath/XQuery 式をデバッグすることができます。ブレークポイントとトレースポイントを設定することができ、評価をステップバイステップで行うことができます。各ステップで、評価の追加アスペクトをチェックするためにカスタムウォッチを設定し変数のコンテンツを確認することができます。
- *式ビルダー:* 式ビルダーには文法的に正確な式の作成を手助けする入力ヘルパーと情報ポップアップが搭載されています。
- *複数の言語のためのサポート:* XPath 1.0 から XPath 3.1 と XQuery 3.1 に言語バージョンを切り替えることができます。
- *ターゲット言語のための柔軟なスコープ:* 式は現在アクティブな XML または JSON ドキュメント、または、ドキュメントのユーザー定義セット上でテストすることができます。
- *ファイル型の自動検知:* 現在のドキュメント (XML または JSON) のファイル型は自動的に検出され、正確なターゲットドキュメントのモードが自動的に設定されます。スコープがドキュメントのセットの場合、ターゲットドキュメントの型を手動で選択することができます。
- *ドキュメントとドキュメントノードへ直接移動することのできるリンク:* 結果ペイン内のラインには対応するドキュメント、またはドキュメントノードへのリンクが含まれています。これにより特定のノードに直接移動し、その箇所に存在するデータをチェックすることができます。
- *デバッガーのアナリティクス:* デバッグモード内では、分析情報が広範囲により表示されます。更に、評価の追加アスペクトをチェックするカスタム条件式を設定することができます。

XPath/XQuery ウィンドウ機能の詳細に関しては、XPath/XQuery 式[506] を参照してください。

3.1.6 出力ウィンドウ: XSL アウトライン

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

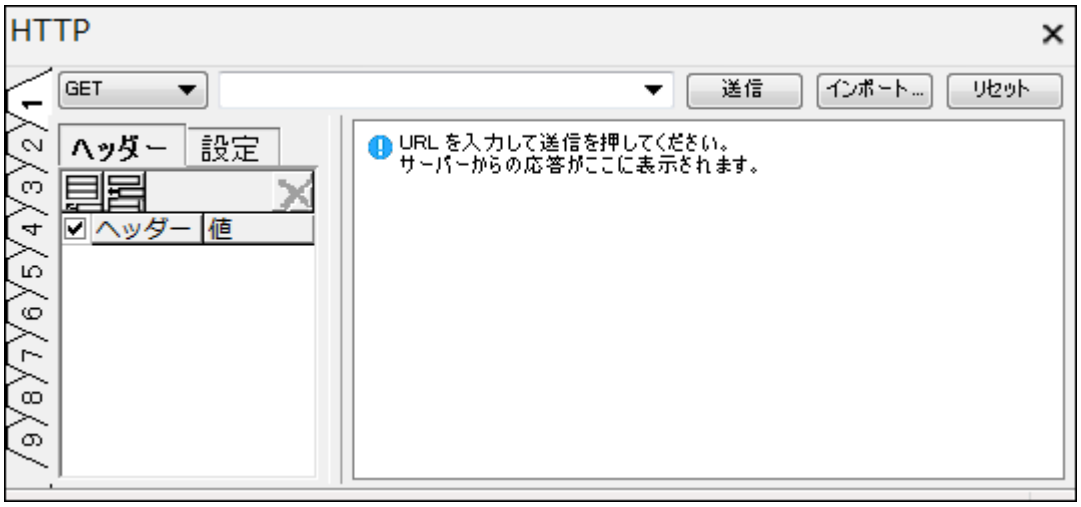


XSL アウトラインウィンドウでは、スタイルシート内のテンプレートや関数に関する情報が表示されます。ウィンドウでは、これら情報の並べ替えや検索を行い、ツールバーにあるコマンドを使用することで名前付きテンプレートに呼び出しを挿入し、名前付きテンプレートを変換の開始点にセットすることができます。XSL アウトラインウィンドウの使用方法に関しては、XSLT とXQuery | XSLT | XSL アウトライン | XSL アウトラインウィンドウ 434 のセクションを参照ください。

メモ スタイルシートのファイルに関連した情報やコマンドは、Info ウィンドウの XSLT タブから使用することができます。これらコマンドの使用方法については、XSLT とXQuery | XSLT | XSL アウトライン | Info ウィンドウ 437 のセクションを参照ください。

3.1.7 出力ウィンドウ: HTTP

HTTP 出力ウィンドウには9つのタブが存在します (下のスクリーンショット参照)。各タブ内に個別のリクエストを保管することができ、タブ間を切り替えることが出来ます。ウィンドウ内でリクエストを作成した後、「送信」ボタンをクリックしてリクエストを送信することができます。レスポンスは直接ウィンドウ内に表示されます。



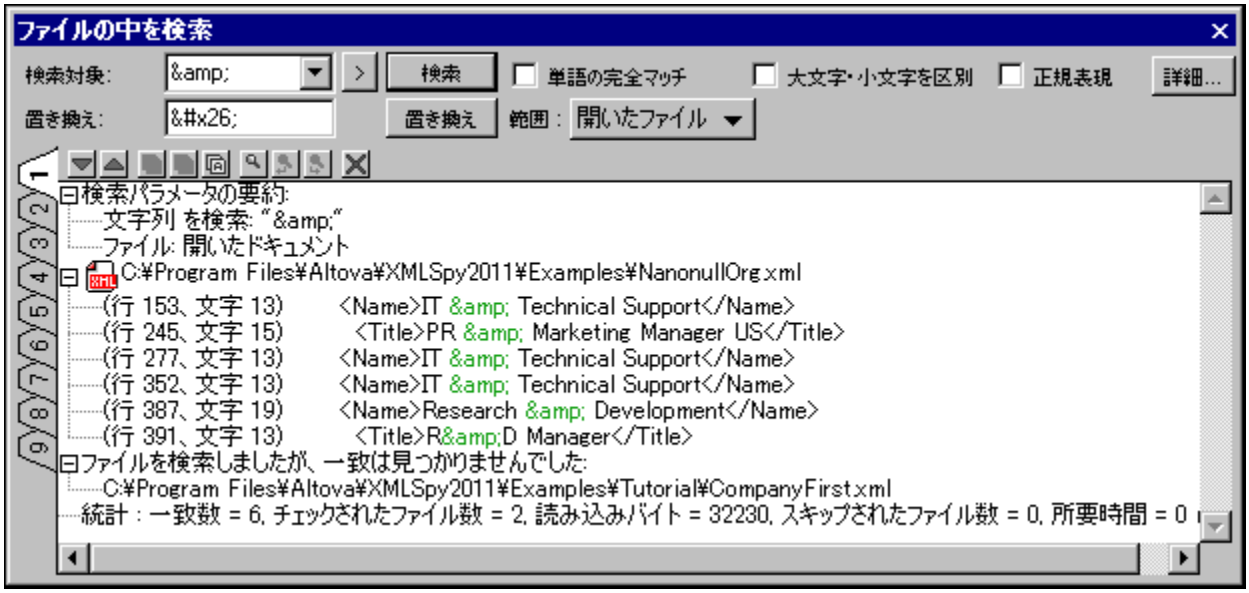
ウィンドウは以下のパートにより構成されています:

- 上の部分: (i) 使用するHTTP メソッドを選択するコンボボックス。(ii) W eb サーバーのURL のための入力フィールド。(iii) HTTP リクエストの実行に関連したボタン (「送信」、「インポート」、および「リセット」)。
- リクエストの作成 712 のための左側のペイン
- レスポンスの表示 718 のための右側のペイン

HTTP 出力ウィンドウの使用方法に関しては、HTTP 711 のセクションを参照してください。

3.1.8 出力ウィンドウ: ファイルの中を検索

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



検索条件

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

検索する対象 : 検索を行う文字列は、検索対象テキストボックスに入力します。検索の対象が、単語として完全に一致する必要がある場合、単語の完全マッチチェックボックスにチェックを入れます。例えば、fit という検索文字列に対して単語の完全一致オプションを有効にすると、fit という単語に対してだけ検索が行われ、例えばfitness という単語は検索結果に含まれません。検索において大文字・小文字の区別をしたい場合は、大文字・小文字を区別チェックボックスにチェックを入れます。検索テキストボックスに入力された検索文字列が正規表現となる場合、正規表現チェックボックスにチェックを入れる必要があります。 ボタンをクリックすることで、正規表現の入力を簡単に行うことができるようになります。検索における正規表現の使用方法に関しては、検索[1159]のセクションで説明されます。詳細ボタンをクリックすることで、ファイルの中を検索[1164]が表示され、検索における高度な条件やアクションをセットすることができます。詳細については、編集 | ファイルの中を検索[1164] を参照ください。

検索範囲 : 検索は、(i) GUI 上に開かれている全てのファイル、(ii) 現在のプロジェクトにあるファイル、または (iii) 選択されたフォルダーのファイルに対して行うことができます。詳細ボタンをクリックすることで表示されるファイルの中を検索ダイアログ[1164]で、更に詳細な条件を追加することができます。

置き換え

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

結果

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

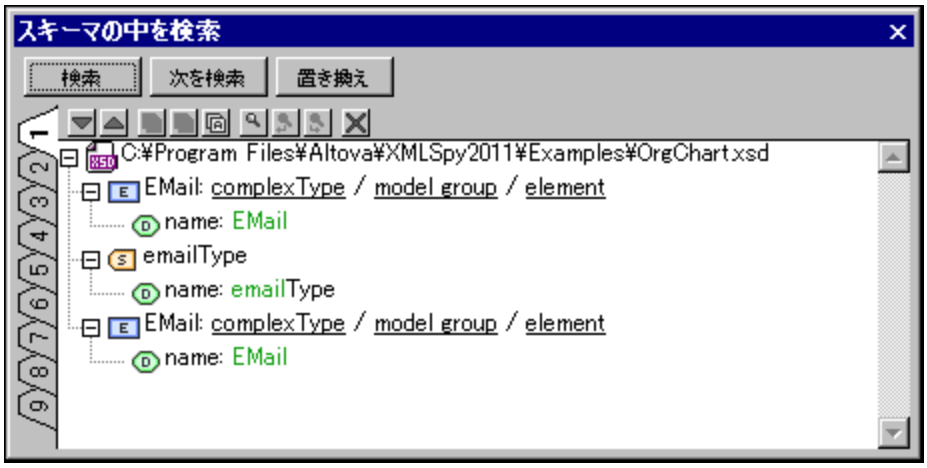
- 検索文字列や検索対象となった範囲を表示する、検索パラメータの要約。
- 検索または置き換えられた文字列のリスト(検索、または「置き換え」ボタンが押下により変化)。このリストに表示されるアイテムは、検索または置き換えがなされた箇所へのリンクとなり、クリックすることでドキュメントの該当箇所がテキストビューで表示されます。ドキュメントが開かれていない場合、そのドキュメントがテキストビューで開かれ、該当箇所がハイライトされます。

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

メモ ファイルの中を検索機能では、複数のファイルに対して、一度に**検索**や**置き換え**コマンドを実行し、その結果を出力ウィンドウに表示します。結果を逐次確認しながら進んでいく検索を行うには、検索[1159]コマンドを使用してください。

3.1.9 出力ウィンドウ: スキーマの中を検索

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



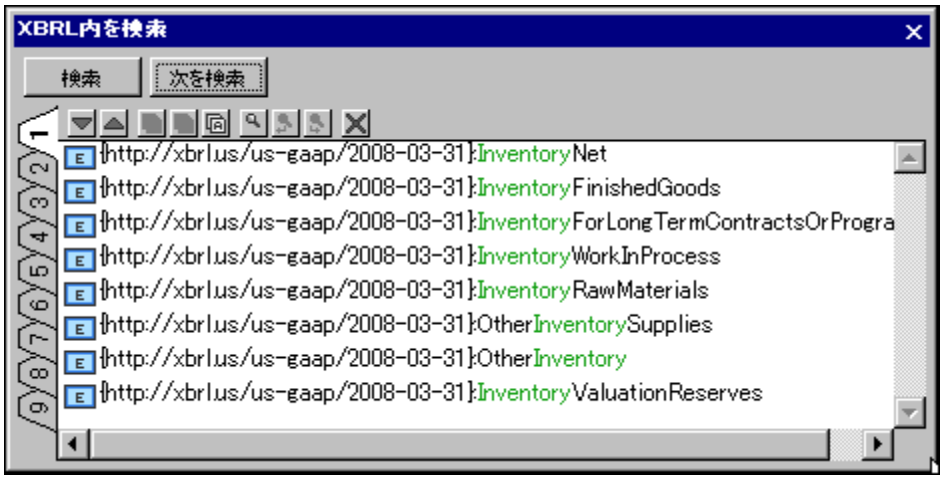
スキーマビューでの検索ならびに置き換えの結果は、スキーマの中を検索出力ウィンドウに表示されます。検索文字列は緑色で表示されます (上のスクリーンショットでは、email が検索文字列で、大文字・小文字の区別をしないように指定されたことが分かります)。スキーマファイル内の位置も示されていることにご注目してください。

結果は (1から9と番号の振られた) 9つあるタブに表示されます。行われた検索の結果を1つのタブに保持しておき、新たな検索の結果を別のタブにて行うことで、検索結果を比較することができます。検索を行う前に新しいタブを選択することで、新たなタブにて検索が行われます。スキーマの中を検索ウィンドウ内に表示される結果をクリックすることで、メインウィンドウのスキーマビューにて、該当する箇所が表示、ハイライトされます。出力タブに表示されるメッセージはクリップボードにコピーすることもできます。詳細については、結果と情報[425] 機能の、スキーマ内の検索[415] のセクションを参照ください。

3.1.10 出力ウィンドウ: XBRL 内を検索

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

XBRL 内を検索する機能は、XBRL タクソノミがXBRL ビューで開かれているときに使用することができます。この検索機能の使用方法については、ユーザマニュアルの XBRL セクション[721] にある XBRL 内を検索[829] を参照ください。



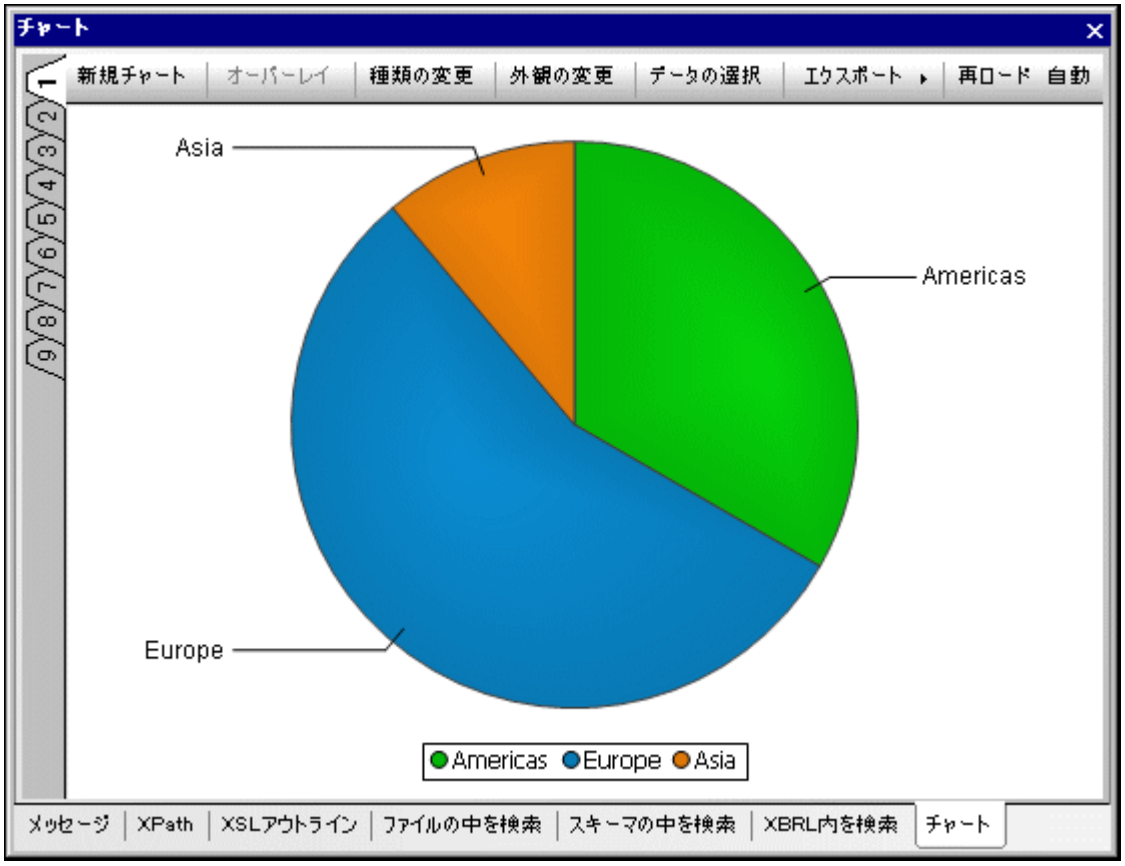
検索ボタンをクリックすることで、検索ダイアログが表示されます。検索が一度行われた後、次を検索ボタンをクリックすることで、XBRL ビューにて現在カーソルが位置している箇所を起点に、検索文字列に一致する次のインスタンスが出力ウィンドウに表示されます。

以下のコマンドがXBRL 内で検索ツールバーで使用できます。

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

3.1.11 出力ウィンドウ : チャート

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



チャートの作成

チャートを作成するための大まかなステップは以下のとおりです:

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

詳細については、ユーザマニュアルの[XML セクション](#) 261 にある、[チャート](#) 310 を参照ください。

チャートの修正と管理

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

ウィンドウの上部にあるボタンは以下のとおりです:

- *新規チャート* : カラムの選択ダイアログが表示され、チャートデータテーブルの設定を行うことができます。

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

詳細に関しては、ユーザーマニュアルの XML セクション[261] のチャート[310] セクションを参照してください。

3.1.12 メニューバー、ツールバー、ステータスバー

メニューバー

メニューバー (*図を参照*[108]) には、様々なアプリケーションメニューが収められています。以下の慣習が適用されます:

- メニュー内のコマンドが、その時のビューや、カーソルの位置に対して適用できない場合、コマンドは無効になります。
- いくつかのメニューコマンドはサブメニューを持っており、更に別のオプションを選択することができます。サブメニューがあるメニューコマンドは、コマンド名の右側に表示される右方向矢印により示されます。
- いくつかのメニューコマンドはサブメニューを持っており、更に別のオプションを選択することができます。サブメニューがあるメニューコマンドは、コマンド名の右側に表示される右方向矢印により示されます。
- メニューコマンドにアクセスするには、メニュー名をクリックした後に、コマンドを選択します。メニューアイテムにサブメニューがある場合、そのメニューアイテムにマウスを移動することで、サブメニューが開かれます。その後、目的のサブメニューアイテムをクリックします。
- 適切なキーのコンビネーションにより、キーボードからメニューを開くこともできます。各メニューへのキーコンビネーションは **Alt+*KEY*** で、***KEY*** がメニュー名のキーになります。例えば、「**ファイル(F)**」メニューへのアクセスを行うには、**Alt+F** を押下します。
- (i) メニューに対するキーの組み合わせ(前の項目を参照)の後に、(ii) 特定のコマンドに対するキーの組み合わせを入力することで、(メニュー内に収められているコマンドの) メニューコマンドへアクセスすることができます。例えば、新規ファイルを作成(「**ファイル(F) | 新規作成(N)**」)するには、**Alt+F** を押下した後、**Alt+N** を押下します。
- 特定のメニューコマンドは、**ショートカットキー**または、キーのコンビネーション(**Ctrl+*Key***)により直接選択することができます。ショートカットが割り当てられたコマンドは、コマンド名の右側にショートカットキーや、キーのコンビネーションが示されます。例えば、ショートカットキーのコンビネーションである **Ctrl+N** を使用することで、ファイルの新規作成を行うことができ、**F8** により XML ファイルの検証を行うことができます。カスタマイズダイアログ(「**ツール | カスタマイズ**」)にあるキーボードタブにて、独自のショートカットを作成[1443] することもできます

ツールバー

メニューバー (*図を参照*[108]) には、メニューコマンドを選択するためのショートカットとなるアイコンが収められています。マウスポインターをアイコン上に移動させると、コマンドの名前が表示されます。アイコンをクリックすると、コマンドが実行されます。

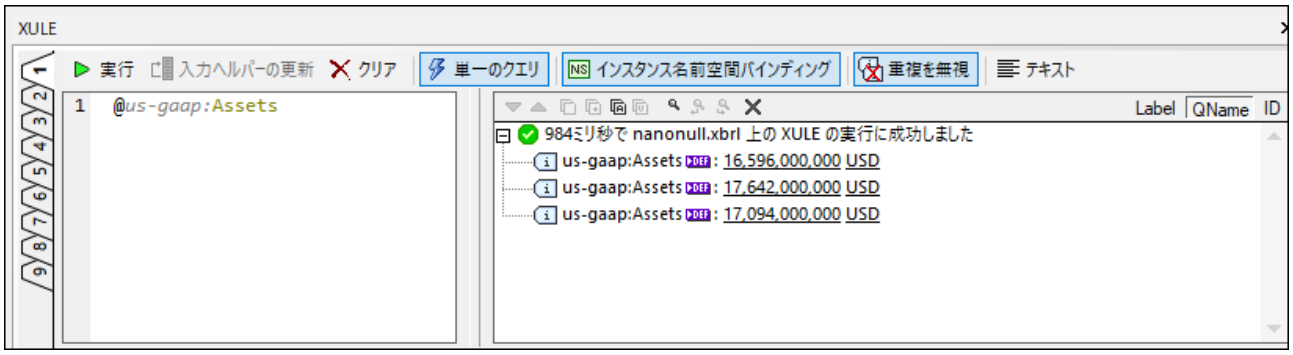
ツールバーボタンはグループでまとめられています。「ツール | カスタマイズ | ツールバー[1440]」ダイアログでは、表示するツールバーのグループを選択することができます。これらの設定は現在のビューに対して適用されます。他のビューの設定を変更するには、そのビューに切り替えた後、「ツール | カスタマイズ | ツールバー[1440]」にて設定を変更します。GUI では、ツールバーグループのハンドル(またはタイトルバー)を別の位置にドラッグすることで、ツールバーグループの位置を変更することができます。ハンドルをダブルクリックすると、ツールバーがドッキングからフロート状態になり、タイトルバーをダブルクリックすることで、元の場所にドッキングされます。

ステータスバー

ステータスバーはアプリケーションウィンドウの下部に位置(*図を参照*[108])しており、(i) ファイルのロードに関する情報や、(ii) マウスポインターがメニューコマンドや、コマンドショートカットにある時に、それらコマンドに関する情報を表示します。64ビット版のXMLSpyを使用している場合、XMLSpy の後に (x64) という表記が加わります。この表記は32ビットバージョンにおいてはなされません。

3.1.13 出力ウィンドウ: XULE

XULE ウィンドウによりアクティブなインスタンスドキュメントをインタラクティブにクエリし、クエリの結果を確認することができます。XULE ウィンドウには9つのタブが存在し、それぞれが2つのペインに分割されています: (i)実行するXULE 式(またはXULE ルール)を挿入する XULE 式ペイン、および(ii) 実行の結果を表示する結果ペイン。



XULE 式をアクティブな XBRL インスタンスドキュメント上でインタラクティブに実行するには以下を行います:

1. クエリする XBRL インスタンドキュメントをメインウィンドウ内でアクティブなドキュメントにします。
2. XULE 式ペイン(*左側のペイン*)内にXULE 式を入力します.
3. 式を実行するためにウィンドウのツールバー内の「実行」をクリックします。
4. 実行の結果は結果ページ内に表示されます (*右側のペイン*).

詳細に関しては XBRL セクション内のXULE ウィンドウ[824] を参照してください。

3.2 アプリケーション環境

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

このセクションは以下のように構成されます:

- *設定、カスタマイズ*[122]: 重要な設定やカスタマイズを行う場所と、その変更方法について説明します。
- *チュートリアル、プロジェクト、サンプル*[124] アプリケーションパッケージに収められている、様々な非プログラムファイルの場所と、その注釈について記述します。
- *製品機能とドキュメンテーション、Altova 製品*[125]: : 製品の機能や、その他の形式で記述されたヘルプ、その他の Altova 製品に関する情報が記述されたAltova W eb サイトへのリンクが掲載されます。

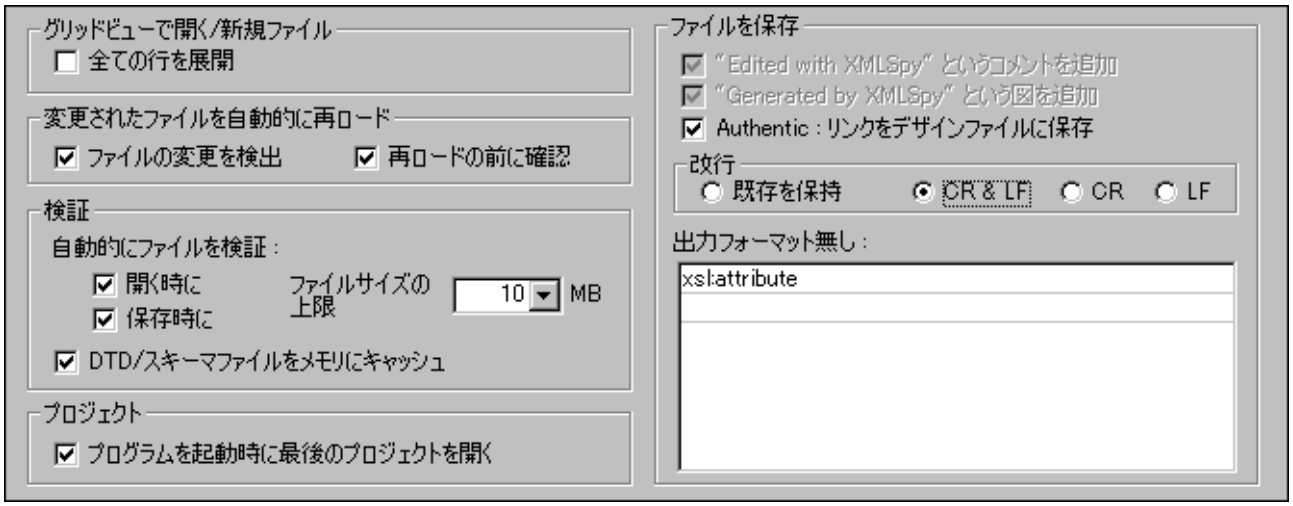
3.2.1 設定、カスタマイズ

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

- 設定[122]
- カスタマイズ[123]
- メニュー[123]

設定

オプションダイアログのタブには、XMLSpy が持つ重要な設定が定義されています（以下のスクリーンショットはメニューコマンド「ツール | オプション[1454]」によりアクセスすることができます）利用可能なオプションがどのようなものか確認してください。

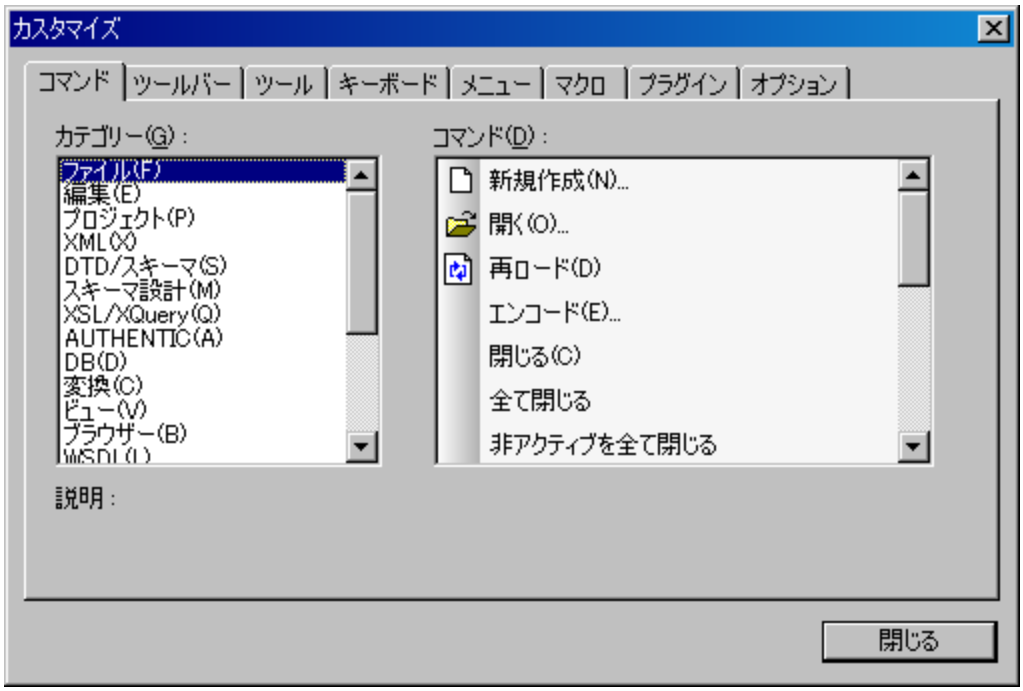


以下は最も重要な設定の要約です。詳細については、ユーザリファレンスにあるオプションダイアログ[1454] の記述を参照ください。

- *ファイルタイプとデフォルトビュー* : ファイルタイプタブでは、XMLSpy が認識するファイルの種類を追加することができます。新たなファイル拡張子をリストの中に追加してください。各ファイルタイプに対して、そのファイルが準拠する仕様や、その種類(拡張子)のファイルを開く際に使用される XMLSpy のビュー(デフォルトビュー)を指定します。
- *ファイルの検証* : ファイルタブでは、ファイルを開く、保存する際に、ファイルの検証を自動的に行うか設定することができます。ファイルタイプタブでは、指定したファイルの種類の検証を無効にすることができます。
- *編集機能* : 編集タブでは、入力ヘルパーの管理方法や、新規要素の生成、更に自動補完を有効にするなどの設定を行うことができます。ビュータブにおいても、その他の設定を個別に行うことができます。各ビューに対するフォントタブでは、これらビュー内部におけるノードの各種類のフォント設定を行うことができます。
- *XSLT と FO エンジン* : XSL タブでは、GUI から行われた変換で使われる外部 XSL エンジンを指定することができます。XMLSpy. 内部から FO 処理を行う際に使用される FO プロセッサーの場所も指定する必要があります。詳細については、XSLT 処理[431] のセクションを参照ください。
- *エンコーディング* : XML ならびに非 XML ファイルのエンコーディングはエンコードタブにより指定されます。

カスタマイズ

ユーザーは GUI の外観など、XMLSpy の様々な側面をカスタマイズすることができます。これらのカスタマイズオプションは、カスタマイズダイアログからアクセスすることができます (以下のスクリーンショットを参照) メニューコマンド「ツール | カスタマイズ[1438]」からアクセスすることができます)。



カスタマイズコマンドに関する詳細は ユーザーレファレンス[1438] のセクションを参照ください。

メニュー

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

- *ファイルの種類* : コマンド「DTD/スキーマ | 他の DTD の取り込み」は、アクティブなファイルが DTD でないと有効になりません。同様に、WSDL メニューのコマンドも、WSDL ファイルがアクティブでないと有効になりません。

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

メモ: メニューのカスタマイズ(「ツール | カスタマイズ[1438]」)では、アイテムをGUIにドラッグすることで、コマンドの再構成を行うことができます(メニューバー、ツールバーステータスバー[120]を参照)。

3.2.2 チュートリアル、プロジェクト、サンプル

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

チュートリアル、プロジェクトサンプルファイルの場所

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

```
C:\Documents and Settings\<username>\My Documents\
Altova\XMLSpy2021\Examples\
```

The `My Documents\Altova\XMLSpy2021`フォルダーは、PC 上で登録されたユーザーごとに、ユーザーの<username> フォルダー以下にインストールされます。従って、このインストールシステムでは、各ユーザーが、それぞれの場所に`Examples` フォルダーを持つことになります。

マスターXMLSpy フォルダーに関する注意

XMLSpy がインストールされると、以下の場所に、マスター `Altova\XMLSpy2021` フォルダーが次の場所に作成されます:

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

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

チュートリアル、プロジェクト、サンプルファイルの場所

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

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

3.2.3 XMLSpy 製品機能と Altova 製品

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

XMLSpy 機能の一覧

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

XMLSpy ヘルプ

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

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

サポートオプション

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

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

Altova 製品

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

4 編集ビュー

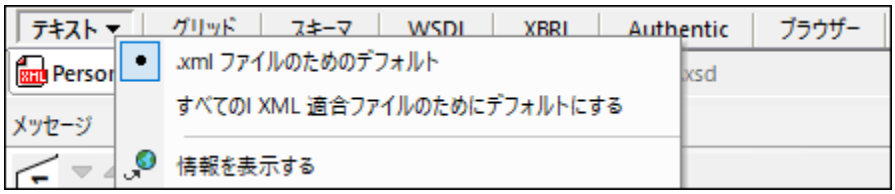
XMLSpy にはパワフルな編集ビューがあります。インテリジェント編集機能を備えたテキストビューに加えて、ドキュメントの編集を飛躍的に簡易化して表示するグラフィカルなビューも備わっています。XMLSpy にてアクティブになっているドキュメントの種類により、メインウィンドウには1つまたは複数の編集ビューが表示されます。例えば、Office Open XML ファイルや ZIP ファイルがアクティブになっていれば、メインウィンドウには1つだけの編集ビュー(アーカイブビュー)が表示されます。HTMLドキュメントがアクティブな場合、テキストビューとブラウザービューの2つの編集ビューが表示されます。XML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

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

- テキストビュー 130
- グリッドビュー 149
- スキーマビュー 159
- W SDL ビュー 231
- XBRL ビュー 243
- Authentic ビュー 255
- ブラウザービュー 256
- アーカイブビュー 257

デフォルトビューの選択

特定の型(例えば、XML ドキュメント、またはJSON ドキュメント)を複数のビューで確認することができます。現在のビューのタブをクリック、または右クリックし表示されるメニューから適切なオプションを選択しデフォルトの編集ビューを直接インターフェイス内で選択することもできます *(下のスクリーンショット参照。)*:



メニューオプションは以下の通りです:

- *ファイル拡張子のためのデフォルト:* 現在の編集ビューが同じファイル拡張子としてアクティブなファイル(例えば、`.xml` ファイル、または `.xq` ファイル)を持つファイルのデフォルトのビューになります。
- *同様に準拠するファイルのためのデフォルト:* 現在の編集ビューが同じ適合型としてアクティブなファイル(例えば、全てのXML-適合 ファイル、または、すべてのXQuery-適合ファイル) を持つファイルのデフォルトのビューになります。これはファイル拡張子のために更に大きいセットであることができます。例えば、2つの異なるファイル拡張子 `.xq` と `.xquery` は XQuery-適合となることができます。
- *情報の表示:* 現在の編集ビューの機能を説明するAltova W eb サイト 上のページへのリンク。

デフォルトビューオプションが選択されると、横にラジオボタンが表示されます *(上のスクリーンショットを参照)*.

以下の場合、デフォルト編集ビューの変更は不可能です。

- *JSON スキーマファイル:* JSON ファイルのようなこれらのファイルは、`.json` 拡張子を持ちます。しかしながら、自動的にJSON スキーマのコンテンツから検知され、常にJSON Schema View 630 で開かれます。ます。

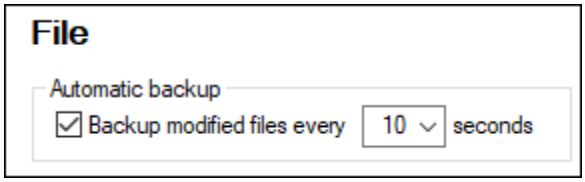
- *Authentic-が有効化された XML ファイル:* これらはXMLドキュメントをAuthentic XMLドキュメントとしてビューすることを可能にするAltova StyleVision スタイルシートを割り当てられた(**.xml** ファイル拡張子を持つ) XML ファイルです。 これらのファイルはAuthentic View[255] で開かれます。
- *XBRL (XBRL タクソノミ)のためのXSDファイル:* これらのファイルには **.xsd** 拡張子が存在します。これらは(XMLスキーマではなく) XBRL タクソノミであるコンテンツから自動的に検知されるため、常にXBRL ビュー[243]で開かれます。

メモ ファイル拡張子のファイルコンポーネント、および各ファイル型に対応するデフォルトの編集ビューのファイルの適合性は、オプションダイアログのファイル型セクション[1458]内で設定することができます。しかしながら、(ここで説明されているとおり)編集ビュー自身内でのデフォルトビューの選択により他の方法よりも早い方法でデフォルトビューを設定することができます。

4.1 ファイルの自動的なバックアップ

XMLSpy 内で変更されたファイルは定期的な頻度で自動的にバックアップされます。(「ツール | オプション | ファイル」[1455])オプションダイアログの *ファイル* タブで以下のスクリーンショットで示されている通り以下を行うことができます:

- 自動バックアップを自動的にオン/オフに切り替える
- バックアップの頻度 (5 秒から 300 秒まで) を指定します。



XMLSpy 内で編集可能なすべてのファイル型はバックアップすることが可能です。ZIP アーカイブはバックアップされません。

インジケーター

メインウィンドウの下のファイルタブにはファイル名の右側に表示されるシンボルが含まれています。これらのシンボルはファイルの保存済み/未保存状態およびバックアップ状態を表示しています *(下のスクリーンショットを参照)*。

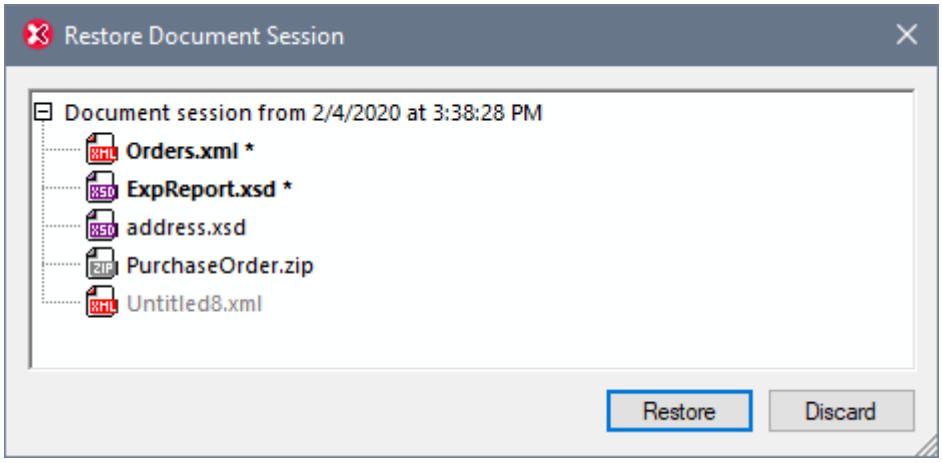
保存済み / 未保存
色付けされたサークルのシンボルはファイルが変更済みの場合表示されます。このようなシンボルが存在しない場合、ファイルは開かれた時点または最後の保存時から変更されていないことを意味します。上のスクリーンショットでは、例えば、`address.xsd` は前回の保存時から変更されておらず、`Untitled8.xml` は作成時より変更または保存されていない新しいファイルであることが示されています。

バックアップの状態
サークルのシンボルはファイルのバックアップの状態を示しています。

- *黄色:* ファイルが変更済みで、最後の変更時からバックアップ (または未保存) の場合。
- *緑:* ファイルがバックアップ済みで、最後のバックアップ時から変更されていない場合。しかしながら、ファイルが保存されていない場合。(保存されている場合、赤いサークルのシンボルは表示されません。)
- *赤:* このファイルのためのバックアップはサポートされない場合 (例えば、ZIP ファイルのアーカイブビュー[257] など) 。またはバックアップに失敗した場合。
- *グレー:* 自動的なバックアップ機能が無効化されている場合 (via the オプションダイアログ[1455] の使用。*上記を参照してください*)。このシンボルの存在はしかしながらファイルが最後の変更時から保存されていないことを意味します。(保存されている場合、赤いサークルのシンボルは表示されません。)

バックアップの復元

XMLSpy が予期せずに終了した場合、アプリケーションが次回起動されるとアプリケーションの終了時に開かれていたすべてのドキュメントのリストを含む「ドキュメントの復元」ダイアログが表示されます *(下のスクリーンショット)*。パスを確認するために各ファイルにマウスをポイントすることができます。保存されていない一時ファイルの場合、ファイルパスはそのファイルのために開かれている名前を付けて保存ダイアログが現在のデフォルトのパスになります。



リスト内の各ファイルのためのフォントスタイルとアスタリスクの存在/不在により以下の情報を提供されます:

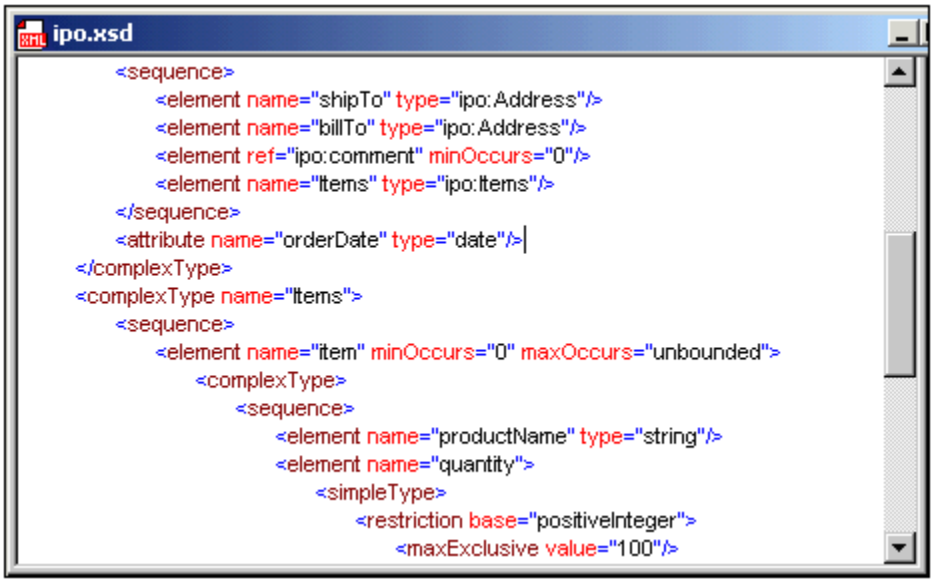
- 太字とアスタリスクはファイルに保存されていない変更が含まれていることを示しています。このようなファイルは最後にバックアップされた状態で復元されます。
- 標準のスタイルはファイルが保存済みで未保存のファイルが存在しないことを示しています。このようなファイルは保存された状態で復元されます。
- 灰色表示されているスタイルはファイルが保存されておらず、またバックアップされていないことを表示しています（例えば、編集されていない新規のファイルなどが挙げられます）。このようなファイルは復元されません。

以下の一つを行うことができます:

- 「**復元**」をクリックして GUI 内のファイル最後のバックアップの状態に復元します。
- 「**破棄**」をクリックしてリストされているファイルを開くことなく、使用可能なバックアップを破棄します。

4.2 テキストビュー

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



このセクションでは、全ての種類のドキュメントで使用することのできる一般的なテキストビューの機能を紹介します。XMLやXQuery、そしてCSS といった特定の種類のドキュメントに対して割り当てられている機能については、それらのドキュメントタイプに関するセクションにて記述されます。例えば、テキストビューにおけるXML 独自の機能についてはXML | テキストビューでXMLを編集する 266セクションで記述されます。

一般的なテキストビューの機能は以下のように分類されています:

- テキストビューでのフォーマット 131 ドキュメントのフォントプロパティ、インデンテーション、そしてワードラップの設定方法を説明します。
- ドキュメント表示 132 行番号、ブックマーク、ノードの展開/折りたたみ、そしてその他の表示に関わる機能についての情報を提供します。
- テキストビューでの編集 135 編集中に使用できる機能、特にインテリジェント編集機能について説明します。
- ドキュメント内の移動 137 テキストビュー内でドキュメントをナビゲートする様々な方法について説明しています。
- 検索および置換 140 テキストビューの検索および置換機能について説明しています。
- 入力ヘルパー 145 コンテキストにより変化するデータ入力オプションを提供するウィンドウです。例えば、指定された位置で入力することのできる要素または属性が入力ヘルパーに表示され、ダブルクリックすることによりオプションを挿入することができます。
- 分割ビュー 146 はテキストビューのメインウィンドウを2つに分割し、あくまでいちなドキュメントを両方のビューで表示します。これにより、長いドキュメントのところなるパートを横に並べて確認することができます。
- テキストビューショートカット 148 はテキストビューコマンドにより一般的に使用されるデフォルトのショートカットをリストしています。

テキストビューに切り替える

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

4.2.1　テキストビューでのフォーマット

テキストビューでは様々なフォーマットオプションが提供されます。これらは以下でリストされています。

フォント

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

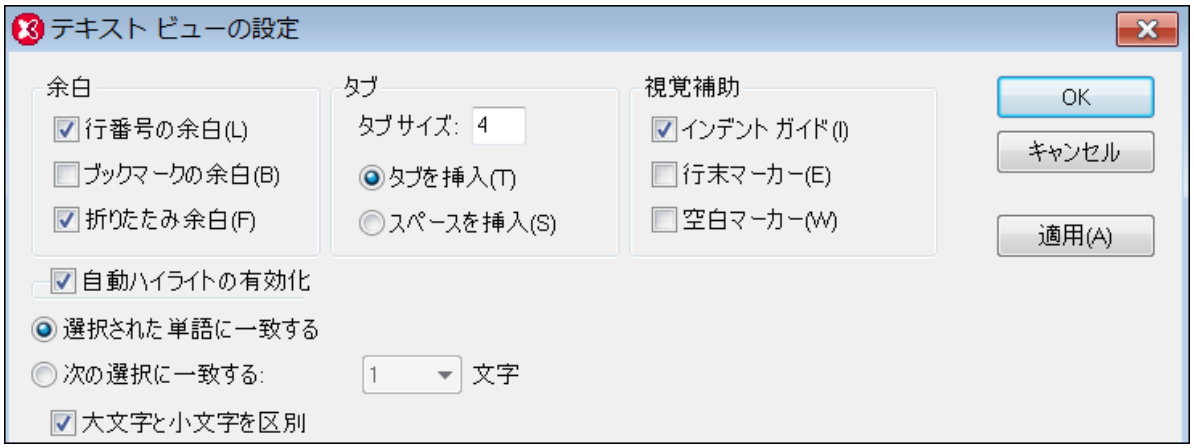
異なる意味を持つドキュメント内のテキストアイテムには別々の色を与えることができます。例えば、要素の名前、属性の名前、そして要素の内容を別々に色付けすることができます。テキストアイテムに対して異なる色付けをした場合、構文のカラー強調表示機能が有効になります。[オプションダイアログのフォントと色のセクション](#)[1470]内でテキストフォントをカスタム化することができます。

インデント

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

ドキュメントをこの方法で表示するには、以下を行う必要があります:

1. オプションダイアログのビューセクション内で、整形出力のためのインデントの使用をチェックします。これにより階層構造を示すためにドキュメントがインデント共に整形出力されます。更に深いレベルは親要素より1つ深いインデントにより表示されます。インデントの使用オプションがチェックされていない場合、ドキュメント内のすべてのラインはインデントがゼロの箇所から開始します。
2. テキストビューの設定ダイアログ(「[ビュー | テキストビュー設定](#)[1363]」*下のスクリーンショットを参照*)にて、「タブを挿入」か「スペースを挿入」を選択します。これによりドキュメントが整形出力される場合インデントにタブまたはスペースが使われるのかが決定されます。スペースが指定されている場合、階層のより深い箇所に、テキストビュー設定のタブサイズ内で指定されている数だけのスペースが挿入されインデントを形成します。



3. 「編集 | 整形出力」[1157] コマンドを選択するか、ツールバーの整形出力アイコンをクリックします。これによりドキュメントテキストがオプションダイアログの整形出力[1462] セクション内で指定されているフォーマットに従い表示されます。整形出力コマンドは不必要な先頭または末尾の空白を削除することにご注意してください。

メモ インデントはガイドは垂直な点線です（*このセクションの始めのスクリーンショットを参照してください*）。これらはテキストビュー設定ダイアログの *資格補助* ペイン内のチェックボックスによりオンとオフに切り替えられます（*上のスクリーンショット参照*）。

フォーマットにタブとスペースを使用する

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

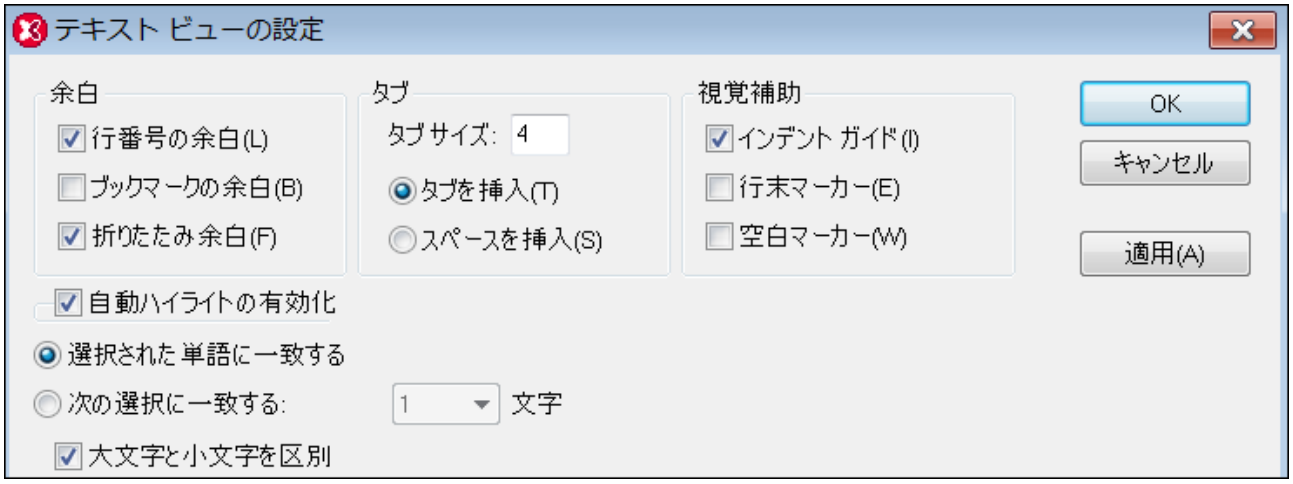
ワードラップ

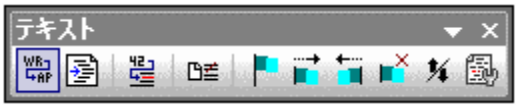
「表示 | ワードラップ」[1362] コマンドを有効にすることで、メインウィンドウの幅より先長い行は折り返して表示されます。対応するアイコンはテキストツールバー[132] にもあります。

4.2.2 ドキュメントの表示

テキストビューでは大規模なテキストの閲覧と編集を簡単にする視覚機能を使用することができます。とても便利な機能として: (i) 行番号[133]、(ii) ブックマーク[133]、(iii) ソースの折りたたみ[133]（ノードごとに展開し折りたたんだりする機能）、(iv) インデントガイド[134]、そして (v) 行末マーカーと空白マーカー[134] があります。これらのコマンドはテキストビュー設定ダイアログ（1番目のスクリーンショット）とテキストツールバー（2番目のスクリーンショットを参照）にて使用することができます。

テキストビュー設定ダイアログ[1363] には「**表示 | テキストビューの設定**」コマンドまたは、テキストツールバーの「**テキストビューの設定**」ボタンからアクセスすることができます。テキストビュー設定ダイアログの設定内容はアクティブなドキュメントだけでなく、アプリケーション全体に対して適用されます。





その他の便利な機能として「ズーム」[135]と「ナビゲートと検索」[137]機能があります。

行番号

行番号は行番号余白に表示され(下のスクリーンショットを参照)、テキストビュー設定ダイアログで有効/無効にすることができます(上のスクリーンショットを参照)。テキストのセクションが折りたたまれているときには、折りたたまれている行の番号も同様に隠されます。関連するコマンドに「行/文字へ移動」[137]コマンドがあります。

ブックマーク

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

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

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

ソースの折りたたみ

テキストの折りたたみ(またはソースのおりたたみ)はXQuery 中かっこ、XQuery コメント、XML 要素、XML コメントにより有効化され、これらのノードを展開または折りたたむ機能を意味します。このようなノードは、+/- サインによりソース折りたたみ余白内で表示されます(下のスクリーンショット参照)。余白はテキストビュー設定ダイアログ[1363]により切り替えのオンとオフを行うことができます(上のスクリーンショット参照)。下のスクリーンショット内では、3つのノードが折りたたまれています: `shipTo` 要素2つの `item` 要素。ノードが折りたたまれると、視覚的に省略記号により表示されます(下のスクリーンショットでは緑の枠でかこまれています)。内容がポップアップでの表示に大きすぎる場合、ポップアップのしたの部分に省略記号で表示されます。

テキストツールバーの「全て展開/折りたたみ」アイコン 全てのノードを展開、またはトップレベルドキュメント要素に折りたたむことができます。
ノードの切り替えアイコン(＋/－アイコン)をクリックする際に以下のオプションが利用可能になります:

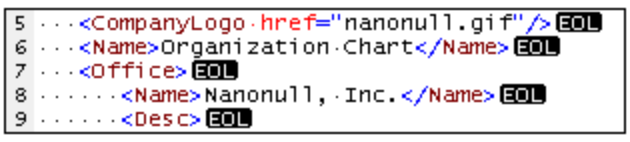
クリック [-]	ノードの折りたたみを行ないます。
クリック [+]	クリックすることでノードが展開され、そのノードの子孫は、折りたたみが行われる前の状態で表示されます。
Shift+クリック [-]	全ての子孫ノードを子ノードのレベルまで折りたたみます。クリックされたノードそのものは開かれ、折りたたまれた全ての子ノードが表示されます。この操作は、枝の末端にあるノードを除き、全てのノードに対して適用することができます。
Ctrl+クリック [+]	ノードまたはその子孫が折りたたまれている場合、「Ctrl」+ クリックにより、折りたたまれている全ての子孫ノードを展開することができます。つまり、こ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

インデントガイド

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

行末マーカー、空白マーカー

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



ズームインとアウト

「Ctrl」キーを押しながらスクロール(マウスのホイール)を使用することでテキストビューの拡大と縮小を行うことができます。この機能を使うことで、テキストビューのテキストサイズを調整することができます。フォントサイズを変更したい場合は、オプションダイアログ[131]にて行うことができます。

4.2.3 テキストビューでの編集

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

- 構文の色分け[135]
- インテリジェント編集[135]
- 自動補完[136]
- 開始タグと終了タグのマッチ[136]
- 同じ階層で要素を移動させる[137]
- 要素全体を選択し親要素に移動する[137]
- ドラッグアンドドロップとコンテキストメニュー[137]
- 制限無しに元の状態に戻す[137]
- スペルチェック[137]

(XML[261] やXQuery[442] などの)ドキュメントの種類によっては、更に独自の機能が利用可能です。これらの機能については、それぞれ対応するドキュメントの種類に関するセクションで記述されています。

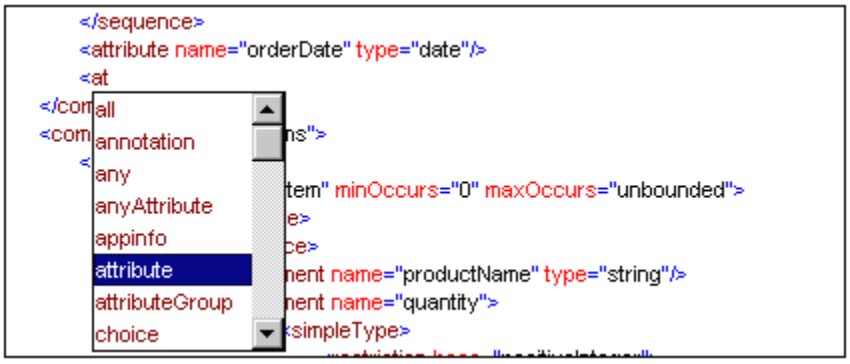
メモ 大きなファイルに関しては、速いロードと編集のため自動補完と入力ヘルパーが無効化されます。ファイルのサイズの敷居はユーザーにより指定することができます。更に詳しい情報に関しては、「オプション | 編集」[1461]を参照してください。

構文の色分け

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

インテリジェントな編集

スキーマをベースにしたXMLドキュメントを扱っている場合、XMLSpy のテキストビューで様々なインテリジェント編集機能を使用することができます。これらの機能を使用することで、現在編集しているコンテンツモデルに従った正しい要素、属性、または属性の値を素早く入力することができます。例えば、要素の開始タグの入力を開始すると、この時点でスキーマにより許可されている全ての要素の名前がポップアップで表示されます(下のスクリーンショットを参照)。要素名を選択し、「Enter」を押すことで開始タグにその要素名が入力されます。また、開始タグが作成された後、終了タグが自動的に追加されます(*下の自動補完[136] below を参照してください*)。



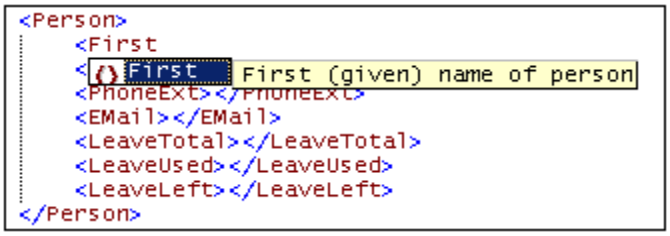
ポップアップウィンドウは以下の場合にも表示されます:

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

自動補完

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

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



自動補完は[オプションダイアログの編集タブ](#)1461(「**ツール | オプション | 編集**」)にて有効/無効化することができます。

開始と終了タグの一致

カーソルがマークアップ要素の開始もしくは終了タグの中をポイントしているとき、「Ctrl」+ E を押すことで選択を他のペアのメンバーに移動することができます。「Ctrl」+E を繰り返し押すことで、開始タグと終了タグを切り替えることができます。この機能を使って、簡単にXML 要素の開始タブと終了タブを見つけることができます。これは、XML 要素のタグを開始と終了を検索するために役にたちます。更に、開始タグで、入力を始めると、開始と終了タグの名前が一致するか否かにより[要素の名前は異なる色でハイライトされます](#)1470。これは視覚的な編

集エイドとしての役割を果たします。色のハイライトは、オプションダイアログ[1470]で設定することができます。開始タグ内で要素の名前を編集すると、自動的に終了タグが編集されます。

互いに相対する要素を移動する

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

要素全体を選択し、親要素に移動する

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

ドラッグアンドドロップとコンテキストメニュー

ドラッグアンドドロップやテキストのブロックを他の場所に移動し、右クリックにより表示されるコンテキストメニューを使用して、(切り取り[1149] やコピー[1149]、ペースト[1149]、削除[1149]、メールで送信[1143]、そして行/文字に移動[1362]などの) よく使われるコマンドにアクセスすることでもテキストの移動を行うことができます。

制限無しに元の状態に戻す

XMLSpy は全ての編集オペレーションに対して制限無しに元に戻す[1149] および やり直し[1149]を提供します。

スペルチェック

テキストビューでは、内蔵の言語辞書を使用することでドキュメントのスペルチェックを行うことができます。ユーザー辞書を作成することで、言語処理に含まれていない単語を加えることができます。詳細については、スペリング[1414]ならびにスペルチェッカーのオプション[1418]コマンドを参照ください。

4.2.4 ドキュメント内の移動

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

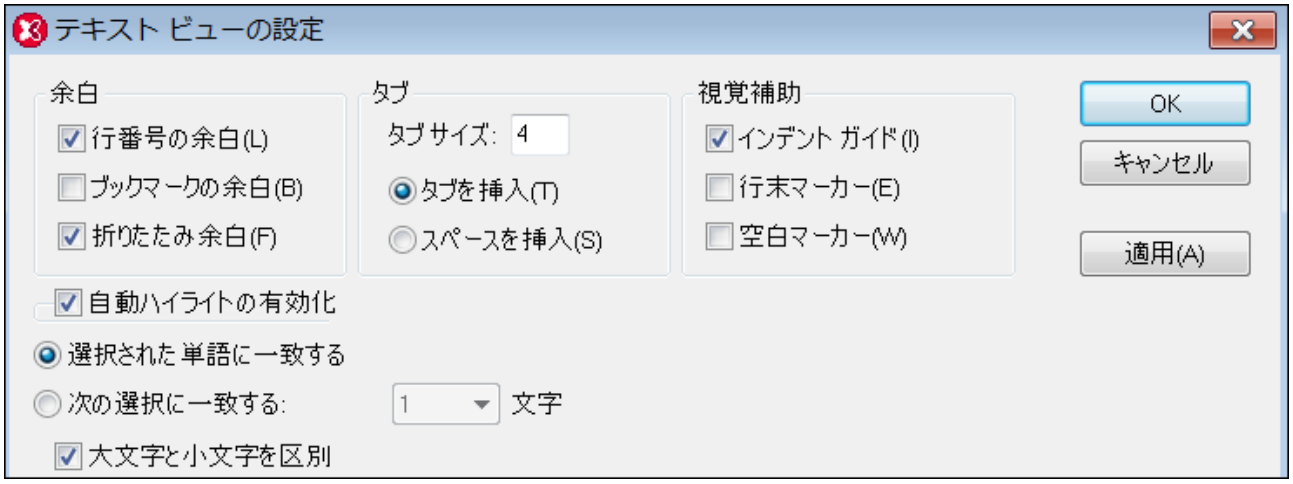
- テキストのハイライト[137] により、選択したテキスト文字列または文字に一致する全てのマッチを検索することが可能になります。それぞれの一致は、スクロールバーに示されており、全ての一致に簡単に移動することができます。
- スクロールバー内のドキュメントの概要[139] により、ドキュメント内でのカーソルとテキストの相対的な位置を表示します。
- ライン/文字に移動[139] により、指定したラインと文字に直接移動することができます。

テキストのハイライト

テキストビュー設定ダイアログ(「表示 | テキストビュー設定」[1363]) 内でテキストのハイライトが有効化されていると、選択されたテキストのドキュメント内の全ての一致がハイライトされます。選択は、薄い青色でハイライトされ、一致する箇所は薄いオレンジ色でハイライトされます (*下のスクリーンショット参照*)。選択と、その一致する箇所は、スクロールバー上で四角の灰色のマーカーで表示されます。スクロールバー内で現在のカーソルの位置は、青いカーソルマーカーで示されています。

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

テキストのハイライトをオンに切り替えるには、テキストビュー設定ダイアログ（「表示 | テキストビューの設定」1363 内の*自動ハイライト*の有効化を選択します。*下のスクリーンショット*）。選択は、文字全体、または、文字の固定された数値で定義することができます。また、キャストを考慮するかを選択することもできます。



以下の点に留意してください：

- 文字の部分では、選択内で一致する必要がある文字の最小数を指定することができます。例えば、2つ以上の文字が一致するかを選択することができます。この場合、一文字の選択は、一致とはみなされず、2つ以上の文字から構成される選択のみがマッチ

されます。ですから、この場合、`t` を選択すると、一致する項目は表示されません。`ty` を選択する場合、`ty` に一致するマッチが表示されます。`typ` を選択すると、`typ` に一致するすべてのマッチが表示されます。
- 文字の検索の場合、文字を区別するために次の条件が考慮されます: (角かっこのない) 要素名、要素タグの角かっこ、属性名、引用符なしの属性の値。

メモ　要素開始タグと要素完了タグ[135] は、自動ハイライトの有効化に影響されない個別の機能です。

スクロールバー内のドキュメントの概要

スクロールバーには以下の機能が搭載されています:

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

以下の点に留意してください:

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

ライン/文字に移動

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

4.2.5 検索および置換

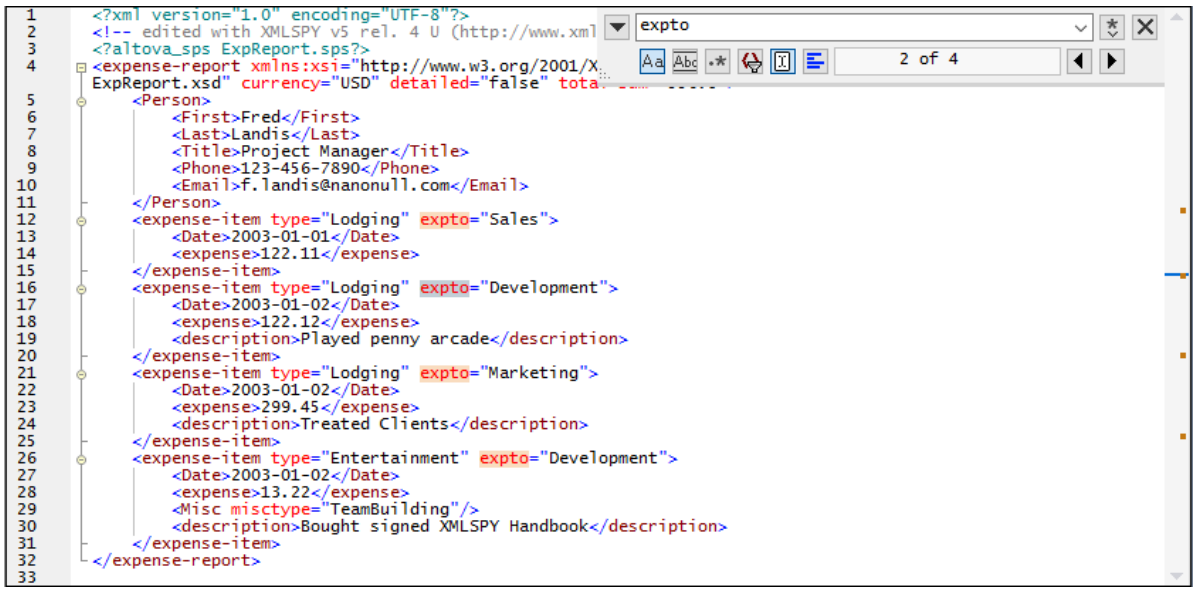
このセクション:

- ドキュメント内のテキストの検索 140
- 検索オプション 141
- テキストの置換え 141
- 正規表現の使用 142
- 正規表現のサンプル 143

ドキュメント内でのテキスト検索

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

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



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

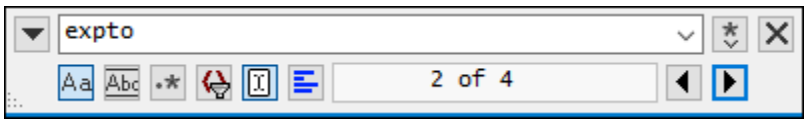
以下の点に留意してください:

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

- ダイアログボックスを開く前にテキストが選択されている場合、選択されたテキストは、自動的に検索フィールドとして既知の検索用語フィールドに挿入されます。
- 範囲内で検索する場合、以下を行います: (i) 選択範囲をマークします。(ii) *選択範囲内での検索*[141] オプションをオンにして、選択をロックします。(iii) 検索用語を入力します。他の選択を検索するには、現在の選択を、*選択範囲内での検索*[141] オプションをオフにして、ロックを解除してください。新しい選択を決定し、*選択範囲内での検索*[141] オプションをオンにします。
- 検索ダイアログが閉じられた後、現在の検索を「 **F3** 」を押して繰り返すことができます。または、「**Shift+F3**」を押して後ろ向きに検索することができます。検索ダイアログが表示されます。

検索オプション

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



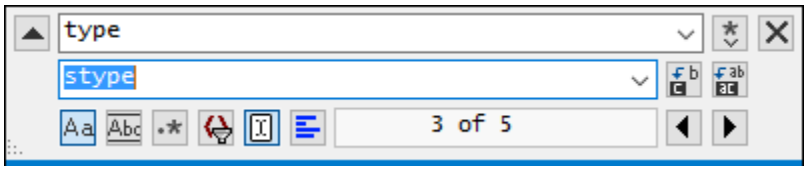
以下のオプションから選択することができます:

- *大文字と小文字を区別する*: オンに切り替える場合は大文字と小文字が区別されます (`Address` は、`address` とは異なります)。
- *単語単位*: テキスト内の単語単位が一致するもののみを表示する。例えば、単語単位がオンに切り替えられており、入力文字列が `fit` の場合、`fit` のみが検索文字列に一致します。例えば、`fitness` 内の `fit` は、一致ではありません。
- *正規表現*: オンに切り替えられていると、検索用語は正規表現として読み込まれます。どのように正規表現が使用されるかに関する説明は、下の *正規表現*[142] を参照してください。
- *フィルターの結果*: 検索が実行されるいつまたは1つ以上のドキュメントコンポーネントを選択します。
  - *アンカーの検索*: 検索用語が入力されると、ドキュメント内の一致がハイライトされ、一致の内の1つが現在の選択としてマークされます。アンカーの検索は最初の選択がカーソルに位置に相対するかを決定します。アンカーの検索がオンになっていると、最初の現在選択されている一致は、カーソルの位置の次の一致を指します。アンカーの検索がオフになっている場合、最初の現在選択されている一致は、ドキュメントの始めから数えた一番目の一致です。
- *選択範囲内での検索*: オンに切り替えられると、現在の選択範囲がロックされ、検索が選択範囲内に制限されます。それ以外の場合、ドキュメント全体が検索されます。テキストの新しい範囲を選択する前に、現在選択されている検索範囲を検索範囲内オプションをオフにしてロックを解除してください。

テキストの置換え

テキスト文字列を置き換えるには、以下を行います:

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



2. 検索フィールド内で置き換えられる文字列を入力し、フィールドの置換内に新規の文字列を入力します。置き換えられる一致するテキストの数と、現在選択されているマッチは、フィールドの置換えの下に表示されます。また、一致の場所は巣クロークバー上でベージュのマーカーで表示されています。また、一致する箇所はスクロールバー内でベージュのマーカーで表示されています。(詳細に関しては、上の *ドキュメント内でのテキストの検索*[140] を参照してください)。例えば、上のスクリーンショットは、文字列 `type` に対して5つの一致が存在することを示しており、3番目の一致が現在選択されています。

3. 「次を置き換える」と「全てを置き換える」ボタンは置換フィールドの右側にあります。「次を置き換える」をクリックすると、以下の1つが起こります: (i) カーソルが一致、または、一致内に存在する場合、置換えが実行されます。(ii) カーソルが一致の外に存在する場合、次の一致に移動します。「次を置き換える」をクリックして、この一致を置き換え、「全てを置き換える」をクリックして全ての一致を置き換えます。

以下の点に留意してください:

- ドキュメント全体ではなく、選択範囲内でテキストを置き換えるには、以下を行います: (i) 選択をマークする。(ii) *選択範囲内で検索*[141] オプションをオンに切り替えて、選択をロックする。(iii) 検索と置き換えテキスト文字列を入力します。(iii) 「次を置き換える」または「全てを置き換える」を必要に応じてクリックします。他の選択範囲内のテキストを置き換えるには、現在の選択範囲を*選択範囲内で検索*[141]オプションをオフに切り替えてロック解除し、新しい選択を指定し、*選択範囲内で検索*[141] オプションをオンにして新しい選択をロックします。
- 置換えのアクションを元に戻す場合は、「**Ctrl+Z**」または、「**編集 | 元に戻す**」を選択します。

正規表現の使用

正規表現 (regex) を使用して、テキスト文字列を検索することができます。これを行うには、*正規表現*のオプションをオンに切り替えます (上の*検索オプション*[141] を参照してください)。これにより、検索用語フィールド内のテキストが正規表現として評価されます。正規表現の構築へのヘルプは、検索用語フィールドの右にある「**正規表現 ビルダー**」ボタンをクリックしてください (*下のスクリーンショット参照*)。ビルダー内のアイテムをクリックして、正規表現メタ文字を対応する検索用語フィールドに入力します。下のスクリーンショットは、電子メールアドレスを検索するための簡単な正規表現を表示しています。メタ文字に関する簡単な説明は、下の*正規表現 メタ文字*[142] セクションを参照してください。

正規表現メタ文字

正規表現 メタ文字のリストは以下の通りです。

.	任意の文字を一致する。これは単一の文字のプレースホルダーです。
\(	一致をタグした範囲の開始をマークする。
\)	一致をタグした範囲の終了をマークする。
\(abc\)	\(と \) メタ文字は、タグされた式の開始と終了をマークします。一致する箇所を、後で参照する (バックレファレンス) 目的のためにタグ (「記録」) する場合、役に立つ可能性があります。9つまでの式をタグ付けすることが

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

特別文字の表記
以下の式に注意してください。

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

正規表現のサンプル

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

サンプル1: 要素の名前変更

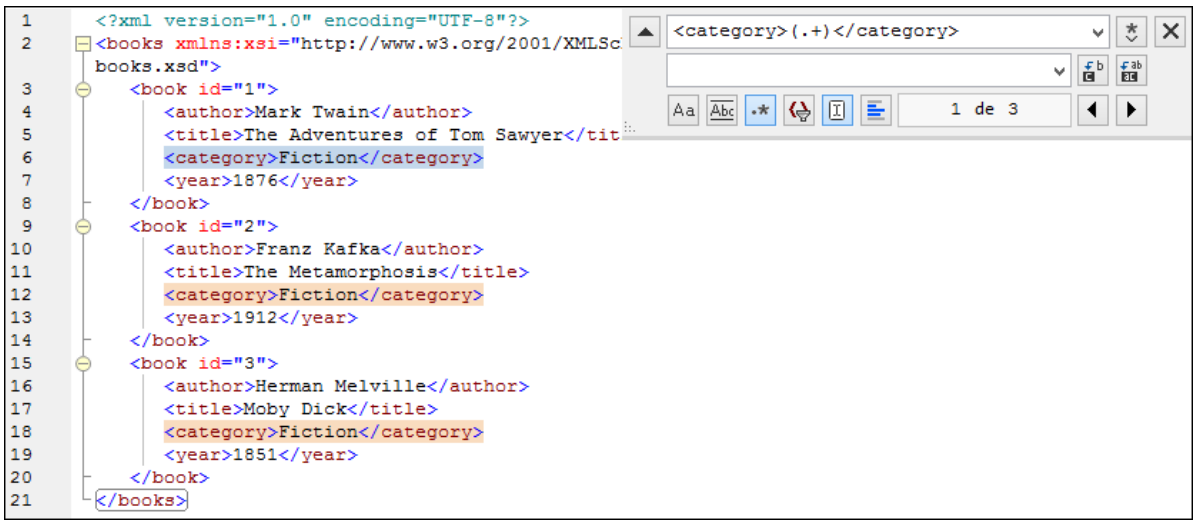
サンプルXMLコードリストには、ブックのリストが含まれています。このサンプルの目的を、の各ブックの `<Category>` 要素を `<Genre>` と置き換えることとします。この目的を達成するために正規表現を使用します。

```
<?xml version="1.0" encoding="UTF-8"?>
<books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:noNamespaceSchemaLocation="books.xsd">
   <book id="1">
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <category>Fiction</category>
      <year>1876</year>
   </book>
   <book id="2">
      <author>Franz Kafka</author>
      <title>The Metamorphosis</title>
      <category>Fiction</category>
      <year>1912</year>
   </book>
   <book id="3">
      <author>Herman Melville</author>
      <title>Moby Dick</title>
      <category>Fiction</category>
      <year>1851</year>
   </book>
</books>
```

必要条件を満たすには、以下のステップを踏んでください：

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



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

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

名前を変更されました。

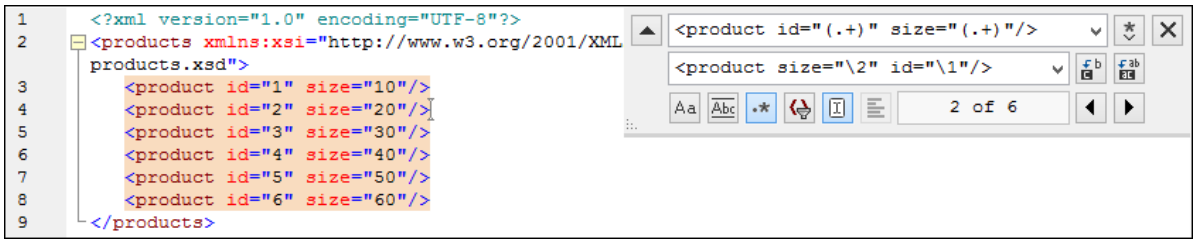
サンプル2: 属性の順序を変更する

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

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

必要条件を満たすには、以下のステップを踏んでください:

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



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

4.2.6 テキストビューでの入力ヘルパー

テキストビューで利用可能な入力ヘルパーの内容は編集中のドキュメントの種類により変化します。標準的なドキュメントの種類に対する入力ヘルパーは以下に示されています。入力ヘルパーの一般的な使用方法については以下に記述されて146います。特定のドキュメントの種類に対応したその他の機能については、それぞれのドキュメントの種類のセクションでそれぞれ記述されます。

- *XML*: 要素(下のスクリーンショット)、属性、エンティティ

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

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

入力ヘルパーアイテムの表示と使用方法

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

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

メモ 大きなファイルでは、自動補完と入力ヘルパーを無効化し、より速いロードと編集を行うことができます。 しきいファイルのサイズはユーザーにより指定することができます。詳細は[オプション| 編集](1461)のセクションを参照してください。

4.2.7 分割ビュー

分割ビューは、テキストビューのメインウィンドウを2 つに分割し、両方のビュー内のアクティブなドキュメントを表示します。ドキュメントを横に並べ、異なる部分を確認することのできるようにする各ビューを個別にナビゲートすることができます。変更の編集は、基のドキュメントに追加され、両方のビュー内で反映されます。

分割ビューと単一ビュー間の切り替え

アクティブなドキュメント の分割ビューを以下のように作成します：

- *水平分割:* 右上にある水平分割からドラッグします（下のスクリーンショットでは緑色で丸囲みされています）。
- *垂直分割:* 左下の垂直分割を右側にドラッグします（下のスクリーンショットでは青色で丸囲みされています）。

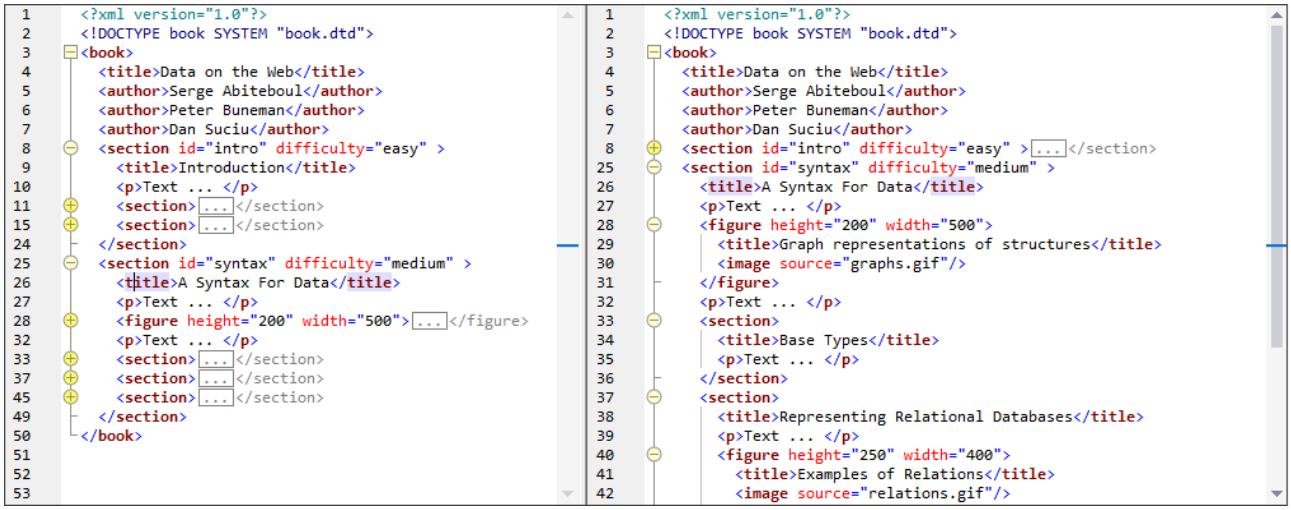
分割ビューから単一ビューに戻るには、以下の1つを行います:

- 分割バーをダブルクリックします。または、
- メインウィンドウの1つの分割バーを平行なエッジに移動します。

メモ 分割ビューは各ドキュメントのために個別に作成されます。

分割ビュー内でのナビゲートを編集

分割ビュー内での作業の利点は、両方のビュー内でドキュメントの編集が可能で、かつ、長いドキュメントの異なるパーツを横に並べて表示できることです。下のスクリーンショットは垂直の分割ビュー内のドキュメントを表示しています。



以下の点に注意してください：

- 分割ビューの両方のビュー内で使用することのできるすべての機能の表示 132 (ラインの番号づけとソースの折りたたみなど)、編集機能 135、移動機能 137 など (上のスクリーンショットを参照)。
- 分割ビューの各ビュー内で個別にスクロールし、ナビゲートすることができます。
- 各ビュー内で個別にソースの折りたたみを使用することができます。
- 入力ヘルパーアクションを含む、ドキュメント上のすべての編集アクションは両方のビュー内で反映されています。

4.2.8 テキストビューショートカット

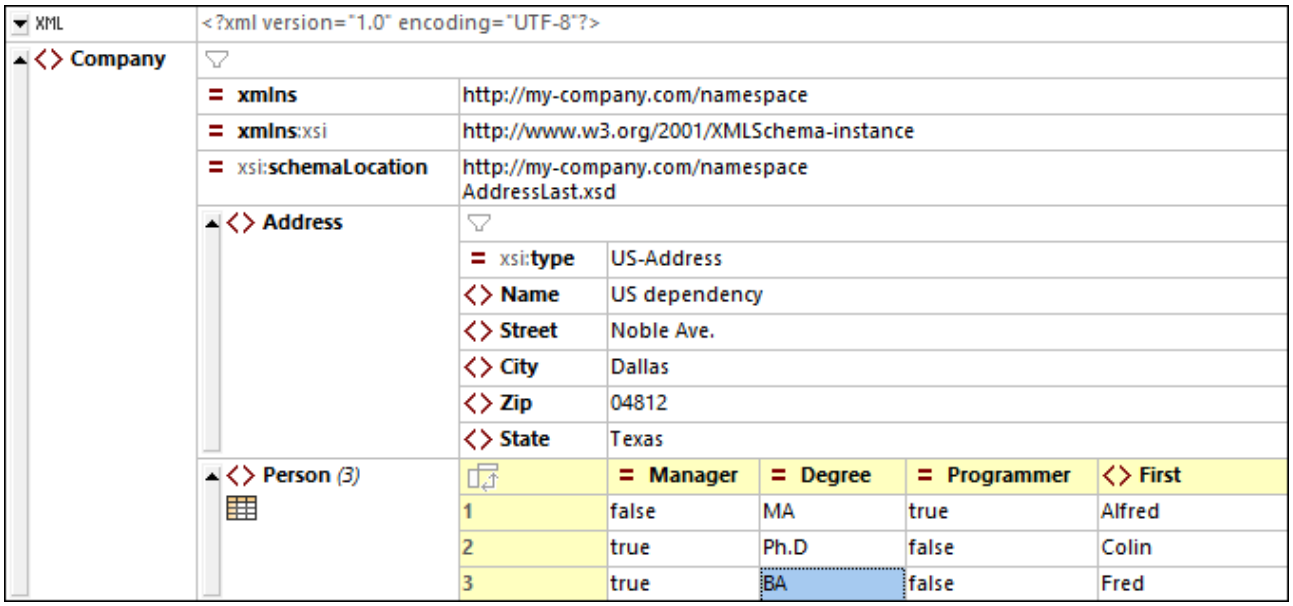
テキストビューコマンドで頻繁に使用されるデフォルトのショートカットは、以下にリストされているとおりです。カスタマイズダイアログのキーボードタブ 1443 内でデフォルトのショートカットを変更することができます。

テキストビューコマンド

CTRL + E	開始/終了タグ間を移動する
CTRL + Shift + E	カーソルを含む要素を選択する
CTRL + Alt + E	親要素に移動する
CTRL + "+"	ズームイン
CTRL + "-"	ズームアウト
CTRL + 0	ズームのリセット
CTRL + マウスホイールを前にロールする	ズームイン
CTRL + マウスホイールを後ろにロールする	ズームアウト

4.3 グリッドビュー

XMLドキュメント、JSONドキュメントおよびDTD のためにグリッドビューを使用することができます。(*下のスクリーンショットはXMLドキュメントのグリッドビューです。*)グリッドビューはドキュメント の階層構造をネストされたコンテナーを介して表示しています。ドキュメントの構造の明確なピクチャを簡単に展開または折りたたむことができます。この結果、グリッドビューではコンテンツと構造を簡単に操作することができます。



グリッドビュー内での作業方法に関しては、以下のセクションを参照してください:

- グリッドビュー内のXML 268
- JSON グリッドビュー 600
- DTD 384

グリッドビューのカスタム化

- 列のサイズを調整するには、カーソルを適切な枠線にポイントして希望する幅を達成するためにドラッグします。
- 最も大きいエントリの幅に列をサイズ調整するには、その列の右のグリッドラインをダブルクリックします。
- 全てのコンテンツを表示するために列の幅を調整するにはメニューアイテム、表示 | 最適な幅 1362 コマンドを選択、または、グリッドビューツールバー内の最適な幅アイコンをクリックします。
- セルの高さはコンテンツにより決定されます。メニューオプション[ツール | オプション | ビュー | 拡張されたグリッドビュー、[セルの高さを xx 行に制限する]により調節することができます。

メモ　グリッドビュー内のデータをマークし、テキストビューに切り替える場合、そのデータはテキストビュー内でもマークされます。

4.3.1 グリッドビューでの編集

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

要素屋属性名の挿入または追加

選択されたアイテムに関連する要素や属性を挿入するには、XML メニューのコマンドを使用するか、**属性と要素ツールバー** にあるアイコンを使用します。このツールバーが表示されていない場合、メニューオプションの「**ツール | カスタマイズ**」を選択して、**ツールバー**タブから**属性と要素**にチェックを入れてください。ツールバーにあるこれらアイコンコマンドとXML メニューコマンドについては、XML メニュー[1198]の項を参照ください。
要素または属性を挿入または追加するには:

1. 挿入または追加する要素または属性に関連するアイテムを選択します。
2. 属性と要素ツールバー内の適切なアイコンをクリックするか、XML メニュー[1198]から適切なコマンド(**挿入**、**追加**、または**子を追加**)を選択します。
3 目的の要素または属性をポップアップから選択し、クリックするか「Enter」を押下します。それ以外にも、目的の要素または属性の名前を入力することができます。

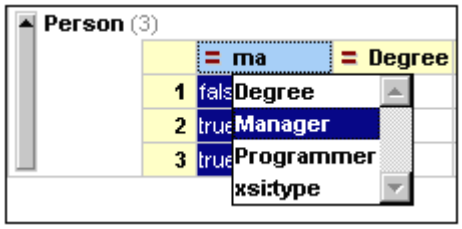
グリッドビューで選択されているものによって、入力ヘルパー[155]の内容が常に変わり続けるのに気がついたかと思います。情報ウィンドウ[1494]に表示される情報も、選択されている要素や属性に従って常に変わり続けます。

グリッドビューにおける改行とタブ

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

要素または属性の名前を編集する

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



要素名または属性名を編集するには:

1. 要素または属性名をダブルクリックします。利用可能なオプションがポップアップで表示されます。
2. ポップアップメニューから必要な要素または属性を選択します。
3. 「Return」を押す、または、名前をクリックして選択を確認します(**Esc** により変更が破棄されます)。

名前空間プレフィックスを追加する

「XML | 名前空間プレフィックス[1233]」コマンドによりグリッドビューの選択されたノードとその子孫(要素と属性)に名前空間プレフィックスを追加することができます。名前空間はこれとは別に宣言されなければなりません。

グリッドビューコンテキストメニュー

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

スペルチェック

グリッドビューでは、内蔵の言語辞書を使用することでドキュメントのスペルチェックを行うことができます。ユーザー辞書を作成することで、言語処理に含まれていない単語を加えることができます。詳細については、スペリング[1414] および スペルチェッカーのオプション[1418] コマンドを参照ください。

XML 検証

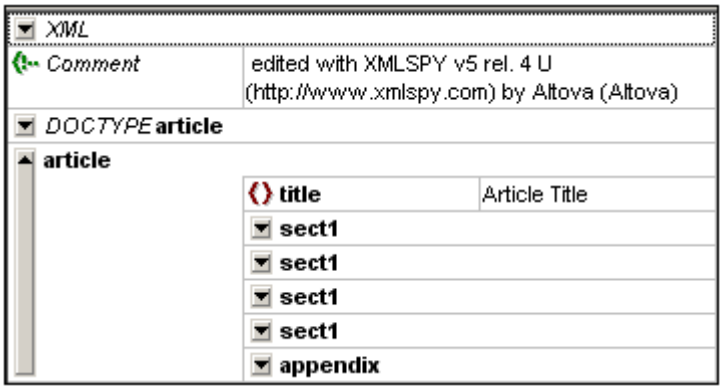
グリッドビューではユーザーの入力を部分的に検証することができます。例えば、属性値における出現とパターンのエラーを検知することが可能です。しかしテキストだけを含めることができる要素に子要素を追加しても、エラーとは検知されません。XML ドキュメントが妥当であることを保証するには、テキストビューへ切り替えた後に検証処理を行なってください。

4.3.2　グリッドビューテーブル

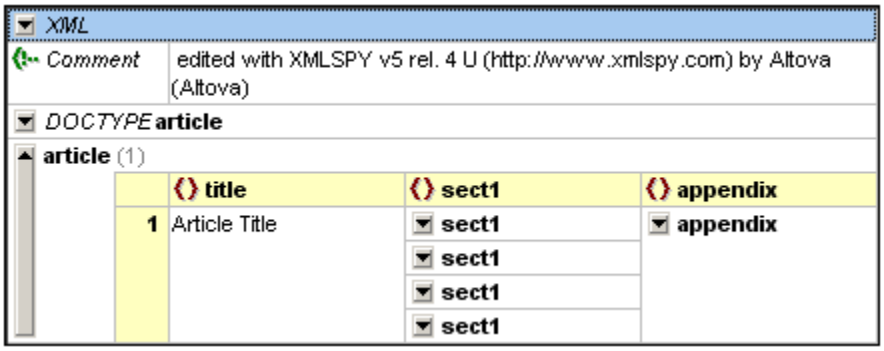
テーブルビューは再帰的な要素をテーブルの形で表示するグリッドビューに統合されたビューです。テーブルビューは通常のグリッドビューとは違い、テーブルとして表示されている要素の子となる型をそれぞれ列として表現します。それから全ての列や選択に対してのプロパティなどを設定することができます。例えば、以下のようなXMLドキュメントを考えてみましょう:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!-- edited with XMLSPY v5 rel. 4 U (ht
3  <!DOCTYPE article SYSTEM "C:\Program Fi
4  <article>
5    <title>Article Title</title>
6    <sect1>
7      <title>Section 1</title>
8      <simpara/>
9    </sect1>
10   <sect1>
11     <title>Section 2</title>
12     <para/>
13   </sect1>
14   <sect1>
15     <title>Section 3</title>
16     <para/>
17     <para/>
18   </sect1>
19   <sect1>
20     <title>Section 4</title>
21     <para/>
22   </sect1>
23   <appendix>
24     <title/>
25     <para/>
26   </appendix>
27 </article>
```

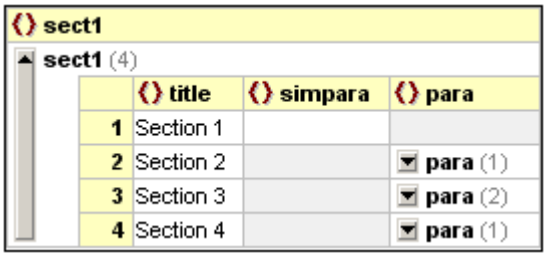
ドキュメントの要素は**article** で、article は以下の子要素を含んでいます: title 要素 1 つ、**sect1** 要素 4 つ、**appendix** 要素を 1 つ。**sect1** と**appendix** 要素はそれぞれ 1 つのtitle 要素を含んでおり、任意の数の**para** か**simpara** 要素がそれに続きます。このドキュメントは通常のグリッドビューでは以下のように表示されます:



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



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



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

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

テーブルデータを操作する

以下のようにしてテーブルのデータを操作することができます:

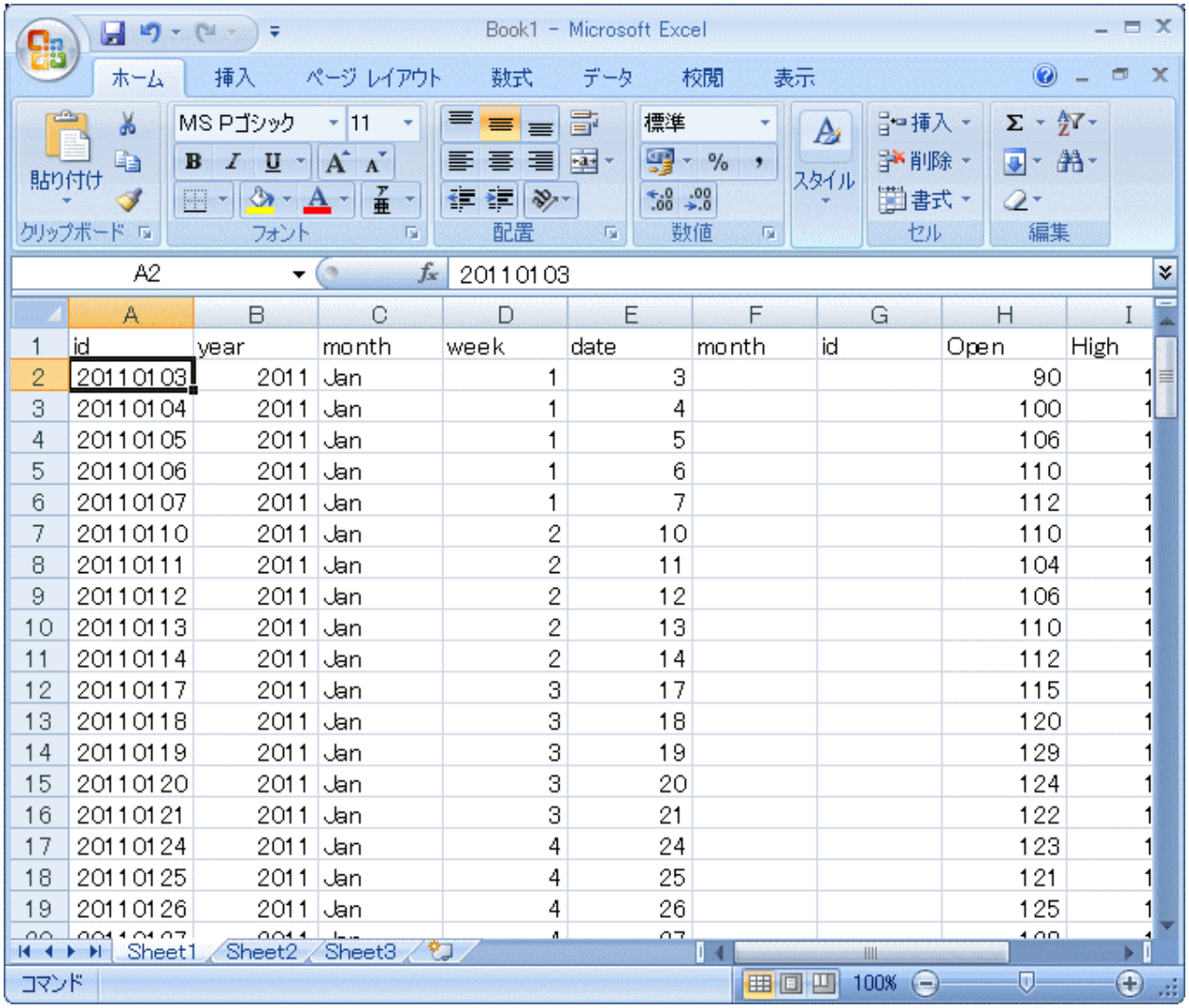
- 列のヘッダをドラッグアンドドロップして列を移動する。
- メニューコマンド「XMI | テーブル | 昇順に並び替え」を使用してテキストノードの列のデータをソートする(降順でも可能)。
- メニューコマンド「XML | テーブル | 行を挿入」(または**行を追加**)で行の挿入(や追加)を行う。

テーブルビューと外部アプリケーションでデータをやり取りする

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

expense-item (3)		type	expto	Date	expense	description
	1	Lodging	Sales	2003-01-01	122.11	
	2	Lodging	Development	2003-01-02	122.12	Played penny arcade
	3	Lodging	Marketing	2003-01-02	299.45	Treated Clients

上のスクリーンショットはテーブルビュー内でテーブルとして表示されている6個の `item` 要素を表示しています。これらの2つの行をExcel シートにコピーするには、構成されたテキストとしてコピーし、Excel シートに張り付けます。結果は以下に示される通りです。詳細に関しては、構造化されたテキストとしてコピーするを参照してください。



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

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

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

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

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

更に複雑なデータのやり取りを行う方法として、XMLSpy にはテキストファイル、W ord ドキュメント、または、データベースを直接 XML データにインポート/エクスポート[1331] する機能が搭載されています。

4.3.3 グリッドビューでの入力ヘルパー

要素入力ヘルパー

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

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

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

属性入力ヘルパー

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

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

注意: 既存の属性で複数の出現が許されていないものはグレーで表示されます。

エンティティ入力ヘルパー

エンティティ入力ヘルパーには内部または外部のDTD の宣言されたエンティティを全て表示します。グリッドビューにてテキストノードまたは属性ノードが選択されている場合、そのようなノードは子を持つことができないので、「子を追加」タブは空で表示されます。

グリッドビューにてカーソルを使ったエンティティの挿入を行うには、テキストフィールドの必要な場所にカーソルを移動するか、必要なフィールドを選択した後、適切なタブを選択し、エンティティをダブルクリックします。以下の点に注意してください:

- カーソルが(属性フィールドを含む)テキストフィールド内に位置しているとき、カーソルの挿入ポイントにエンティティは挿入されます。
- (#PCDATA やsimpleType のような)テキストしか表示できない子が選択されていながらカーソルはテキストフィールド内にあるとき、既存のテキストコンテンツがエンティティにより置き換えられます。
- テキストではないフィールドが選択されている場合、選択された入力ヘルパーのタブに対応した位置にテキストとしてエンティティが作成されます。

注意: I内部エンティティを追加した場合、その上方をエンティティ入力ヘルパーに表示するためには、ドキュメントを一度保存して再度開く必要があります。

4.3.4 グリッドビューショートカット

グリッドビューコマンドで頻繁に使用されるデフォルトのショートカットは、以下にリストされているとおりです。[カスタマイズダイアログのキーボードタブ](1443)内でデフォルトのショートカットを変更することができます。。

関数キーのショートカット(検証と変換のためのショートカットも含む)機能キーのショートカット

F1	ヘルプメニュー
F1 + Alt	最後に開かれたファイルを開く
F3	次を検索
F4 + CTRL	アクティブなウィンドウを閉じる
F4 + Alt	XMLSpyを閉じる
F5	更新
F6 + CTRL	開かれているウィンドウを循環する
F7	整形式チェック
F8	検証
F10	XSL 変換
F10 + CTRL	XSL:FO 変換

ファイルとアプリケーションコマンド

Alt + F1	最後に開かれたファイルを開く
CTRL + O	ファイルを開く
CTRL + N	新規ファイル
CTRL + P	ファイルの印刷
CTRL + S	ファイルの保存
CTRL + F4	アクティブなウィンドウを閉じる
CTRL + F6	開かれているウィンドウを循環する

CTRL + TAB	開かれているドキュメントを切り替える
Alt + F4	XMLSpyを閉じる

その他のキー

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

編集コマンド

CTRL + A	すべて選択
CTRL + F	検索
CTRL + G	ライン/文字に移動する
CTRL + H	置き換える
CTRL + V	貼り付け
CTRL + X	切り取り
CTRL + Y	やり直し
CTRL + Z	元に戻す

グリッドビューコマンド

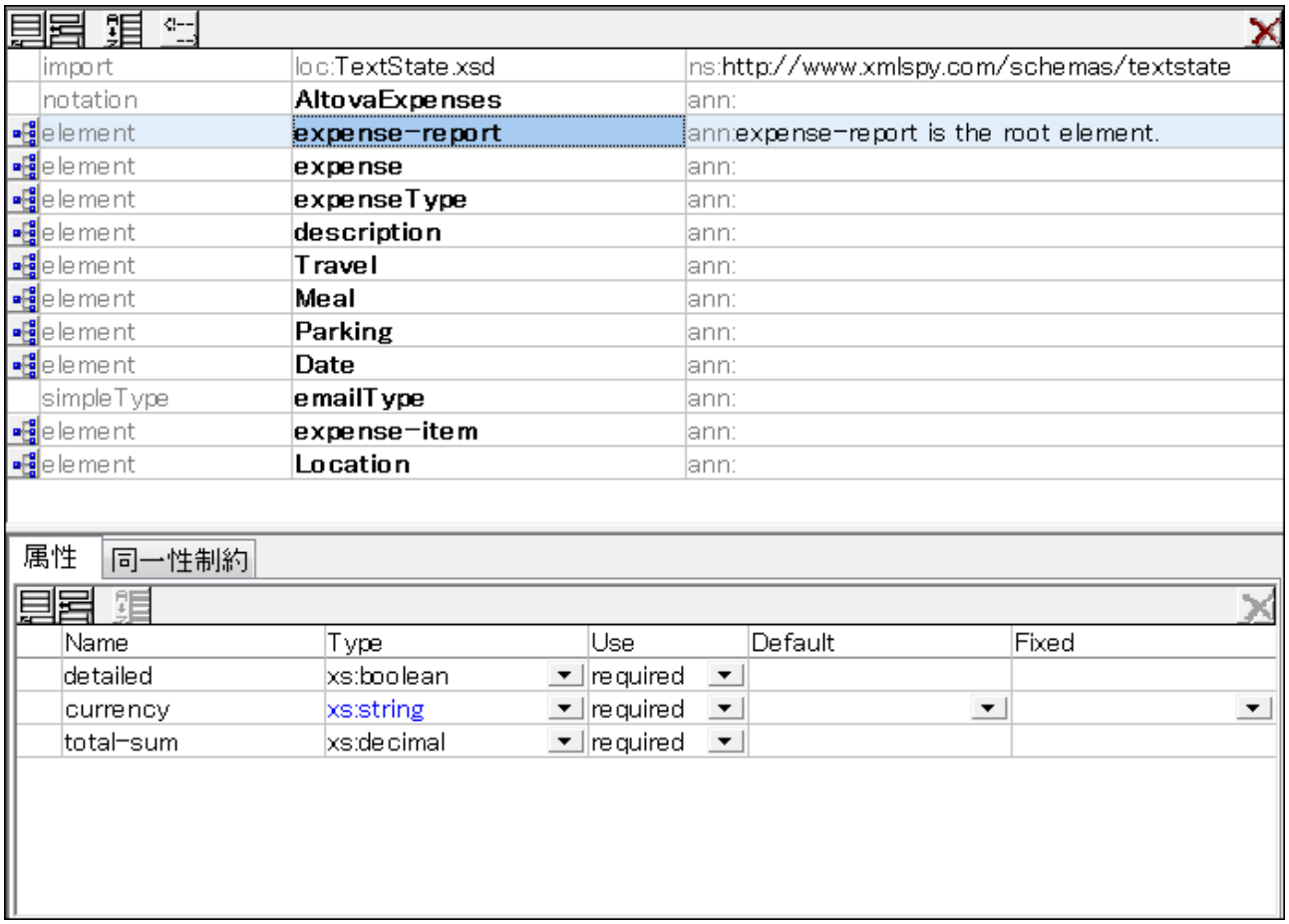
CTRL + D	CDATA の追加
CTRL + E	要素の追加
CTRL + I	属性の追加
CTRL + M	コメントの追加
CTRL + T	テキストの追加
CTRL + Shift + D	CDATA の挿入
CTRL + Shift + E	要素の挿入
CTRL + Shift + I	属性の挿入
CTRL + Shift + M	コメントの挿入
CTRL + Shift + T	テキストの挿入
CTRL + Alt + D	子 CDATA の追加
CTRL + Alt + E	子要素の追加
CTRL + Alt + I	子属性の追加
CTRL + Alt + M	子コメントの追加
CTRL + Alt + T	子テキストの追加

(C) 2015-2021 Altova GmbH

4.4 スキーマ ビュー

Altova W eb サイト: XML スキーマエディター

スキーマビューではXML スキーマの参照と編集を行うことができます(以下のスクリーンショットを参照)。グラフィカルなインターフェイスと一般的な GUI の機能によりスキーマの構築を素早く正確に行うことができます。スキーマビューには(i) スキーマ コンポーネント間の構造的なリレーションシップをデザインするための上部ペインと(ii) 上部ペインで選択されたコンポーネントに関連する定義を行うための下部ペインが表示されます。スキーマビューには、妥当なスキーマを作成するためのコンポーネント、詳細、ファセット入力ヘルパーも搭載されます。



上部ペイン: スキーマのデザイン

スキーマビュー上部のペインでは2 つのモードを切り替えることができます:

- スキーマ概要[164]: スキーマの全グローバルコンポーネント(グローバル要素や複合型など) がシンプルなテーブルリストで表示されます(スクリーンショットを参照[164])。グローバルコンポーネントのアイコンをクリックすることで、そのグローバルコンポーネントがコンテンツ モデルビューで表示されます。すべてのグローバルコンポーネントがコンテンツ モデルを持てるわけではない(例えば単純型) ということにご注意してください。
- コンテンツ モデルビュー[176]: 選択されたグローバルコンポーネントのコンテンツ モデルが表示されます(スクリーンショットを参照[176])。ペイン左上にあるグローバルの表示アイコンをクリックすることで、スキーマ概要が表示されます。

コンテンツ モデル ビューへの切り替え: コンテンツ モデルを含むグローバル コンポーネントに対して利用でき、グローバル コンポーネントのコンテンツ モデルがコンテンツ モデル ビュー[176]にて表示されます。

グローバルの表示: コンテンツ モデル ビューで選択することができ、スキーマ概要[164]への切り替えを行います。

下部ペイン: 属性、アサーション、同一性制約

スキーマ ビューの下部には、(上部ペインの) デザインで選択されているコンポーネントの属性、アサーション、同一性制約[195]の定義が表示されます (スクリーンショットを参照[195])。このペインは便宜上 AAIDC ペインと呼びます。

- XSD 1.0 モードでは、(i) 属性[196]と (ii) 同一性制約[202]という 2 つのタブが下部ペインに表示されます。
- XSD 1.1 モードでは、(i) 属性[196]、(ii) アサーション[199]、(iii) 同一性制約[202]の 3 つのタブが下部ペインに表示されます。

AAIDC ペインはスキーマ概要で常に表示されるほか、コンテンツ モデル ビューにて表示されることもあります。コンテンツ モデル ビューでは、3 種類の定義 (属性、アサーション、同一性制約) すべてを AAIDC ペインではなくデザイン内部に表示することもできます。スキーマ デザイン ツール バーの (i) **属性を図に表示**、(ii) **アサーションを図に表示**、(iii) **制約を図に表示** アイコンを切り替えることで、これらの表示を切り替えることができます。スキーマ表示構成ダイアログ (「スキーマ設計 | 表示設定」[1264]) にてこれらの設定を指定することもできます。
AAIDC ペインのすべての定義型が図内に表示され、コンテンツ モデル ビューにて下部ペインが表示されなくなります。

スキーマ設定

スキーマ設定ダイアログ (「スキーマ設計 | スキーマ設定」[1256]) へはスキーマ ビューからアクセスすることができ、アクティブなスキーマに対するグローバルな設定を定義することができます。これらの設定は `xs:schema` 要素の属性に対応します。

このセクションの構成

このセクションは、以下に示されるサブセクションにより構成されます:

- XSD モード: XSD 1.0 と1.1[161]: 2 つの編集モードから選択を行えます。
- スキーマ概要[164]: グローバル コンポーネントのプロパティを編集します。
- コンテンツ モデル ビュー[176]: 個々のグローバル コンポーネントのコンテンツ モデルを編集します。
- 属性、アサーション、同一性制約[195]: コンポーネントに含まれるこれらプロパティを定義します。
- 入力ヘルパー[209]: コンポーネントに含まれる様々なプロパティを素早く定義します。
- スマート制約[223]: 基底型から派生した型を視覚的に作成ならびに編集します。
- xml の使用: 固定された属性[228]: `base`、`id`、`lang`、`space` 属性をスキーマ コンポーネントへ視覚的に追加します。
- 戻ると進む: 位置の移動[230]: 以前に表示された内容を表示するためのスキーマ ビュー機能について説明します。

SchemaAgent へ接続

XMLSpy からは SchemaAgent への接続を行い、他のスキーマから得られたコンポーネントを GUI に表示して、これらコンポーネントを編集されているスキーマにて使用することができます。XMLSpy にて SchemaAgent を使用する方法については、「SchemaAgent を使用する」[397]のセクションを参照ください。

スキーマの中を検索

スキーマの中を検索機能により、スキーマ内部のインテリジェントな検索 (スキーマに関連した条件による絞り込み) を行うことができます。例えば、検索の結果を特定種類のコンポーネントへ絞り込むことで、より効率的な検索を行うことができます。スキーマの中を検索機能の詳細については、「DTD と XML スキーマ」[415]のセクションを参照ください。

4.4.1 XSD モード: XSD 1.0 と 1.1

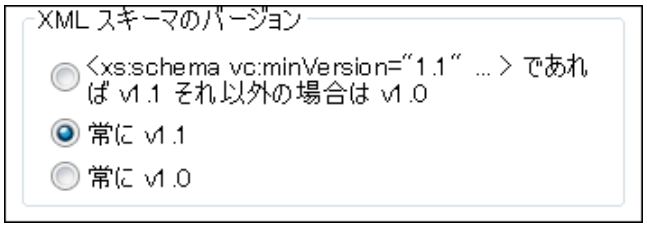
スキーマビューでは、 XML スキーマ(XSD) が編集されるか、そして、XML Schema 1.0 仕様(XSD 1.0) または XML Schema 1.1 仕様(XSD 1.1) に従い検証されるかを選択することができます。**XSD モード** は、次の2 つの設定をベースにした編集で使用されます: アプリケーション内、他のXSD ドキュメント内。

XSD モードの選択

XSD モードは、アクティブなドキュメントのためのスキーマビューの編集と検証機能(XSD 1.0 または1.1) を決定します。以下のどちらかを行うことができます: (i) 選択されたモードですべてのXSD ドキュメントをスキーマビュー内で編集されることのできるアプリケーション全体の設定にする、または(ii) XSD ドキュメント内のXSD バージョン番号を保存して、この情報に従い、アプリケーションは、XSD モードを自動的に選択します。

アプリケーション全体モード

アプリケーション全体設定は、オプションダイアログ のファイルセクションで行います(**「ツール | オプション」**、 *下のスクリーンショット参照*)。 *バージョン 1.0* または *バージョン 1.1* ラジオボタンを選択すると、選択されたモードはアプリケーション全体モードになります。スキーマビュー内で開かれたすべてのXML スキーマドキュメントはこのモードで編集することができます。(*v1.1 を選択する場合は、if <xs:schema vc:minVersion="1.1"の場合です。その他の場合は、v1.0 設定を選択します。* モードは、ドキュメント内の情報により、アプリケーション全体ではありません。この点に関しての情報に関しては下の他のセクションとドキュメント固有のモードを参照してください。)



オプションダイアログのXML スキーマ バージョン 設定内で選択することによりいつでも 2 つのアプリケーション全体モード間(*バージョン 1.0* および *バージョン 1.1*) を切り替えることができます (*上のスクリーンショット*)。

メモ　現在の設定が、アプリケーション全体設定である場合、設定およびモードの切り替えは、スキーマデザイン ツール バー(*次のセクション参照*) 内の**XSD 1.0** ツール バーボタンまたは**XSD 1.1** ツール バーボタンを使用して行うことができます。 モードの切り替えは、一時的であり、ドキュメントが再ロードされると、アプリケーションの全体モードに切り返されます。 ビューの変更またはスキーマビューが更新されると(**ファイル | 再ロード**を介して)再ロードが実行されます。

ドキュメント固有のモード

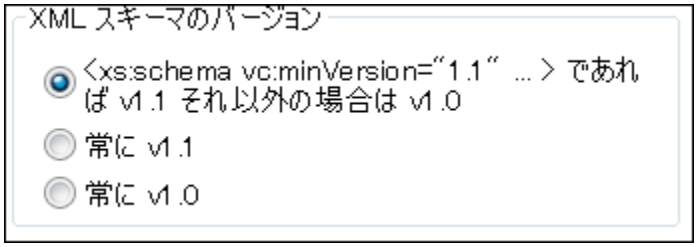
XSD ドキュメント内にXSD モード 情報を保存することを選択できます。ドキュメントがロードされると、自動的にスキーマビューのXSD モードへの切り替えを有効化します。スキーマデザイン ツール バー内の **XSD 1.0** または**XSD 1.1** ボタンをクリックすることにより、XSD モード 情報をXSD ドキュメントに追加することができます。(*下のスクリーンショット*). これを行うにあたり、選択されたモードは、 トップレベル `xs:schema` 要素の `vc:minVersion` 属性内に保存されます (`vc:minVersion` の値は、手動でテキストビューに追加することもできます)。

メモ　`vc:minVersion` 属性 が存在する場合は、名前空間 `http://www.w3.org/2007/XMLSchema-versioning` 内に存在する必要があります。この場合は、XML スキーマドキュメントは、この名前空間に`vc:` 名前空間 プレフィックスをバインドする名前空間宣言が必要になります。**XSD 1.1** ツール バーボタンを使用すると、名前空間は自動的に追加されます(*上のス*

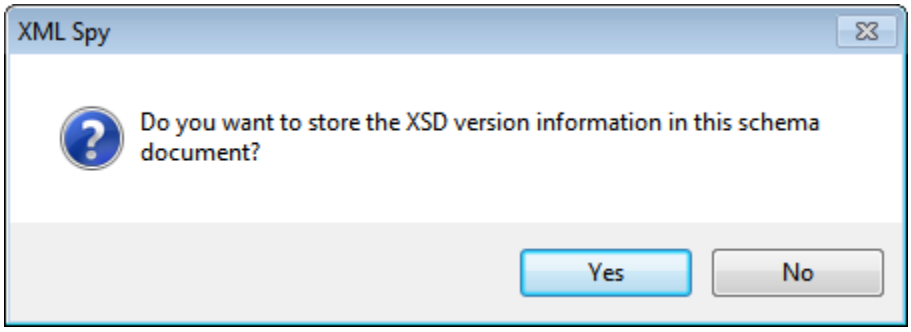
クリーンショット)。ドキュメント内の他のノード名がこの名前空間に存在しない場合、**XSD 1.0** ツールバーボタンをクリックすると、名前空間宣言が削除されます。

ドキュメント固有のモードをアクティブ化し、ドキュメントのXSD モードを指定するには、次を行います:

1. *ドキュメント固有のモードのアクティブ化:* オプションダイアログのファイル セクション(「**ツール | オプション**」)で、XML スキーマ バージョン オプション(*<xs:schema vc:minVersion="1.1"の場合は、v1.1 に、その他の場合は v1.0* に設定します(*下のスクリーンショット参照*) 。これは、スキーマビュー内のXSD モードが`xs:schema` 要素の`vc:minVersion` 属性に対して設定されるようにXMLSpy に示します。



2. *ドキュメントの XSD バージョンの指定:* スキーマデザイン ツールバー内の(*上のスクリーンショット*) **XSD 1.0** または**XSD 1.1** ボタンをクリックします。確認 ダイアログがポップアップします (*下のスクリーンショット*)。



3. 「**はい**」をクリックすると、次の結果を引き起こします: (i) `xs:schema` 要素の `vc:minVersion` 属性 に対応する値を入力します。(ii) **XSD 1.1** が選択されている場合、`vc:` 名前空間 プレフィックスへのバインドと共に `XMLSchema-versioning` 名前空間が宣言されます。**XSD 1.0** が選択されている場合、`XMLSchema-versioning` 名前空間内に他のノードが存在しないと、名前空間宣言は削除されます。XML スキーマドキュメントは、XSD バージョン番号を含むようになります。 ファイルを保存する際、XSD モード 情報も保存されます。ファイルを再度開くまたは再ロードすると、スキーマビューは、`xs:schema` 要素の`vc:minVersion` 属性に含まれているとおり自動的にドキュメントのXSD モードに切り替えられます。

メモ ドキュメント固有のモード オプションが選択され、そしてXSD ドキュメントが`vc:minVersion` 属性 または`1.0` または`1.1` 以外の`vc:minVersion` 属性の値を持つ場合、 スキーマビューのデフォルトはXSD 1.0 モードです。

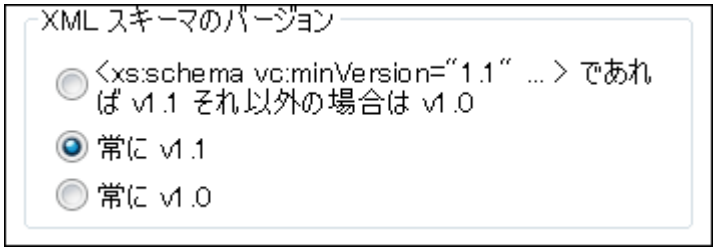
メモ `vc:minVersion` 属性 と `xsd:version` 属性を混同しないようご注意してください。前者は、XSD バージョン番号を有し、後者は、バージョン番号を有します。

新しいドキュメントのXSD モード

新しいXSD ドキュメントを作成する場合、XSD 1.0 またはXSD 1.1 ドキュメントとして作成するかを問われます。XSD 1.1 が選択されると、新しいドキュメントが属性 `/xs:schema/@vc:minVersion="1.1"` と共に作成され、`vc:` 名前空間 プレフィックスへのバインドを持つ `XMLSchema-versioning` 名前空間が宣言されます。XSD 1.0 が選択されると、`vc:minVersion` 属性または `XMLSchema-versioning` 名前空間宣言が追加されません。しかしながら、スキーマビュー内でどのXSD モードが有効化されるかは、オプションダイアログのファイルのセクションで選択されている XML スキーマ バージョンによります (「**ツール | オプション**」)。 これら2つの設定どのように相互に作用するかについての詳細は、次のセクションを参照してください。

有効化されたXSD モード

XSD モード が有効化される場合、スキーマビュー内 の以下に影響されます(i)　存在/不在—そして、存在する場合、 / XSD ドキュメントの`xs:schema/@vc:minVersion` 属性の値、および(ii) オプションダイアログのファイル セクションで選択された XML スキーマ バージョン オプション(「ツール | オプション」、*下のスクリーンショット*)。



以下のシチュエーションが考えられます。下のテーブルの *XML スキーマ バージョン*は、上に表示されるXML スキーマ バージョン ペイン内の選択を参照します。下のテーブル内の`vc:minVersion` 値 はXML スキーマドキュメント内の`xs:schema/@vc:minVersion` 属性の値を参照します。

XML スキーマ バージョン	`vc:minVersion` 属性	XSD モード
常にv1.0	存在しません、または すべての値と存在します	1.0
常にv1.1	存在しません、または すべての値と存在します	1.1
@ vc:minVersion の値	属性は、`1.1` の値を持ちます	1.1
@ vc:minVersion の値	属性は、存在しません、または 属性 は、`1.1` 以外の値と存在します	1.0

メモ　最初の2 つの行で説明されている、シチュエーションの場合、 XSD 1.1 スキーマをXSD 1.0 モード内で開くこと、またその逆は可能です。矛盾の扱い方については下で説明されています。

XSD モード 機能

インターフェイスおよびスキーマビューの 編集機能は、有効化されるXSD モード (XSD 1.0 またはXSD 1.1) により変更されます。

XSD 1.0 モードが有効化される場合:

- 新しいXML Schema 1.1 コンポーネント およびプロパティ への編集サポートを使用することができません。しかしながら、XSD 1.1 コンポーネント またはプロパティがXSD ドキュメント内で既存の場合、これらは表示され、削除されることもできます。
- 検証は、XSD 1.0 仕様に対して行われます。ですから、 XSD 1.1 コンポーネント またはプロパティが(既に) スキーマ内に存在する場合、検証 エラーが報告されます。

XSD 1.1 モード が有効化されていると、 XML Schema 1.1 仕様に関するXML Schema 1.1. 検証のすべての機能のための編集サポートが提供されます。

XSD 1.0 モード内でのXSD 1.1 機能の扱いかた

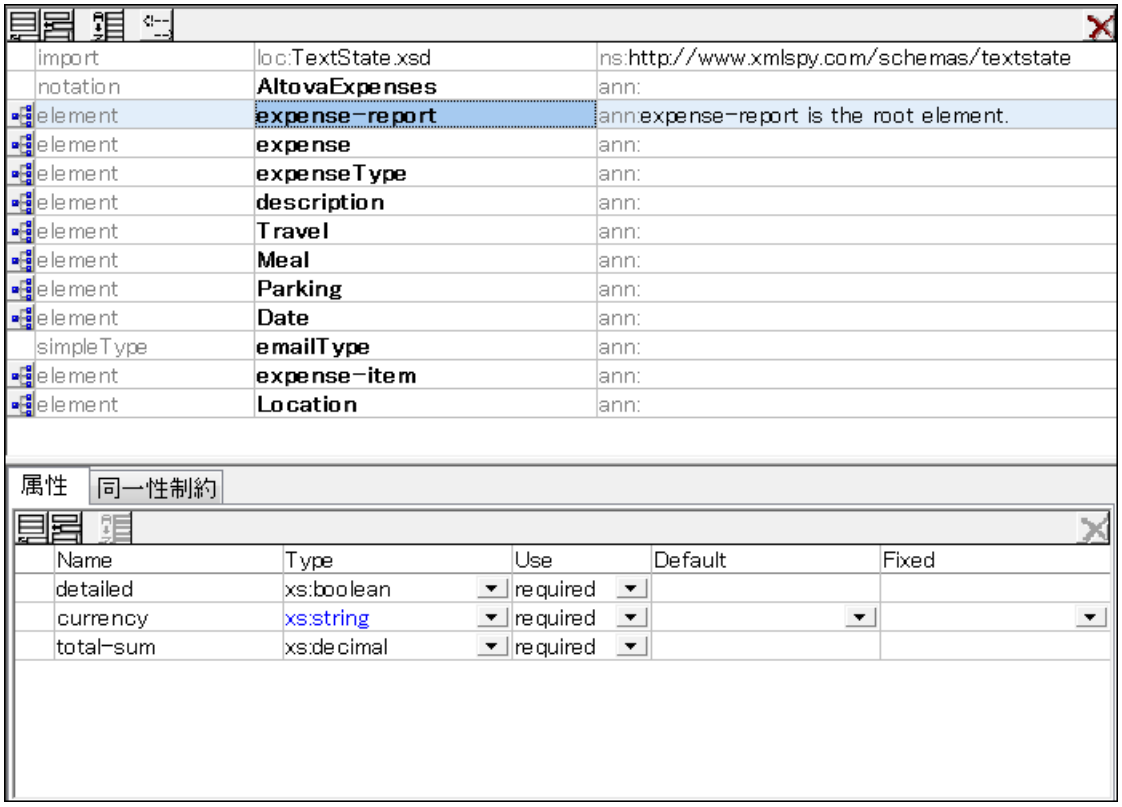
XSD 1.0 によりサポートされないXSD 1.1 機能(例えば アサーション)がドキュメント内に存在する場合、その機能が表示され、そして以下に説明されるXSD 1.0 モードにより取り扱われます。

- *アサーション:*　選択された単純型でアサーションが1 つでも存在する場合、ファセット 入力ヘルパーで*アサーション* タブが存在します。アサーションの削除以外の編集を行うことはできません。

- *アサート:* 存在する場合、複合型ダイアグラム内でアサーションが表示されます。AAIDC ペインでアサーション タブを使用することはできません。また、アサーションはコンテキスト メニューを介し 追加されることはできません。削除以外のプロパティの編集を行うことはできません。
- *属性:* 存在する場合、新しい プロパティ inheri テーブル が表示されます。空の値を選択する以外の編集を行うことはできません(これはより効果的なプロパティの削除方法です)。
- *複合型:* 存在する場合 新しい プロパティ defaultAttributesApply が表示されます。in 詳細入力ヘルパー。空の値を選択する以外の編集を行うことはできません(これはより効果的なプロパティの削除方法です).
- *ドキュメント:* 新しいXSD 1.1-固有のコンポーネント およびプロパティは、スキーマビュー内ドキュメントに含まれていません。
- *ファセット:* 未知のファセットは、検証 エラーを引き起こし、そして赤で表示されます。
- *スキーマ内の検索:* 新しいXSD 1.1-固有のコンポーネント およびプロパティは、無視されます。
- *ID 制約 (IDC):* プロパティ isRef が表示されます。参照の場合、はオフに切り替えることができます。名前のIDC がオフに切り替えられると、
- *複数の代替 グループ:* コンボボックスは、単一の代替 グループを選択します(XSD 1.0 内では単一の代替 グループのみが許可されています)。
- *オープン コンテンツ:* 存在する場合はダイアグラムに表示されます。コンテキスト メニューを介して追加されません。削除以外の編集を行うことはできません。 デフォルトのオープンコンテンツが、複合型内部で表示されません。
- *オーバーライド:* 存在する場合、グローバルグリッドを表示します。これは、 メニューを介して追加されません。削除以外の(ロケーションの)編集を行うことはできません。 コンポーネント (xs: overri de の子) は、無視されます、そして コンポーネント 入力ヘルパーに含まれません。
- *スキーマ設定:* 存在する場合、新しい プロパティ defaultAttributes および xpathDefaultNamespace は、ダイアログ内に表示されます。空の値を選択する以外編集は不可能です(これはより効果的なプロパティの削除方法です)。
- *単純型:* 未知の型は検証 エラーを引き起こし、赤字で表示されます。
- *型 代替:* 存在する場合ダイアグラムに表示されます。コンテキスト メニューを介して追加されません。削除する以外 (プロパティの)の編集を行うことはできません。
- *ワイルドカード:* 新しい プロパティが存在する場合は表示されます。空の値を選択する以外編集は不可能です(これはより効果的なプロパティの削除方法です)。

4.4.2 スキーマ概要

スキーマ概要ではスキーマ内に含まれるすべてのグローバル コンポーネント (import 要素、グローバル要素、複合型など) が表示されます(以下のスクリーンショットを参照)。



グローバルコンポーネントの挿入や追加、削除のほか、プロパティを修正することができます。プロパティを修正するには、スキーマ概要リストにてグローバルコンポーネントを選択してください。
選択されたグローバルコンポーネントのプロパティは、詳細入力ヘルパー[213]、ファセット入力ヘルパー[215]、または属性/アサーション/同一性制約（AAIDC）ペイン[195]で編集することができます。

コンテンツモデルを含んでいるグローバルコンポーネントには、グローバルコンポーネント リストの左に「**コンテンツ モデル ビューへの切り替え**」アイコンが表示されます。このアイコンをクリックすることで、そのコンポーネントがコンテンツ モデル ビュー[176]で表示され、コンポーネントのコンテンツ モデルを編集することができます。

コンテンツ モデル ビューへの切り替え: コンテンツ モデルが含まれるグローバルコンポーネントにて表示され、コンポーネントのコンテンツ モデルをコンテンツ モデル ビュー[176]にて表示します。

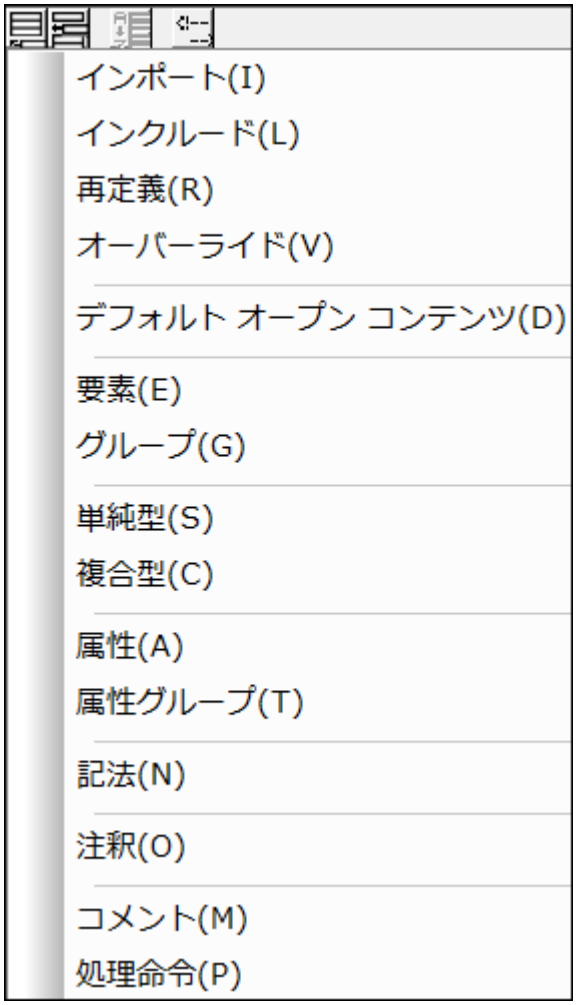
グローバルの表示: コンテンツ モデル ビューにて表示され、クリックすることでスキーマ概要[164]が表示されます。

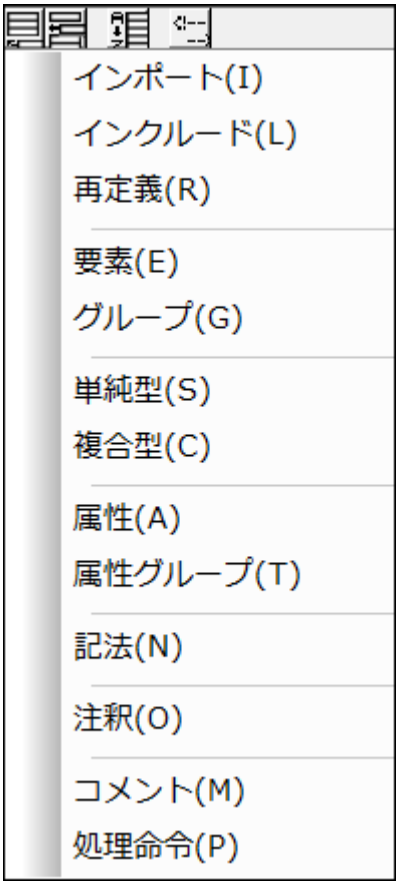
このセクションでは、最初にスキーマ概要のGUI メカニズム[165]が説明され、各種グローバルコンポーネント[169]の特徴について記述されます。

4.4.2.1 GUI メカニズム

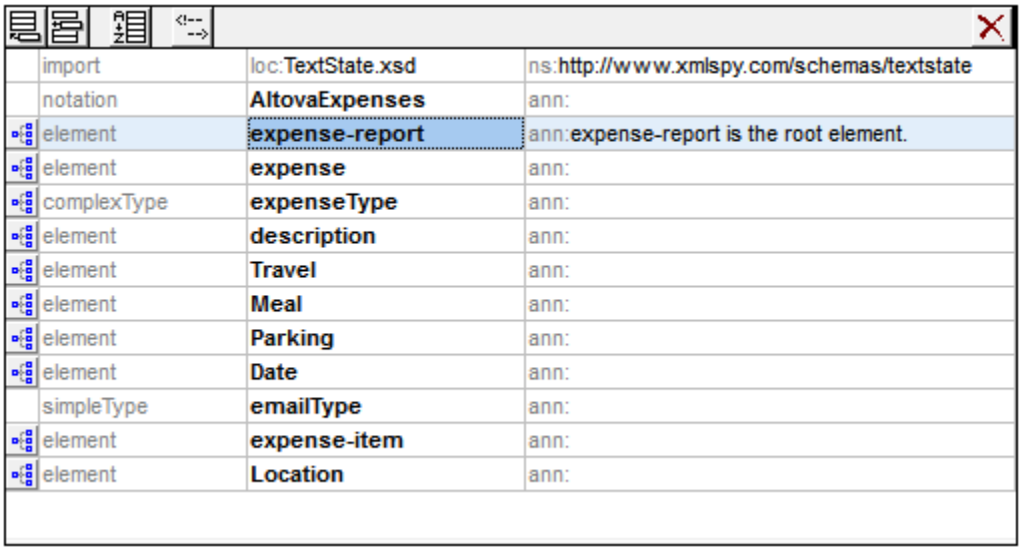
グローバルコンポーネントは、トップレベルの`xs:schema`要素の子として追加されます。グローバルコンポーネントは*下にあるアイコンおリストを確認してください*）ペイン上部にある「**追加**」アイコンまたは「**挿入**」アイコンをクリックし、追加するグローバルコンポーネントをメニューから選択することにより追加することができます（*以下のスクリーンショットを参照*）。

以下のスクリーンショットでは、追加可能なグローバルコンポーネントが表示されています: XSD 1.1 モードで追加することのできるグローバルコンポーネントを左側に、XSD 1.0 モードで追加することのできるコンポーネントが右側に示されています。(*オーバーライド*と*デフォルト オープン コンテンツ*はXSD 1.1 の機能です)。





追加することのできるグローバルコンポーネントの数に制限はありません。スキーマ内に含まれているすべてのグローバルコンポーネントがスキーマ概要のテーブルに表示されます(*以下のスクリーンショットを参照*)。



	import	loc:TextState.xsd	ns:http://www.xmlspy.com/schemas/textstate
	notation	**AltovaExpenses**	ann:
	element	**expense-report**	ann:expense-report is the root element.
	element	**expense**	ann:
	complexType	**expenseType**	ann:
	element	**description**	ann:
	element	**Travel**	ann:
	element	**Meal**	ann:
	element	**Parking**	ann:
	element	**Date**	ann:
	simpleType	**emailType**	ann:
	element	**expense-item**	ann:
	element	**Location**	ann:

スキーマ概要での編集

スキーマ概要の編集機能に関する以下の点に留意してください：

- スキーマ概要に表示されたリストはドラッグ アンド ドロップにより並び替えることができます。
- キーボードの矢印キーと「タブ」キーを使用することで、ナビゲーションを行うことができます。
- グローバル コンポーネントや属性、アサーション、同一性制約を切り取り/コピーして、ダイアグラム内の別の位置や、他のダイアグラムに貼り付けることができます。またスキーマからスキーマに張り付けることができます。
- コンポーネントを右クリックすることで表示されるコンテキスト メニューにより、そのコンポーネントの注釈データを切り取り、コピー、貼り付け、削除することができます。
- グローバルコメントやグローバルな注釈内で改行を入力するには、「Ctrl」+「Enter」を押下してください。タブを入力するには、「Ctrl」+「タブ」を押下してください。

スキーマ概要と関連するアイコン

	グローバル コンポーネントの追加: グローバルコンポーネントのリストの最後に新たなグローバルコンポーネントを追加します。定義によりコンポーネントがドキュメントの最初に出現する必要のある場合、リストの最初に追加されます。
	グローバル コンポーネントの挿入: 選択されているコンポーネントの直前にグローバルコンポーネントを追加します。定義によりコンポーネントがドキュメントの最初に出現する必要のある場合、リストの最初に追加されます。
	並べ替え: コンポーネントの並び替えダイアログが表示され、実際に並び替えを行う前に、並び替えを行うための条件（種類の前に名前で並び替えを行う、またはその逆）を指定することができます。*以下の記述を参照ください。*
	コメント: グローバルコメントの表示を複数行で行うのか、または1 行で行うのかを指定することができます。*以下の記述を参照ください。*
	コンテンツ モデル ビューへの切り替え: コンテンツモデルを持つグローバルコンポーネントのために使用することができます。[コンテンツモデルビュー](#)[176] 内のグローバルコンポーネントのコンテンツモデルを開きます。
	グローバルの表示: コンテンツモデルビュー内で使用が可能です: [スキーマ概要](#)[164] に切り替えます。

スキーマ概要とコンテンツ モデル ビューの切り替え

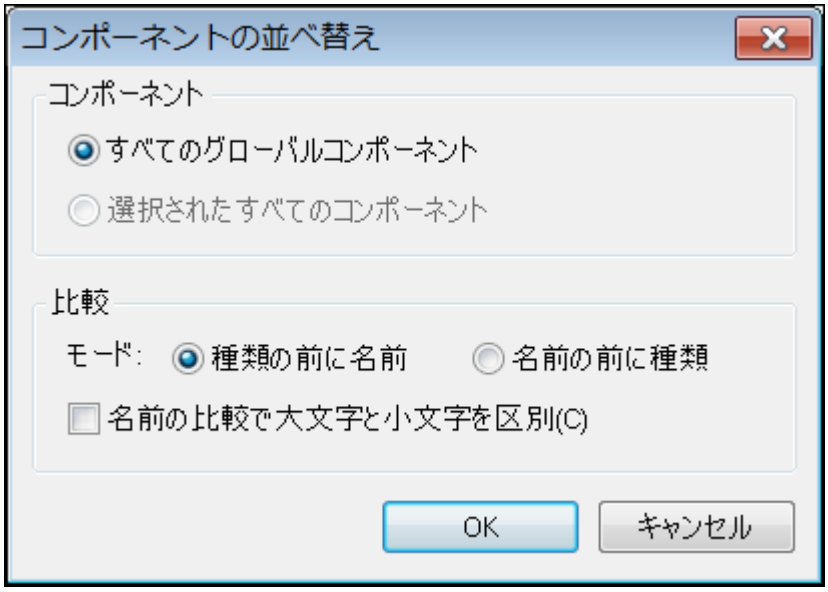
(複合型、要素宣言、モデル グループなどの) グローバル コンポーネントには、コンポーネント名の左側に「**コンテンツ モデル ビューへの切り替え**」アイコンが表示されます (*上記にあるアイコンのリストを参照ください*)。これにより、コンポーネントの構造とコンテンツが記述されるコンテンツ モデルを、これらグローバル コンポーネントに含めることができることが示されます。

このアイコンをクリックすることで、スキーマ概要から、そのグローバル コンポーネントのコンテンツ モデル ビュー[176]への切り替えが行われます。その他のグローバル コンポーネント (注釈や単純型、属性グループなど) にコンテンツ モデルは含まれておらず、「**コンテンツ モデル ビューへの切り替え**」アイコンも表示されません。**グローバルの表示** アイコンをクリックすることでコンテンツモデルビューからスキーマ概要に切り替えることができます (*上のアイコンのリストを確認してください*)。

グローバル コンポーネントの並べ替え

スキーマ概要ツール バーの**並び替え**アイコンをクリックすることで、グローバル コンポーネントの並べ替えを行うことができます (*上記のアイコン リストを参照ください*)。コンポーネントの並べ替えダイアログが表示され (*下のスクリーンショット参照*)、並べ替え可能なすべてのグローバル コンポーネントを並べ替えるのか、または選択されたコンポーネントのセットだけを並べ替えるのか指定することができます。「**Shift**」キーを押しながらクリックすることでコンポーネントの範囲を選択することができ、「**Ctrl**」キーを押しながらクリックすることで、選択対象となるコンポーネントを追加することができます。

メモ ドキュメントの最初に出現しなければならないグローバル コンポーネント (`include` や `import` など) は並び替えの機能により影響を受けません。これらのコンポーネントは、並べ替えを行うことのできるグローバル コンポーネントの範囲に含まれません。

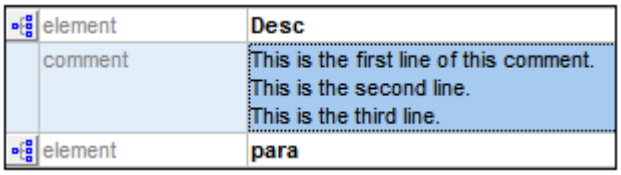


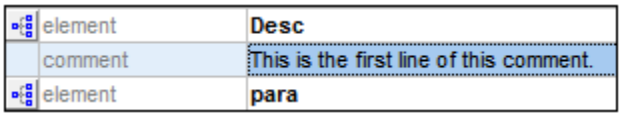
並べ替えを行う範囲を選択した後には、(*名前か型の前のアルファベット順で並べ替えを行うのか、またはコンポーネントの種類により並べ替えを行った後に名前による並べ替えを行うか*) を選択することができます。

並べ替えの順序はスキーマのテキストに反映されます。

グローバル コメント: 行の表示モード

グローバル コンポーネントは複数行のテキスト フィールド、または 1 行のテキスト フィールドに表示することができます (*以下のスクリーンショットを参照*)。





これら2 種類の表示モードを切り替えるには、スキーマ概要ペインの上部にある「コメント」アイコンをクリックして、目的のオプションを選択してください。コメントのテキスト内部に改行を入力する（そして複数行のコメントを作成する）には、「Ctrl」+「Enter」キーを押下してください。コメントを1 行で表示するモードの場合、カーソルを複数行コメントに移動させることで、複数行のコメントがポップアップで表示されます。

4.4.2.2 グローバル コンポーネント

グローバルコンポーネントとは、トップレベルの xs:schema 要素の子として追加されるコンポーネントのことです（それに対してローカルコンポーネントは、その他のコンポーネント内部で作成されます）。複合型や要素、属性といったグローバルコンポーネントへは、スキーマ内にあるその他のコンポーネントから参照することができます。

スキーマ概要内でのグローバル コンポーネントの作成

グローバルコンポーネントは通常スキーマ概要[164]で作成ならびに編集されます。スキーマ概要では追加[165] または[165] 挿入[165] アイコン[165]を介して追加されます。（グローバルコンポーネントがコンテンツモデルを有している場合 *下のテーブルを参照*）グローバルコンポーネントのコンテンツ モデルは、そのグローバルコンポーネントのコンテンツ モデル ビュー[176]にて作成ならびに編集することができます。（コンポーネント名の左側にあるコンテンツ モデル ビューへの切り替えアイコンをクリックすることで、コンテンツ モデル ビューが表示されます[176]）。

スキーマ概要にて作成されたグローバルコンポーネントによってはコンポーネント 入力ヘルパー[209]へも追加されます。コンポーネントにコンテンツモデルが含まれる場合、コンポーネント 入力ヘルパーでその名前をダブルクリックすることでコンポーネントがコンテンツ モデル ビュー[176]で開かれ、編集を開始することができます。

グローバルコンポーネントが型定義を持つ場合、コンポーネントを右クリックして「型の定義へ移動」を選択することで、そのコンポーネントの型定義へ移動することができます。ビルトインの単純型の場合、単純型に関する情報が含まれたメッセージボックスが表示されます。

メモ　コンテンツ モデル ビューで編集作業をしながらもグローバルコンポーネント（要素、属性、単純型、複合型、モデル グループ）を作成することができます。ウィンドウを右クリックして、「**新規グローバル**」|「**作成するグローバル コンポーネントの種類**」を選択してください。

メモ　コンテンツ モデル ビューにて編集作業中、ローカル要素をグローバル要素へ、または要素に属性や子要素が含まれる場合にはグローバル複合型へ変換することもできます。ローカル要素が選択されている状態で、ウィンドウを右クリックすることで表示されるコンテキスト メニューから「グローバル化 | 要素」または「グローバル化 | 複合型」を選択してください。

グローバル コンポーネント	スキーマにおける位置	コンテンツ モデル
include	先頭	なし
import	先頭	なし
redefine	先頭	なし

override 1.1	先頭	なし
defaultOpenContent 1.1	Includes、Imports、Redefined、Overrides の後で、それ以外すべてより前	あり
element	defOpenCont 以降ならどこでも	あり
group	defOpenCont 以降ならどこでも	あり
simpleType	defOpenCont 以降ならどこでも	なし
complexType	defOpenCont 以降ならどこでも	あり
attribute	defOpenCont 以降ならどこでも	なし
attributeGroup	defOpenCont 以降ならどこでも	なし
notation	defOpenCont 以降ならどこでも	なし
annotation	defOpenCont 以降ならどこでも	なし
コメント	どこでも	なし
処理命令	どこでも	なし

スキーマビューでこれらのコンポーネントを編集する際に留意すべき点を以下に記します:

インクルード、インポート、再定義、オーバーライド

これら 4 つのグローバルコンポーネントにより、他のスキーマドキュメントを現在編集中のスキーマドキュメントにて再利用することができます。

- インクルードにより、現在編集中のドキュメントと同じターゲット名前空間を持つドキュメントを再利用します。
- インポートにより、現在編集中のドキュメントとは異なるターゲット名前空間を持つドキュメントを再利用します。
- 再定義とオーバーライドはインクルードの一種で、現在編集中のターゲット名前空間と同じターゲット名前空間を持つドキュメントを再利用するという点では同じですが、インクルードされたスキーマの一部を修正するという点で異なっています。再定義は XML Schema 1.0 の機能で、1.1 での利用は推奨されておらず、再定義を置き換える XML Schema 1.1 の機能としてオーバーライドが追加されました。

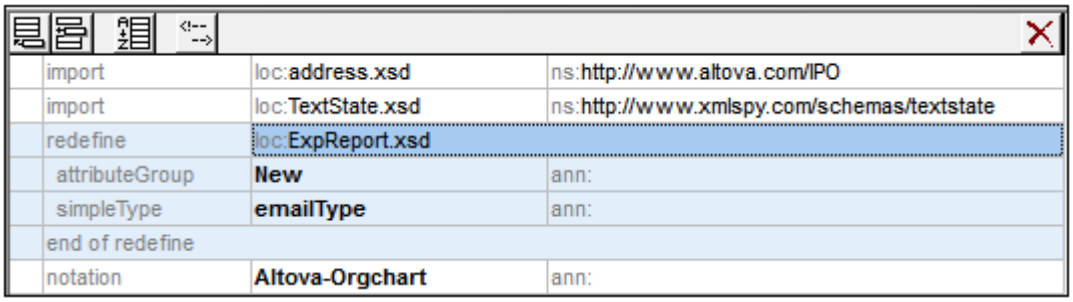
これら 4 つのすべてには再利用されるスキーマを指し示す schemaLocation 属性が含まれています。スキーマビューでこれらコンポーネントの loc フィールドをダブルクリックすると、再利用されるファイルの参照と、現在のドキュメントに対して相対となるパスをセットすることができます。更に import コンポーネントにはインポートされたスキーマのターゲット名前空間を示すための namespace 属性が含まれます。

現在編集中のスキーマドキュメントにてスキーマが（インクルード、インポート、再利用、またはオーバーライドにより）再利用されると、そのグローバルコンポーネント、名前空間、同一性制約が編集中ドキュメントのコンポーネント入力ヘルパー 209に表示されます。

再定義

redefine コンポーネントでは、複合型、単純型、モデルグループ、そして属性グループを修正することができます。再定義されるコンポーネントは、redefine コンポーネントの loc フィールドにて指定されます（以下のスクリーンショットでは、ExpReport.xsd スキーマ内にあるコンポーネントが再定義されます）。redefine コンポーネントが追加されると、グローバルコンポーネント リストの redefine と end of redefine 行の間に再定義するコンポーネントを追加する必要があります（*以下のスクリーンショットでは*、New と emailType コンポーネントが再定義されます）。スキーマ ExpReport.xsd 内に存在する 2 つのコンポーネントは現在のスキーマのために再定義されています。

(C) 2015-2021 Altova GmbH



import	loc:address.xsd	ns:http://www.altova.com/IPO
import	loc:TextState.xsd	ns:http://www.xmlspy.com/schemas/textstate
redefine	loc:ExpReport.xsd	
attributeGroup	New	ann:
simpleType	emailType	ann:
end of redefine		
notation	Altova-Orgchart	ann:

コンポーネントを再定義するには、以下を行います:

1. `end of redefine` 行を選択します。
2. スキーマ概要左上にある「挿入」アイコンをクリックします。
3. 定義するコンポーネントの種類（複合型、単純型、モデルグループ、属性グループ）を選択します。再定義コンポーネント内にコンポーネントが追加されます。
4. 再定義するコンポーネントと同じ名前を与えます。コンポーネントは再利用されるスキーマから得られたコンポーネントのすべてのプロパティを得ます。
5. 詳細ならびにファセット入力ヘルパーからプロパティを選択ならびに修正することができるほか、（コンポーネントにコンテンツモデルが含まれる場合）コンテンツモデルビューにてコンテンツモデルを修正することができます。

メモ　グローバルコンポーネントのリストやコンポーネント入力ヘルパー内から、ドラッグアンドドロップやコピーと貼り付けを行うことで、再定義するコンポーネントを挿入することもできます。

再定義されたコンポーネントは、スキーマ内にある他のコンポーネントから参照することもできます。

オーバーライド

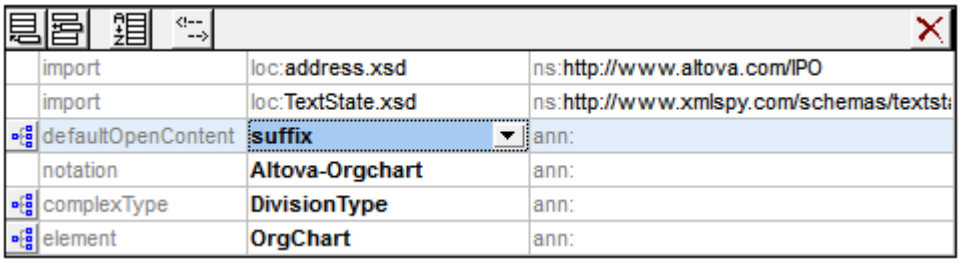
`override` 要素では、以下のコンポーネントを定義することができます: 複合型、単純型、グローバル要素、グローバル属性、モデルグループ、属性グループ、記法。`override` 要素内でこれらコンポーネントのうちの1つが定義されている場合、オーバーライドされるすべてのスキーマにある同じ型と名前を持つすべてのコンポーネントが、このコンポーネントによりオーバーライドされます。オーバーライドされたスキーマは、オーバーライドの `loc` フィールドにて指定されます。

(*上記参照*) オーバーライドは再定義と違い、再利用されるコンポーネントをベースにすることなく、最初から定義されたコンポーネントとなります。スキーマビューでは再定義を行うためのコンポーネントを追加するのと同様に、オーバーライドを行うためのコンポーネントを追加することになります。`end of override` 行の上にオーバーライドするコンポーネントを挿入し、そのプロパティを定義してください。*上記の再定義セクションを参照ください*。オーバーライドと再定義の主な違いは、オーバーライドにコンポーネントが追加されても、そのコンポーネントは再利用されたスキーマから得られたいかなるコンポーネントもベースにしていないということです。

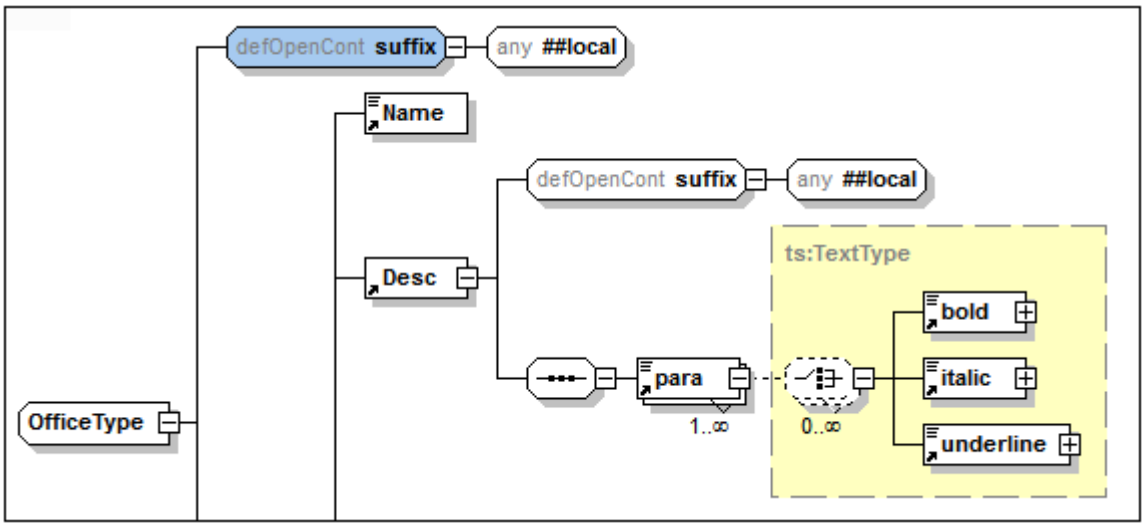
デフォルト オープン コンテンツ

`defaultOpenContent` 要素はXSD 1.1 に新たに追加された仕様で、混合または要素だけのコンテンツを含めることのできる複合型に追加することのできる1つ以上の要素を指定します。XSD 1.1 にて新たに追加された `openContent` 193 要素に似ており、`openContent` 要素が単一の複合型に対して適用されるのに対して、`defaultOpenContent` 要素はスキーマ内にあるすべての複合型に対して適用されます。

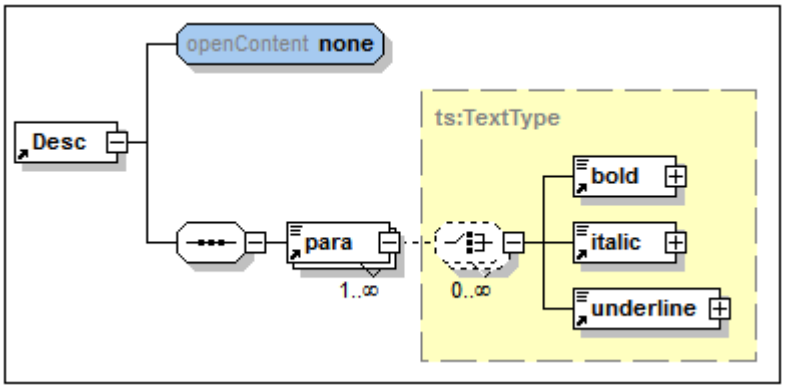
`defaultOpenContent` 要素はinclude、import、redefine の後、そしてコンポーネントの定義前に1回出現します（*以下のスクリーンショットを参照*）。要素には `interleave` または `suffix` という値を含めることのできる `mode` 属性が含まれています。デフォルトは、`interleave` です。



`defaultOpenContent` 要素には、コンテンツ モデルビューで編集することのできるコンテンツ モデルが含まれています。宣言されると `defaultOpenContent` 要素がスキーマ内にあるすべての複合型に対して適用されます。以下のスクリーンショットでは、`defaultOpenContent` が複合型の `OfficeType` ならびに `Desc` 要素に対して自動的に適用されていることが理解できます。



`openContent` 子要素を追加することで、その複合型に対して適用された `defaultOpenContent` 要素をオーバーライドすることができます。下のスクリーンショットでは `defaultOpenContent` 要素を持つ `Desc` 要素は（*上のスクリーンショット*）`defaultOpenContent` 要素をオーバーライドする追加された `openContent` 要素を持ちます。



グローバル要素（element）

スキーマ概要ではグローバル要素を作成することができます。グローバル要素にコンテンツ モデルが含まれる場合、グローバル要素のコンテンツ モデルビューにて定義することができます。要素が選択されている状態で、属性[195]、アサーション[199]、そして同一性制約[202]を AADC

ペイン[195]の対応するタブにて定義することができます。要素のファセット[215]ならびに詳細[213]入力ヘルパーにて定義することができます。グローバル要素は複合型から参照することができます。グローバル要素は複合型から参照することができます。

モデル グループ（group）

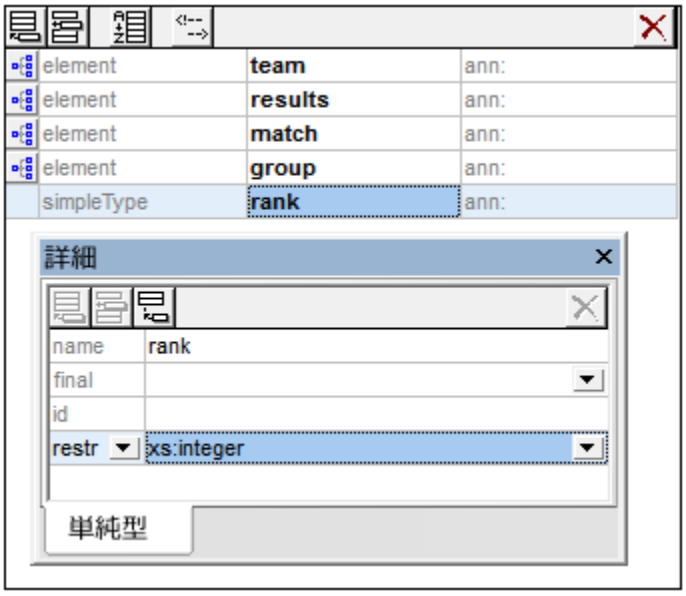
スキーマ概要では名前付きのモデル グループを作成し、複合型にて参照することができます。名前付きのモデル グループ（xs:group 要素）により、再利用可能なコンテンツ モデルを予め定義することができます。sequence グループ、choice グループ、または all グループのうちどれかを含めることができます。

スキーマ概要では、グループ コンポーネントを追加して、名前をつけた後にコンテンツ モデル ビューにてコンテンツ モデルを定義することで、名前付きモデル グループを作成することができます。名前付きモデル グループを複合型のコンテンツ モデルへ追加することができるようになります。

名前付き単純型（simpleType）

スキーマ概要では、名前付きの単純型を作成し、要素や属性の宣言から参照することができます（*以下のスクリーンショットを参照*）。

詳細入力ヘルパーでは、単純型のコンテンツ（restriction、list、union）と対応する種類（それぞれ基底型、アイテム型、メンバー型）を指定することができます。例えば以下のスクリーンショットでは、単純型の制約の基底型に xs:integer が指定されています。詳細については「詳細入力ヘルパー」[213]のセクションを参照ください。ファセットにより単純型の制約を行うには、ファセット入力ヘルパー[215]にあるオプションを使用してください。



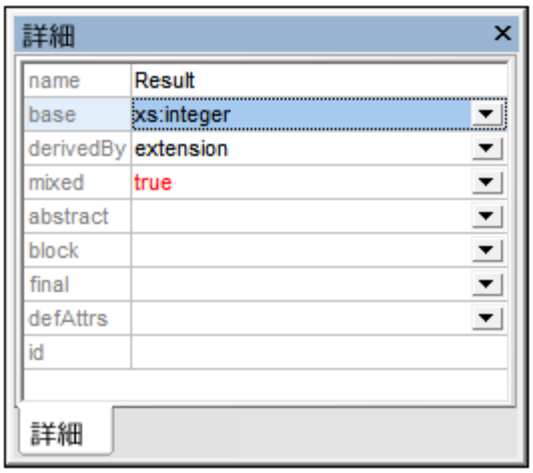
メモ　匿名型は、スキーマ概要[164]または コンテンツ モデル ビュー[176]にて、単純型コンテンツの属性または要素で宣言することができます。（詳細入力ヘルパーにある）derivedBy プロパティの値を restriction、list、または union に指定することで、その要素または属性宣言内部に匿名の単純型が作成されます。制約ファセットをファセット入力ヘルパーで、その他のプロパティを詳細入力ヘルパーで定義することができます。

名前付き複合型（complexType）

スキーマ概要では名前付きの複合型を作成し、要素宣言から参照することができます。名前付きの複合型が選択されている状態で、AAIDC ペイン[195]の対応するタブにて属性[195]とアサーション[199]を定義することができます。

複合型には4 種類のコンテンツを含めることができます（*以下のリストを参照*）。詳細入力ヘルパーにて各種コンテンツを指定し、必要に応じてコンテンツ モデル ビューにてコンテンツ モデル[176]を定義することができます。

- *単純型コンテンツ*:単純コンテンツの基底型をセットします(*以下のスクリーンショットを参照*)。mixed 属性には(デフォルト値の) false をセットする必要があり、以下にあるスクリーンショットでセットされた true は赤色で示されています。コンテンツモデルは許可されています。



- *要素のみのコンテンツ*: コンテンツモデルダイアグラム内に子要素を作成する。ベース型は存在しません。
- *混合コンテンツ*: mixed 属性を true にセットする必要があります。文字データは、子要素ノードとともに、要素内のどこにでも配置することができます。文字データはデータ型を持っていないため、基底型をセットすることはできません(*上のスクリーンショットを参照*)。コンテンツ モデルでは子要素を作成することができます。
- *空のコンテンツ*: 要素には文字データも子要素も含まれません。基底型は含まれず、mixed の値を false にする必要があります。空のコンテンツ要素のデータは、通常属性に収められます。

メモ 属性とアサーションは(AADC ペイン[195])にて 4 種類のコンテンツすべてに対してセットすることができます。

メモ ある要素に対するコンテンツ モデルをコンテンツ モデル ビュー[176]で作成することにより、その要素内部で匿名の複合型が作成されます。

グローバル属性と属性グループ(attribute、attributeGroup)

グローバル属性と属性グループはスキーマ概要にて追加されます。

- **グローバル要素のプロパティ**は属性の詳細入力ヘルパーにて定義されます。
- **グローバル属性グループ**を作成すると、グループに属性を以下のように追加することができます: (i) グローバル コンポーネントのリストにてグローバル属性グループを選択する。(ii) AADC ペイン[195]の属性タブにて属性を追加する。(iii) 各属性のプロパティを選択された属性の詳細入力ヘルパーにて定義することで、属性をグループへ追加する。

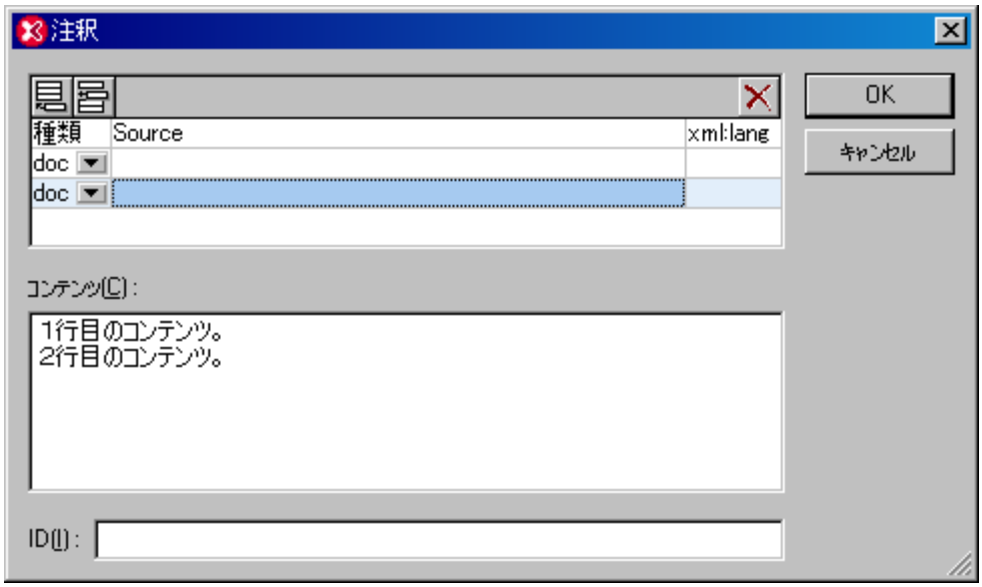
グローバル属性と属性グループが作成されると、要素や複合型の宣言にて参照することができるようになります。

表記(notation)

表記は常にグローバルなコンポーネントで、ローカルの表記というのは存在しません。表記のプロパティは表記の詳細入力ヘルパーにて指定されます。表記の名前はグローバル コンポーネントのリストから直接指定することができます。スキーマ内にあるすべての表記はコンポーネント入力ヘルパーに表示され、簡単に参照することができます。

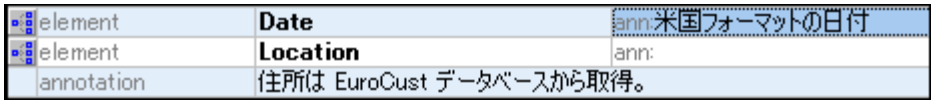
グローバル注釈

グローバル注釈はグローバルなコンポーネントで、いくつかのグローバルコンポーネントにて利用することのできるオプションの注釈とは異なります。注釈グローバルコンポーネントを右クリックして、コンテキスト メニューから「**すべての注釈データ**」を選択すると注釈ダイアログが表示され、グローバル注釈を編集することができます（*以下のスクリーンショットを参照*）。



各注釈には`id` 属性と複数のドキュメント や`appinfo` 子要素を含めることができます。ダイアログの左上にある「**追加**」または「**挿入**」*ボタンをクリック*し、対応するコンボ ボックスから`doc` または`app` を選択することで、ドキュメント または`appinfo` 要素を追加することができます。ダイアログ上部のペインにて`doc` または`app` を選択し、そのコンテンツを「*コンテンツ*」ペインにて入力します。コンテンツにて「**Enter**」キーを押下することで、コンテンツ内に改行を含める（そして複数行のコンテンツを作成する）ことができます。上のスクリーンショットではドキュメント 要素が選択され、2 行からなるコンテンツが確認できます。各 ドキュメント または`appinfo` 要素に対してオプションの`source` や`xml:lang` 属性を入力することもできます。

スキーマ概要では、グローバル注釈にある最初のドキュメント または`appinfo` 要素だけが表示され、編集することができます。その複数行のコンテンツが含まれているコンポーネントの場合、マウスオーバーすることにより、すべての行が複数行のポップアップ ボックスにて表示されます。その他 ドキュメント や`appinfo` 要素のコンテンツを表示するには、グローバル注釈の注釈ダイアログへアクセスしてください。



メモ　いくつかのグローバルコンポーネントで利用することのできるオプションの注釈も注釈ダイアログにて上記と同様の方法により編集することができます。

コメントと処理命令

コメントと処理命令は、スキーマ概要にあるグローバルコンポーネント内の任意の場所へ挿入することができます。コンテンツ モデルビューに追加することはできません。コメントや処理命令が単純型または複合型内部に含まれる場合、まとめて含まれるオブジェクトの最後に移動されます。このような移動を望まない場合、コメントの代わりに注釈を使用する必要があります。

4.4.3 コンテンツ モデル ビュー

コンテンツ モデルはコンポーネントの構造とコンテンツから構成され、以下のコンポーネントにコンテンツ モデルは含まれます:

- 複合型
- 要素
- モデル グループ
- デフォルト オープン コンテンツ

スキーマ概要[164]に示されたこれらのコンポーネントの左には、「**コンテンツ モデル ビューへの切り替え**」アイコンが表示されます。

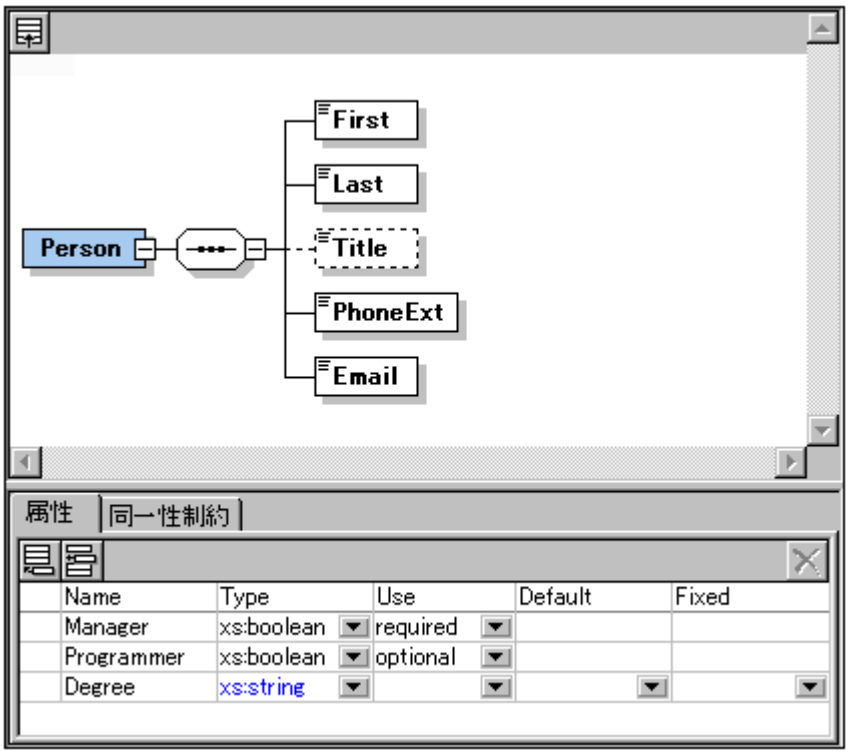
	コンテンツ モデル ビューへの切り替え: コンテンツ モデルが含まれるグローバル コンポーネントにて表示され、コンポーネントのコンテンツ モデルをコンテンツ モデル ビュー[176]にて表示します。
	グローバルの表示: コンテンツ モデル ビューにて表示され、クリックすることでスキーマ概要[164]が表示されます。

コンテンツ モデル ビューへの切り替えアイコンをクリックすることで、そのコンポーネントの**コンテンツ モデル ビュー**が表示されます(以下のスクリーンショットを参照)。その他にも、スキーマ概要[164]にて(i) コンポーネントを選択し、メニューから「**スキーマ設計 | 図の表示**」オプションを選択するか、(ii) コンポーネント入力ヘルパー[209]にあるコンポーネント名をダブルクリックすることでもコンテンツ モデル ビューが表示されます。同時に開くことのできるコンテンツ モデル ビューは、スキーマごとに1 つしかないという点に留意してください。コンテンツ モデルが表示されている場合、「Ctrl」キーを押下しながら現在表示されているコンテンツ モデル内にあるコンポーネントをダブルクリックすることで、そのコンポーネントのコンテンツ モデルへジャンプすることができます。

コンテンツ モデル ビュー全般の説明

コンテンツ モデル ビューではコンテンツ モデルがツリーとして表示されます(以下のスクリーンショットを参照)。メニューの「スキーマ設計 | 表示設定[1264]」により表示されるスキーマ表示構成ダイアログでは、ツリーの外観を設定することができます。

(C) 2015-2021 Altova GmbH



以下の点に留意してください：

- コンテンツ モデル ツリーにはコンポジッターとコンポーネントの2 種類あります。更に、属性、アサーション、同一性制約、オープン コンテンツをコンポーネントに接続されたボックス内に表示することもできます。
- ツリーの各階層は、コンポジッターにより接続されます。コンテンツ モデルに含めることのできる階層の数に制限はありません。
- オブジェクトを右クリックすることで表示されるコンテキスト メニューから、そのオブジェクトに対して相対的に他のオブジェクトを追加することができます。
- コンテンツ モデル内にあるコンポーネントは、ローカル コンポーネントとして定義またはグローバル コンポーネントを参照することができます。
- ドラッグ アンド ドロップ機能によりオブジェクトを移動することができます。
- キーボード ショートカットにより、オブジェクトをコピー（「Ctrl」 + C）または貼り付ける（「Ctrl」 + V）ことができます。
- オブジェクトのプロパティは詳細入力ヘルパーならびにAAIDC ペイン[195]にて編集することができます。
- コンポーネントの属性、アサーション、同一性制約はコンテンツ モデル ビュー下部にあるAAIDC ペイン[195]に表示されます。属性と同一性制約はAAIDC ペイン[195]の代わりにコンテンツ モデルの図に表示することもできます。この表示オプションはスキーマ表示構成ダイアログ[1264]にてセットすることができます。表示オプションはスキーマ デザイン ツール バーにある「図に表示」ボタンからもセットすることができます（以下のスクリーンショットを参照）。

- コンポーネントを右クリックして、コンテキスト メニューから「コンポーネントの並び替え」コマンドを選択することで、兄弟コンポーネントを並び替えることができます。ローカル名とコンポーネントの種類から並べ替えの優先順位を指定することができます。

これらの機能の詳細については、このセクションにあるサブセクションと、チュートリアルを参照ください。

「グローバルの表示」アイコンをクリックするか、メニューオプションの「スキーマ設計 | すべてのグローバルを表示」を選択することで、スキーマ概要[164]が表示されます。

4.4.3.1 コンテンツ モデル オブジェクト

コンテンツ モデルビューでは、図の中に表示されたオブジェクトを以下の3 種類の大まかなグループに分けることができます:

- コンポジッター[178]: (i) シーケンス、(i) 選択、(iii) すべて
- コンポーネント[179]: (i) 要素、(i) 複合型、(iii) モデル グループ、(iv) ワイルドカード
- その他[181]: (i) 属性、(ii) 属性グループ、(iii) アサーション、(iv) 制約、(v) オープン コンテンツ

これらのオブジェクトの外観をそれぞれ以下に示します。

コンポジッター

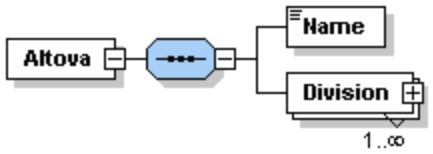
コンポジッターにより、子要素が出現する順序が定義されます。3つのオプションから選択することができます: シーケンス、選択、およびすべて。

以下の操作によりコンポジッターを挿入することができます:

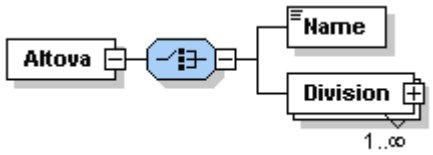
1. 子要素を追加する要素を右クリックします。
2. コンテキスト メニューから「子の追加 | シーケンス (選択 、すべて)」を選択します。

それぞれのコンポジッターは以下のような外観を持ちます。

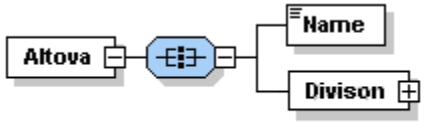
- シーケンス



- 選択



- すべて



コンポジッターの種類を変更するには、そのコンポジッターを右クリックして、コンテキスト メニューから「**「変更」「モデル | シーケンス」**(または「**選択**」または「**すべて**」)」を選択してください。コンポジッターを追加した後には、子要素やモデル グループを追加することになります。

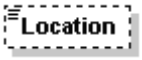
コンポネート

コンテンツ モデルで使用されるコンポーネントのリストを以下に示します。コンポーネントの種類とその構造に関するプロパティの詳細が、各コンポーネントの外観により示されます。

- 必須の要素 1 つ

  属性、*属性グループ*長方形によりそれが要素であることが示され、実線の境界線によりその要素が必須であることが示されます。番号の範囲が示されていないことから、その要素の出現回数が1 回である(例えば、`minOcc=1` and `maxOcc=1`) ことが理解できます。上の例では、`Country` が要素の名前となります。青色によりその要素が現在選択されていることが示されます(クリックすることでコンポーネントを選択することができます)。コンポーネントが選択されていない場合、要素は白色で表示されます。

- 省略可能な要素 1 つ

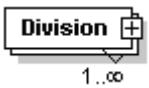
  

  属性、*属性グループ*長方形によりそれが要素であることが示され、点線の境界線によりその要素が省略可能であることが示されます。番号の範囲が示されていないことから、その要素の出現回数が1 回である(`minOcc=0` and `maxOcc=1`) ことが理解できます。上の例では、`Location` が要素の名前となります。
  メモ コンテキスト メニューにある「**省略可能**」 をクリックすることで、その要素がオプションとなります。

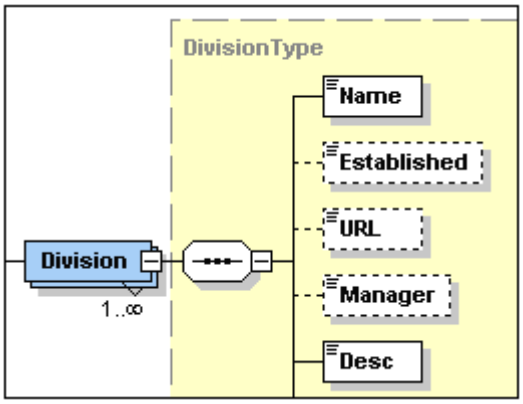
- 複数の必須要素

  属性、*属性グループ*長方形によりそれが要素であることが示され、実線の境界線によりその要素が必須であることが示されます。番号の1..5 は`minOcc=1` と`maxOcc=5` を表しています。上の例では`Alias` が要素の名前となります。

- 子要素を含む複数の必須要素

  属性、*属性グループ*長方形によりそれが要素であることが示され、実線の境界線によりその要素が必須であることが示されます。番号の1..∞は`minOcc=1` と`maxOcc=unbounded` を表しています。プラス記号により混合コンテンツ(少なくとも 1 つの子要素または属性が含まれる) を示しています。`Division` が要素の名前となります。
  メモ コンテキスト メニューの「**無制限**」 コマンドにより、`maxOcc` を`unbounded` にセットすることができます。
  要素の+ 記号をクリックすることで、子要素が表示されます。

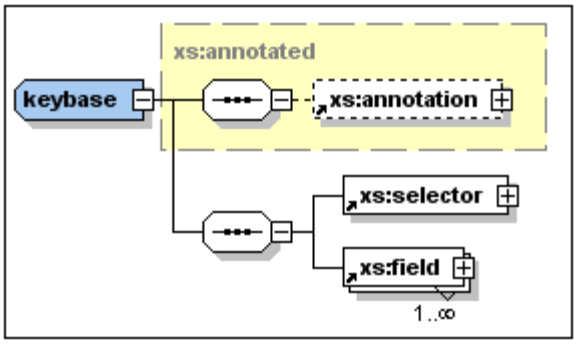


- グローバル要素を参照している要素

属性、*属性グループ*長方形の左下にある矢印により、要素がグローバル要素を参照していることが示されます。長方形によりそれが要素であることが示され、実線の境界線によりその要素が必須であることが示されます。番号の1..∞は`minOcc=1` と `maxOcc=unbounded` を表しています。プラス記号により混合コンテンツ（少なくとも 1 つの子要素または属性が含まれる）を示しています。`xs:field` が要素の名前となります。
メモ グローバル要素は単純型や複合型の定義から参照することができ、スキーマ内にある複数の場所で再利用することができます。グローバル要素は単純型や複合型の定義から参照することができ、スキーマ内にある複数の場所で再利用することができます。グローバル要素への参照は、(i) グローバル要素と同じ名前をローカルの要素名として入力するか、(ii) ローカル要素を右クリックしてコンテキスト メニューから「参照」を選択することで行うことができます。「Ctrl」キーを押下しながら要素をダブルクリックしてその要素が参照しているグローバル要素の定義を確認することができます。また要素を右クリックしてコンテキスト メニューから「定義へ移動」を選択することで確認することもできます。存在しない要素への参照が作成された場合、参照される定義がないということを表すため、要素名が赤色で示されます。

- 複合型

属性、*属性グループ*一部角の取れた長方形とプラス記号により、それが複合型であることが示されます。この例で示される複合型は`keybase` という名前で、このシンボル（角の取れた長方形とプラス記号）により、グローバルの複合型であることが示されます。グローバルの複合型はスキーマ概要で宣言され、そのコンテンツ モデルは通常コンテンツ モデル ビューで定義されます。グローバルな複合型は、(i) 要素のデータ型として使用することができるほか、(ii) 詳細入力ヘルパーにて要素や複合型への割り当てを行うことで別の複合型の基底型として使用することができます。



(C) 2015-2021 Altova GmbH

上に示される keybase 複合型は、xs:annotated を基底型とするようにスキーマ概要で宣言されています。基底型は灰色の破線を境界線とする長方形と、黄色の背景色により示されます。コンテンツ モデルビューでは子要素の xs:selector と xs:field が作成されています（xs:selector と xs:field 長方形の左下には小さな矢印が示されており、これらの要素が同名のグローバル要素を参照していることが理解できます）。

要素以下に子要素や属性をコンテンツ モデルビューにて作成することで、ローカルの複合型が定義されます。ローカルの複合型を表す特別なシンボルは存在しません。

メモ コンテンツ モデルの基底型は、破線のグレー色で表示される境界線と、黄色の背景色により表示されます。名前をダブルクリックすることで、基底型のコンテンツ モデルへ移動することができます。

- モデルグループ

属性、*属性グループ*プラス記号とともにすべての角が取れた長方形はモデル グループを表します。モデル グループにより要素宣言を定義して再利用することができます。
メモ モデル グループを（スキーマ概要にて）宣言するには名前を与える必要があります。その後、（コンテンツ モデルビューにて）要素宣言を含んだ子コンポジッターをモデル グループへ割り当てることになります。モデル グループは、子として挿入、または他のコンポーネントのコンテンツ モデル内に挿入またはお追加することにより使用されます。

- ワイルドカード

属性、*属性グループ*any という名前のついたすべての角が取れた長方形はワイルドカードを表します。
メモ ワイルドカードはスキーマ内部や他の名前空間にて指定されていない要素を表すためのプレースホルダとして使用されます。####other = 要素はスキーマで定義されているターゲット名前空間を除くすべての名前空間に属することができ、##any = 要素はあらゆる名前空間に属することができます。##targetNamespace = 要素はスキーマ内で定義されたターゲット名前空間へ属する必要があります。##local = 要素はどの名前空間へも属することができず、anyURI = 要素は指定された名前空間に属することになります。

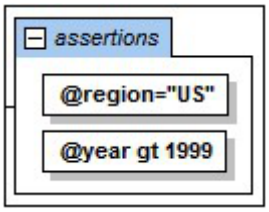
その他のオブジェクト

その他のオブジェクトは属性、属性グループ、アサーション、同一性制約、その他のコンテンツになります。

- 属性、属性グループ

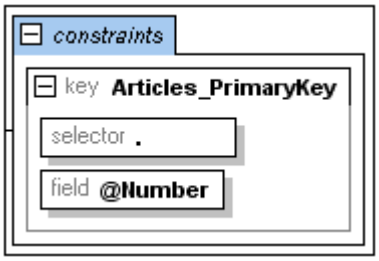
属性、*属性グループ*イタリック体の *'attributes'* という文字と、展開可能な長方形により表示されます。各属性を表す長方形は、(i) 属性が省略可能の場合は破線の境界線、(ii) 属性が必須の場合は実線の境界線で表示されます。属性グループと属性ワイルドカードも *'attributes'* の長方形内に表示されます。
メモ 属性は図の内部と詳細入力ヘルパーの両方で編集することができます。属性は、コンテンツ モデルの図内部またはコンテンツ モデルビュー下部にある AADC ペイン[195] 内部に表示されます。これら 2 つの表示方法は属性の表示 アイコンをクリックすることで切り替えることができます。要素内にある属性の順序を変更するには、属性をドラッグして、目的の場所で矢印が表示されている状態でドロップしてください。

- アサーション メッセージ コントロール



属性、*属性グループ*イタリック体の'assertions' という文字と、展開可能な長方形により表示されます。各アサーションは、ボックス内の長方形として示されます。
メモ アサーションは図の内部と詳細入力ヘルパーの両方で編集することができます。アサーションは、コンテンツ モデルの図内部または*コンテンツ モデル*ビュー下部にある*AADC ペイン*[195] 内部に表示されます。これら 2 つの表示方法はアサーションの表示 アイコンをクリックすることで切り替えることができます。要素内にあるアサーションの順序を変更するには、アサーションをドラッグして、目的の場所で矢印が表示されている状態でドロップしてください。

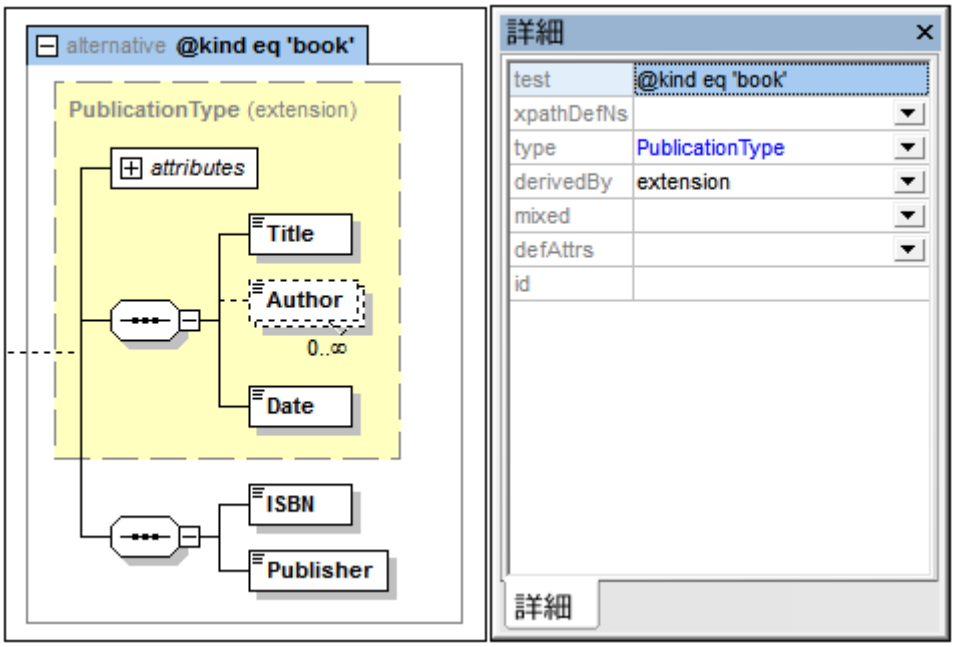
- 同一性制約



属性、*属性グループ*イタリック体の'constraints' という文字と、展開可能な長方形により表示されます。
メモ コンポーネントのコンテンツ モデルに表示された同一性制約により、`key` と `keyref`、そして `unique` 要素により定義された制約が表示されます。`ID` データ型により定義された同一性制約はコンテンツ モデルの図ではなく詳細入力ヘルパーに表示されます。同一性制約はコンテンツ モデルビュー、または*スキーマ概要*の同一性制約タブにて表示ならびに編集することができます。コンテンツ モデルビューでは、制約の表示 アイコンにより、制約ボックスの表示を切り替えることができます。

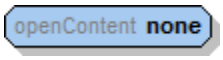
- 条件型の割り当て

属性、*属性グループ*`alternative` という名前のついた長方形には、評価される XPath 条件式が含まれます（*上のスクリーンショットを参照*）。`alternative` 要素の種類は詳細入力ヘルパーにより指定されます。複合型の場合、展開された `alternative` 要素のボックスに表示され、編集を行うことができます（*以下のスクリーンショットを参照*）。単純型は図に表示されませんが、詳細入力ヘルパーの単純型タブにて定義することができます。



メモ `alternative` 要素はXSD 1.1 にて新たに導入されました。XPath 条件式の結果がtrue となる場合、`alternative` 要素により指定された型が選択された型となります。`alternative` 兄弟の中で、最初にtrue と評価された `alternative` 要素が選択されるため、alternative 要素の順序には意味があります。`alternative` 要素のボックスをドラッグすることで、順序は変更することができます。詳細は条件型アサーション[189]を参照してください。

- デフォルト オープン コンテンツ、オープン コンテンツ



属性、*属性グループ*`defaultOpenContent` と`openContent` 要素は、`openContent` と`defOpenContent` というラベルとともにコンテンツ モデル ビューに示されます。ワイルドカード要素コンテンツは`any` ボックスにより示されます（*上のスクリーンショットを参照*）。
メモ `defaultOpenContent` と`openContent` 要素はXSD 1.1 にて新たに追加されたものです。デフォルト オープン コンテンツはグローバル コンポーネントで、スキーマ概要[164]にて作成されます。特定コンポーネントのコンテンツ モデルを表しているコンテンツ モデル ビューでは、デフォルト オープン コンテンツをそのコンポーネントで使用されるオープン コンテンツにより置き換え、スキーマのデフォルト オープン コンテンツをオーバーライドすることができます。コンポーネントの子としてオープン コンテンツを追加してください。デフォルト オープン コンテンツのボックスがオープン コンテンツのボックスに置き換えられます。コンテンツ モデル ビューでは、オープン コンテンツの`mode` 属性と、そのワイルドカード要素の名前空間を図のなかと詳細入力ヘルパーの両方で編集することができます。任意の複合型のコンテンツ モデル ビューから、スキーマ全体に対して適用されるデフォルト オープン コンテンツを修正することもできます。

メモ

- スキーマ表示構成ダイアログ[1264]にて指定された定義済みの詳細情報の表示は、定義済みの詳細情報ツール バー アイコンにより切り替えることができます。

- 属性、アサーション、同一性制約の表示アイコンをクリックすることにより、これらの情報をコンテンツ モデルの図の中で表示するか、または(コンテンツ モデル ビュー下部にある) AADC ペイン[195] 内部に表示するかを切り替えることができます。
- コンテンツ モデル ビューでは、「Ctrl」 キーを押下しながら図の中に表示されているグローバル コンポーネントの名前をダブルクリックすることで、そのグローバル コンポーネントのコンテンツ モデルへジャンプすることができます。
- 基底型の名前をダブルクリックすることでも、基底型のコンテンツ モデルへ移動することができます。

4.4.3.2 コンテンツ モデル ビューでの編集

コンテンツ モデル ビューで編集を行う方法は、以下のカテゴリーにより説明されます:

- コンテンツ モデル ビューの構成[184]
- 属性、アサーション、同一性制約[184]
- コンテンツ モデル ビュー アイコン[184]
- コンテキスト メニューからの操作[185]
- キーボード ショートカットとドラッグ アンド ドロップ[187]
- コンポーネント プロパティ[188]
- 注釈[188]
- コメントと処理命令[189]
- コンテンツ モデルのドキュメント化[189]

コンテンツ モデル ビューの構成

スキーマ表示構成ダイアログ「**スキーマ設計 | 表示設定**」では、スキーマ全体に対するコンテンツ モデル ビューの構成を指定することができます。構成のオプションに関する詳細については、ユーザー リファレンスにある「構成ビュー」[1264] セクションを参照ください。ここで行われる設定はスキーマ全体、そしてスキーマ ドキュメンテーションの出力と印刷の出力にも適用されることにご注意してください。

属性、アサーション、同一性制約

コンポーネントの属性、アサーション、そして同一性制約は、コンテンツ モデル ビューの下部に表示される AADC ペイン[195] に、またはコンテンツ モデル ビューそのもの、つまり図の中に直接表示されます。後者の表示オプションは、スキーマ表示構成ダイアログ[1264] で設定することができます。またスキーマ設計ツールバーにある**図の中に表示**ツールバー ボタンからセットすることができます(*下のスクリーンショット、およびアイコンのリストを参照してください*)。

属性、アサーション、同一性制約の挿入ならびに編集方法については、「属性、アサーション、同一性制約」[195] のセクションを参照ください。

コンテンツ モデル ビュー アイコン

	グローバルの表示: コンテンツ モデル ビュー内で使用が可能です。スキーマ概要[164] に切り替えます。
	定義済みの詳細の追加: コンテンツ モデル ビューが表示されている際にスキーマ設計ツールバーで有効になります。コンポーネント内にある定義済みの詳細情報の表示を切り替えます。
	属性を図に表示: コンテンツ モデル ビューが表示されている際にスキーマ設計ツールバーで有効になります。(オンに切り替え) 属性の表示を図または属性タブで行うか指定します。

	アサーションを図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツール バーで有効になります。(オンに切り替え) アサーションの表示を図またはアサーション タブで行うか指定します。
	制約を図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツール バーで有効になります。(オンに切り替え) 同一性制約の表示を図または同一性制約タブで行うか指定します。
	同一性制約の可視化: コンテンツ モデルビューが表示されている際にスキーマ設計ツール バーで有効になります。同一性制約の情報表示を切り替えます。

コンテキスト メニューからの操作

コンテンツ モデルビューにおける編集のいくつかは、コンテンツ モデルビューで右クリックすることで表示されるコンテキスト メニューから行われます(*以下のスクリーンショットを参照*)。コンテンツ モデルビューで指定された場所で許されている処理のコマンドだけが有効になります。操作は右クリックされたオブジェクトに対して相対的に行われます。例えば「子の追加」コマンドを選択すると、右クリックされたオブジェクトに対する子が追加されます。

コンテキスト メニューから利用可能な操作のリストを以下に示します:

- *子コンポジッターとコンポーネントの追加:* 「**子の追加**」コマンドは追加するコンポジッターまたはコンポーネントを選択することができるサブメニューを開きます。

- *コンポーネントとコンポジッターの挿入/追加:* 選択されたオブジェクトと同じ階層にコンポジッターまたはコンポーネントを挿入します。「**挿入**」コマンドにより選択されたオブジェクトの前に、「**追加**」コマンドにより最後の兄弟オブジェクトの後へ追加されます。

- *コンポジッターの変更:* コンポジッターを右クリックして、「**モデルの変更 | <新たな種類のコンポジッター>**」を選択してください。

- *グローバル コンポーネントの作成:* (i) 「**新規グローバル**」コマンドはコンテンツ モデル ビューのあらゆる場所からアクセスすることができます。作成される新規のグローバルコンポーネントを選択することのできるサブメニューが表示されます。(ii) オブジェクトがグローバルコンポーネントとして作成可能な場合、コンテキスト メニューで「**グローバル化**」コマンドが有効になります。このコマンドを選択することで、オブジェクトがグローバル コンポーネントとして作成されます。コンテンツモデルビュー内で新規に作成されたグローバル コンポーネントへの参照が含まれます。

- *出現定義の変更:* 「**省略可能**」ならびに「**無制限**」コマンドを組み合わせることで希望する出現設定を得ることができます: (i) *「省略可能」* = 0 または1、(ii) *「省略可能」* + *「無制限」* = 0 から無限、(iii) *「無制限」* = 1 から無限、(iv) *なし* = 1 のように指定できます。(メモ *「省略可能」* によりコンポーネントの `minOccurs` 属性が、*「無制限」* により `maxOccurs` 属性が影響を受けます)。

- *ローカルとグローバル 定義の切り替え:* ローカル要素と同じ名前のグローバル要素が存在する場合、「**参照**」コマンドによりグローバル要素への参照と、ローカルの定義を切り替えることができます。

- *別のコンテンツ モデルへジャンプ:* コンポーネントが自身のコンテンツモデルを持つ場合 (例えば、グローバルコンポーネントを参照する場合) 「**定義へ移動**」コマンドが有効化され、コンテンツモデルへ移動することを選択できるようになります。代わりに「Ctrl」キーを押下しながらコンポーネントをダブルクリックしてもジャンプすることができます。

- *コンポーネントの型定義に移動:* コンポーネントが型定義 (単純型または複合型), を持つ場合「**型定義へ移動**」コマンドを使用すると型定義に移動することができます。ビルトインの単純型の場合、単純型に関する情報が含まれたメッセージボックスが表示されます。

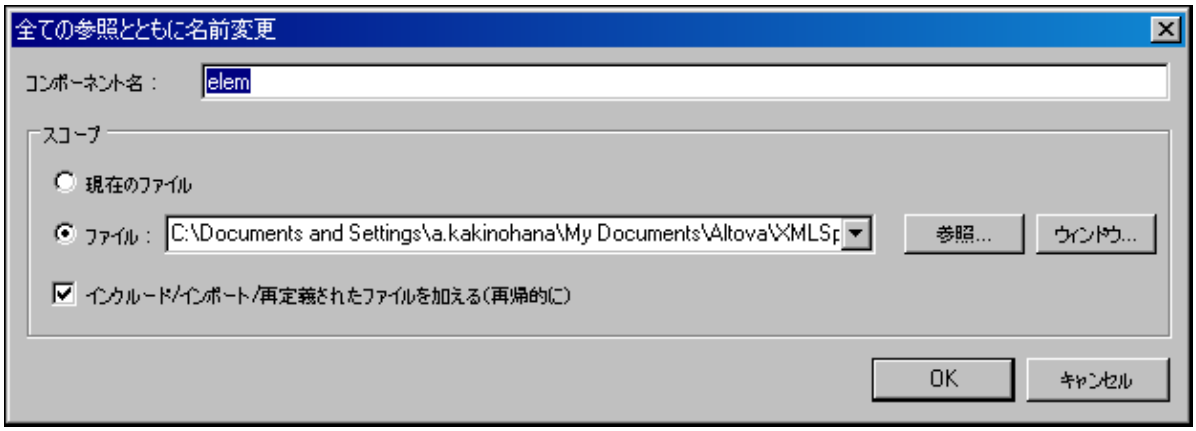
- *定義済みの詳細情報を編集:* 定義済みの詳細が (スキーマ設計ツールバーの**定義済みの詳細を追加**アイコンにて) 図の中に表示されるようセットされている場合、「**編集**」コマンドは定義済みの詳細を含むサブメニューを表示します。定義済みの詳細を選択することで、その値を図の中で編集することができます。

- *コンポーネントとコンポジッター注釈の作成:* 「**注釈の編集**」コマンドにより、コンポジッター/コンポーネントの下部に注釈を作成するための空間が作成されます (*以下のスクリーンショットを参照*)。この空間でお注釈の入力と編集を行うことができます。注釈が既に存在する場合、コメントをクリックすることで注釈テキストがハイライトされ編集することができるようになります。既存の注釈テキストをダブルクリックすることでも編集を行うことができます。

XML スキーマドキュメントでは、コンポジッターまたはコンポーネントの `注釈/ドキュメント` 要素内部に注釈が作成されます。*以下にあるドキュメンテーションに関する記述も参照ください。*

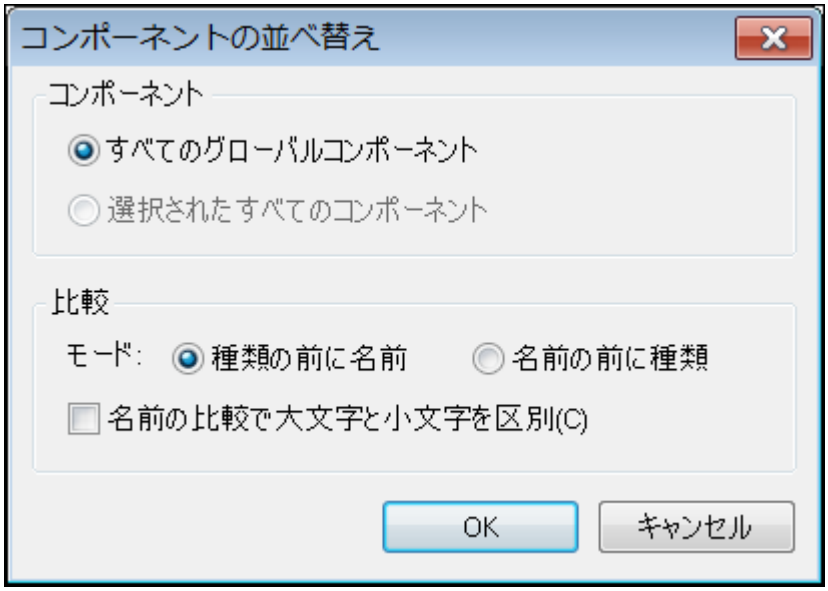
- *インスタンス ノードの XPath をコピー:* 「**インスタンス ノードの XPath をコピー**」コマンドは、グローバル要素ならびにグローバルな複合型にて定義された要素と属性に対して有効になります。このコマンドにより、選択されたノードを指定するための XPath 条件式がクリップボードへコピーされます。path 条件式は、コンテンツ モデル ビューに現在表示されているグローバル コンポーネントから開始されます。

- *コンポーネントの検索と名前変更:* The commands 「**すべての参照を検索**」ならびに「**すべての参照とともに名前変更**」コマンドはグローバル要素に対して有効になります。これらのコマンドは、アクティブなドキュメント、またはアクティブなドキュメントに関連するすべてのドキュメントにて選択されたコンポーネントの全出現を検索し、選択されたコンポーネントの全出現を名前変更します。



上のスクリーンショットでは、右クリックされたコンポーネントと、選択されたスコープ内におけるそのコンポーネントへの参照名が `Email` に置き換えられます。詳細については「グローバルの検索と名前変更」[425]を参照ください。

- *宣言と参照の並べ替え*: 「並べ替え」コマンドにより、選択されているすべてのコンポーネントまたは、選択されたコンポーネントの兄弟を並べ替えることができます。コンポーネントの並べ替えダイアログ(*下のスクリーンショット*)にて設定を行い、「OK」をクリックしてください。



「**Shift**」または「**Ctrl**」キーを押下しながらクリックすることで、複数のコンポーネントを選択することができます。コンポーネント名とその種類を並び替えのキーとして使用することができます。

メモ　コンポーネントをコピー、切り取り、削除、およびドラッグすることができます。一部の場合 `complexType` 制約のこのような属性は許可されない可能性があります。

キーボード ショートカットとドラッグ アンド ドロップ

キーボードの**Ctrl+c** ならびに**Ctrl+v** ショートカットを使用することで、コンテンツ モデル ビュー内にある要素のコピーと貼り付けをそれぞれ行うことができます。構造的な理由からコピーや貼り付けが行えない場合、その旨のメッセージが表示されます。

その他にも、ドラッグ アンド ドロップにより(i) オブジェクトを図の別の場所へ移動、(ii) 属性などのコンポーネントを入力ヘルパーから図の中へコピーすることができます。

コンポーネント プロパティ

コンポーネントのボックス内に定義済みの詳細が表示されるようにコンテンツ モデルビューが設定されている場合、図の中でこれら詳細情報を直接編集することができます。定義済みの詳細表示は、「**定義済みの詳細を追加**」ツールバー アイコンをクリックすることで切り替えることができます（*上記のアイコン リストを参照ください*）。

表示設定がなされている場合、コンポーネントのプロパティを詳細入力ヘルパー[213]で変更することで、その変更がプレースホルダ フィールドに反映されます。

注釈

XML スキーマの注釈は `annotation` 要素内部に含まれます。注釈には 2 つの種類があり、それぞれ `annotation` の子要素に含まれます:

- `documentation` 子要素: スキーマの編集者にとって有用な情報が含まれます。
- `appinfo` 子要素: 処理を行うアプリケーションが使用することのできるスクリプトや情報を挿入することができます。

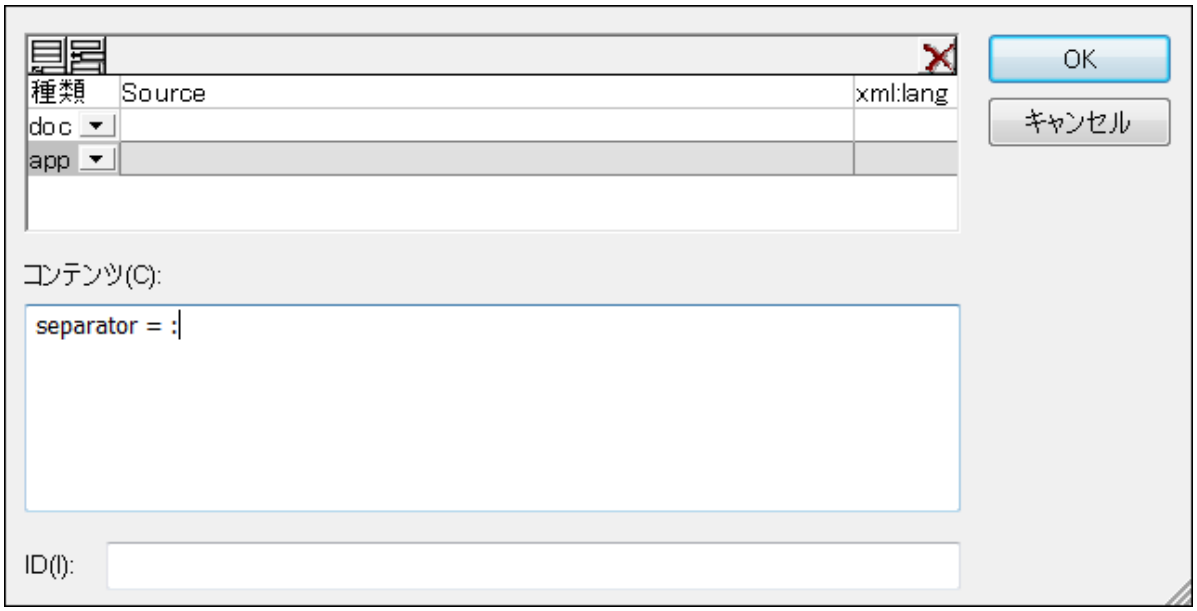
両方の子要素を含んだannotation 要素の例が以下に示されています。

```
<xs:element name="session_date" type="xs:dateTime" nillable="true">
   <xs:annotation>
      <xs:documentation>Date and time when interview was held</xs:documentation>
      <xs:appinfo source="http://www.altova.com/datehandlers/interviews">separator =
:</xs:appinfo>
   </xs:annotation>
</xs:element>
```

コンテンツ モデルビューでは、個別のコンポジターとコンポーネントのために注釈を以下のように作成することができます。

1. コンポジターまたはコンポーネントを右クリックします。

2. コンテキスト メニューから、「**すべての注釈データ**」を選択します。注釈ダイアログ ボックスが表示されます（*以下のスクリーンショットを参照*）。その要素に対して注釈（`documentation` または `appinfo`）が既に存在する場合、ダイアログの対応する行に、それらの情報が表示されます。



3. appinfo 要素を作成するには、ダイアログの左上にある追加または挿入アイコンをクリックしてください。
4. 新たに作成された行の*種類*フィールドから app オプションを選択してください。
5. ダイアログのコンテンツ ペインに、アプリケーションにより処理されるスクリプトや情報を記述します。
6. オプションとして、処理アプリケーションが参照することのできるソース URI をソース フィールドに入力することもできます。

コメントと処理命令

XML スキーマドキュメントが XMLSpy 内部にロードされるまたは変更されると、単純型ならびに複合型内にあるコメントと処理命令が、それらを含んでいるオブジェクトの最後部にまとめて移動されます。このような移動を望まない場合、コメントの代わりに注釈を使用する必要があります。

コンテンツ モデルのドキュメント化

スキーマに関する詳細なドキュメンテーション[1259]を HTML ならびに MS W ord フォーマットで生成することができます。各グローバル コンポーネントに関する詳細なドキュメンテーションが生成され、目次ページに表示されたグローバル コンポーネントのリストから、個々のコンポーネントへ素早くアクセスすることができます。更に、関連する要素（子要素や複合型など）もハイパーリンクにより参照され、要素から要素へのナビゲーションを簡単に行うことができます。スキーマのドキュメンテーションを生成するには、メニュー コマンドから「**スキーマ設計 | ドキュメンテーションの生成**」を選択してください。

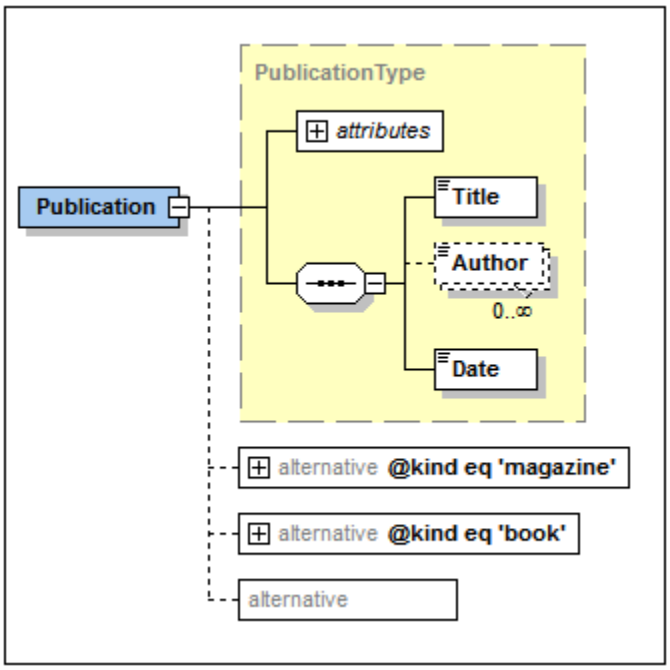
4.4.3.3 条件型割り当て

条件型割り当ては XSD 1.1 に新たに追加された機能で、要素に含まれる属性値や属性が存在するかにより XML ドキュメント内にある要素の型が決定されます。以下の様な XML ドキュメントを例に考えてみましょう

```
<publication kind="magazine">
      ...
</publication>
```

スキーマでは、インスタンス要素の kind 属性値により、複数の publictaion 要素の型を指定することができます。スキーマでは、XSD 1.1 にて新たに追加された alternative 要素が使用されます。各 alternative 要素では、それぞれ異なる型が指定されます。

以下のスクリーンショットでは、test 属性を含む alternative 子要素（@kind eq 'magazine' と @kind eq 'book'）とともに Publication 要素が宣言されています。3 番目の alternative 要素には test 属性が含まれておらず、（詳細入力ヘルパーで割り当てられており、図の中には表示されていない）xs:error の単純型が割り当てられており、必要なときに XML 検証エラーが返されます。



上の宣言を以下に示します:

```
<xs:complexType name="PublicationType">
	<xs:sequence>
		<xs:element name="Title" type="xs:string"/>
		<xs:element name="Author" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
		<xs:element name="Date" type="xs:gYear"/>
	</xs:sequence>
	<xs:attribute name="kind" type="xs:string"/>
</xs:complexType>

<xs:complexType name="MagazineType">
	<xs:complexContent>
		<xs:restriction base="PublicationType">
		<xs:sequence>
				<xs:element name="Title" type="xs:string"/>
				<xs:element name="Date" type="xs:gYear"/>
			</xs:sequence>
		</xs:restriction>
	</xs:complexContent>
</xs:complexType>

<xs:element name="Publication" type="PublicationType">
	<xs:alternative test="@kind eq 'magazine'" type="MagazineType"/>
	<xs:alternative test="@kind eq 'book'">
		<xs:complexType>
			<xs:complexContent>
				<xs:extension base="PublicationType">
					<xs:sequence>
```

```
                                    <xs:element name="ISBN" type="xs:string"/>
                                    <xs:element name="Publisher" type="xs:string"/>
                                </xs:sequence>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:alternative>
                <xs:alternative type="xs:error"/>
</xs:element>
```

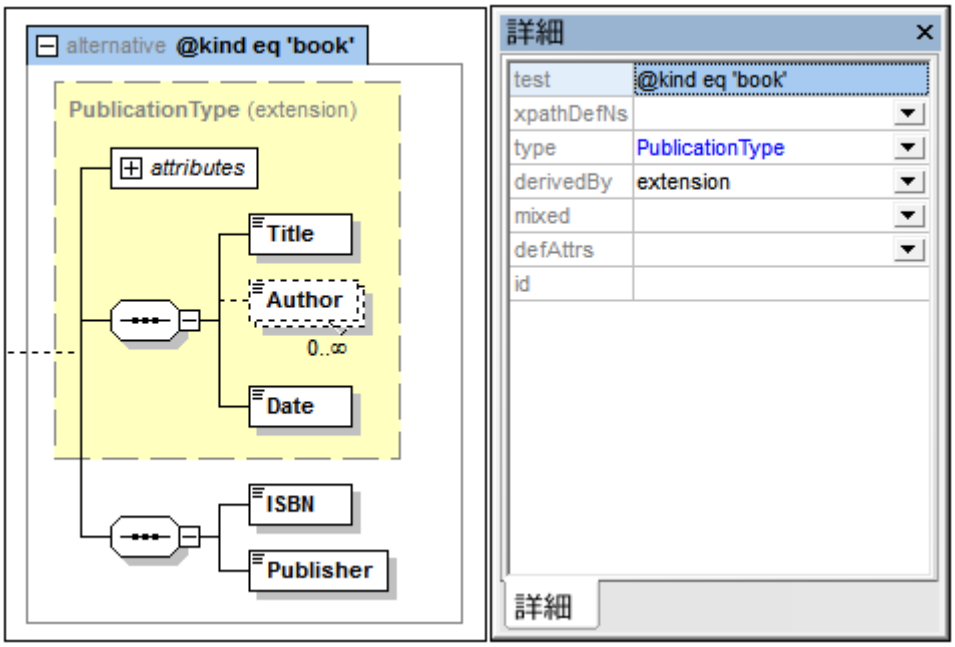
以下の点に注目してください：

- 複数ある`alternative`要素の中で、最初に`true`と評価された`alternative`要素が選択されるため、`alternative`要素の順序は重要な意味を持ちます。コンテンツ モデル ビューでは、`alternative`要素のボックスをドラッグすることで、その順序を変更することができます。
- `Publication`要素には型（`PublicationType`）が含まれていることに注目してください。どの`alternative`要素も使用されない場合、この型がデフォルトの型として使用されます。上の例では、最初の2 つの`alternative`が`false`を返した場合、`xs:error`型の`alternative`要素が使用されます。
- どの`alternative`要素条件も満たされず、デフォルトの型も定義されていない場合、要素は`anyType`型に割り当てられます。これにより、XML コンテンツの整形式を保つことができる場合もあります。
- `alternative`要素と単純型の`xs:erroor`はXSD 1.1 に新たに追加された要素です。

コンテンツ モデル ビューでの編集

要素のコンテキスト メニューから子として`alternative`型を要素宣言に追加することもできます（以下のスクリーンショットにあるコンテンツ モデルを参照）。

`alternative`要素の型は詳細入力ヘルパーで指定されます。複合型の場合、それが`alternative`要素の展開された長方形にて表示され、編集することができます（以下のスクリーンショットを参照）。単純型は図に表示されませんが、詳細入力ヘルパーの単純型タブにて定義することができます。



メモ　詳細入力ヘルパーでは、XPath 条件式の名前空間を`xpathDefaultNamespace`属性にて指定することができます。XPath デフォルト名前空間に関する詳細については、以下の記述を参照ください。

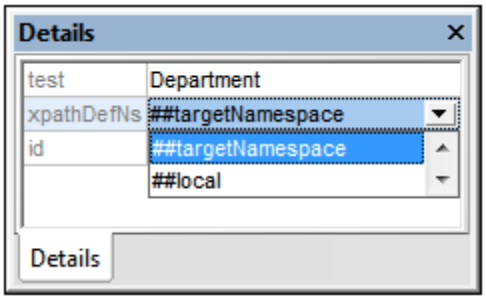
xpathDefaultNamespace の使用

XML スキーマドキュメントで宣言されたデフォルト名前空間は、そのXML スキーマドキュメントのデフォルト名前空間です。スキーマドキュメント内の接頭辞のない要素名に対して適用されますが、スキーマドキュメント内の接頭辞のないXPath 条件式に対しては適用されません。

XSD 1.1 にて新たに追加された xpathDefaultNamespace 属性（ XSD 1.1 の新規機能）は、XPath 条件式にて接頭辞のない要素の名前空間を指定するために使用されます。XPath デフォルト名前空間は、それらが宣言されたXML スキーマ要素上でスコープされます。xpathDefaultNamespace 属性は、以下のXML Schema 1.1 要素内に記述することができます:

- xs:schema
- xs:assert とxs:assertion
- xs:alternative
- xs:selector とxs:field（同一性制約にて）

XSD 1.1 モードでは、xs:schema 上のxpathDefaultNamespace がスキーマ設定ダイアログ（**「スキーマ設計 | スキーマ設定」**）にてセットされます。上に示されるその他の要素の場合、**xpathDefaultNamespace 属性が詳細入力ヘルパーにてセットされます（以下のスクリーンショットを参照）**。xs:schema 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、xpathDefaultNamespace 属性が許されている属性でオーバーライドすることができます（*上のリストを参照ください*）。



Xs:schema 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、xpathDefaultNamespace 属性が許されている属性でオーバーライドすることができます（*上のリストを参照ください*）。

実際の名前空間を含める代わりにxpathDefaultNamespace 属性には、以下のうちのどれかを含めることができます:

- ##targetNamespace: スキーマのターゲット名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- ##defaultNamespace: スキーマの名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- ##local: XPath デフォルト名前空間は指定されません。

ドキュメントに対してXPath デフォルト名前空間が宣言されなかった場合、XPath 条件式における接頭辞なしの要素は名前空間を持たない要素として扱われます。

メモ XPath デフォルト名前空間宣言は属性に対して適用されません。

(C) 2015-2021 Altova GmbH

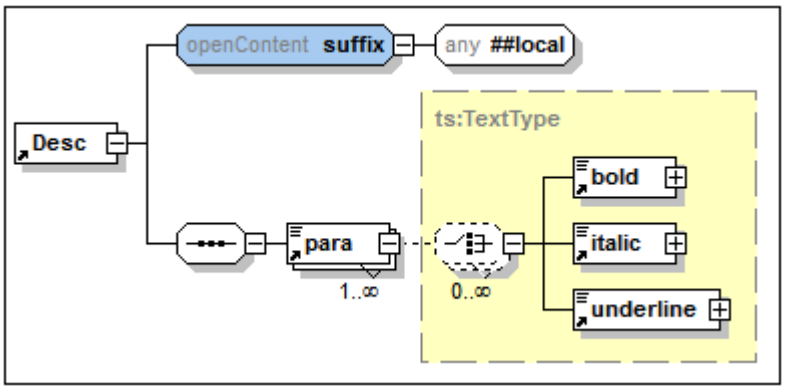
4.4.3.4 オープン コンテンツ モデル

オープン コンテンツはXSD 1.1 に新たに追加された機能で、複合型にて宣言され、(i) コンテンツ モデルにて定義された要素の間に、または(ii) コンテンツ モデルで定義された要素の後に記述されるあらゆる種類の要素 (複合型のコンテンツ モデルでは定義されていない要素) を許可するために使用されます。

openContent 要素は複合型の子要素で、複合型のコンテンツ モデル前に一度出現します (以下のスクリーンショットを参照)。

モード

openContent 要素には必須のmode 属性が含まれており、値にはinterleave、suffix、またはnone のいずれかが使用されます (以下のスクリーンショットを参照)。デフォルトではinterleave が使用されます。



これら値が持つ意味を以下に記します:

- mode="interleave" またはmode="suffix" となっている場合、最低ならびに最大出現数が指定されていないワイルドカード要素コンテンツ (xs:any) が必要になります。これにより、任意の数の未定義要素 (ワイルドカード) が許されることになります。
- モードがinterleave の場合、コンテンツ モデルにて定義された個々の要素の前または後に任意の数の未定義要素が出現できるようになります。
- モードがsuffix の場合、コンテンツ モデルにて定義されている最後の要素の後に任意の数の未定義要素が出現できるようになります。
- モードがnone の場合、未定義の要素 (xs:any 子要素) の出現は許されず、コンテンツ モデルもオープンとはなりません。none 値は、スキーマ全体をスコープにするdefaultOpenContent 169 要素をオーバーライドするために使用されます。

コンテンツ モデル ビューでは、複合型の子要素として (コンテキスト メニューの「子の追加」から) オープン コンテンツが追加されます。図の中にあるopenContent ボックスをダブルクリックして値 (interleave、suffix、またはnone) を選択するか、詳細入力ヘルパーからモードの値を選択してください。

ワイルドカード (xs:any) プロパティ

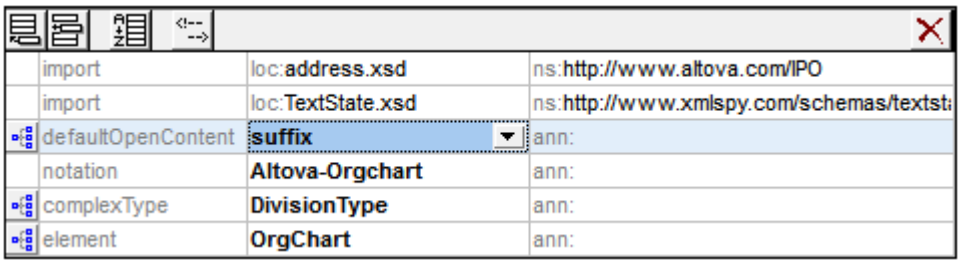
ワイルドカード プロパティは、ワイルドカードの詳細入力ヘルパーにて指定されます。図の中でワイルドカードを選択し、詳細入力ヘルパーにてプロパティ値を入力してください。

デフォルト オープン コンテンツ

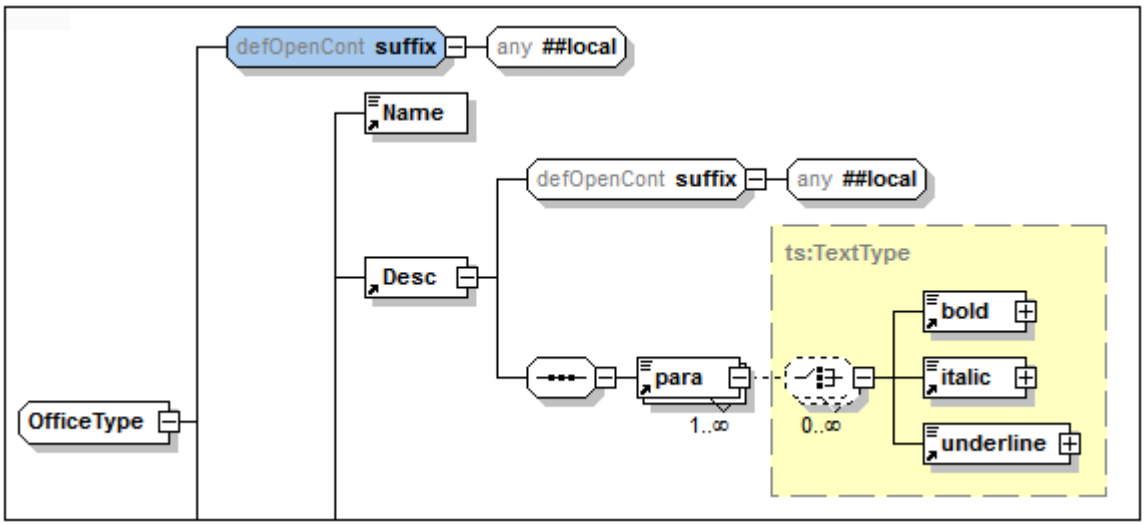
defaultOpenContent 要素はXSD 1.1 にて新たに追加された機能で、混合または子要素だけが許されている複合型に追加することのできる未定義の要素を指定するために使用されます。同じくXSD 1.1 で新たに追加されたopenContent 要素に似ていますが、

openContent 要素が単一の複合型に対して適用されるのに対して、defaultOpenContent 要素はスキーマ内にあるすべての複合型に対して適用されます。

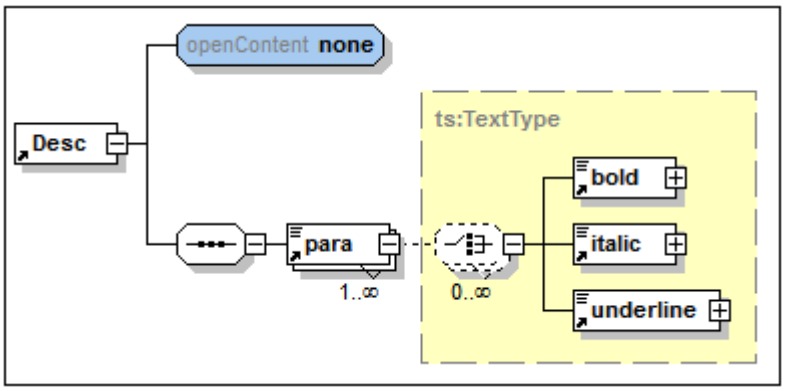
defaultOpenContent 要素はグローバル要素[169]で、ドキュメント内（インクルード、インポート、再定義の後、そしてコンテンツ定義の前）に一度だけ出現することができます（以下のスクリーンショットを参照）。要素には interleave または suffix という値を含めることのできる mode 属性が含まれており、デフォルトでは interleave 値が使用されます。



defaultOpenContent 要素には openContent 要素の定義と同様に、コンテンツ モデル ビューで編集することのできるコンテンツ モデルが含まれています（上記参照）。一旦 defaultOpenContent 要素が宣言されると、スキーマ内にあるすべての複合型に対して自動的に適用され、各複合型のコンテンツ モデルに表示されます。以下のスクリーンショットでは、defaultOpenContent が OfficeType と Desc という複合型に対して自動的に適用されているのが確認できます。

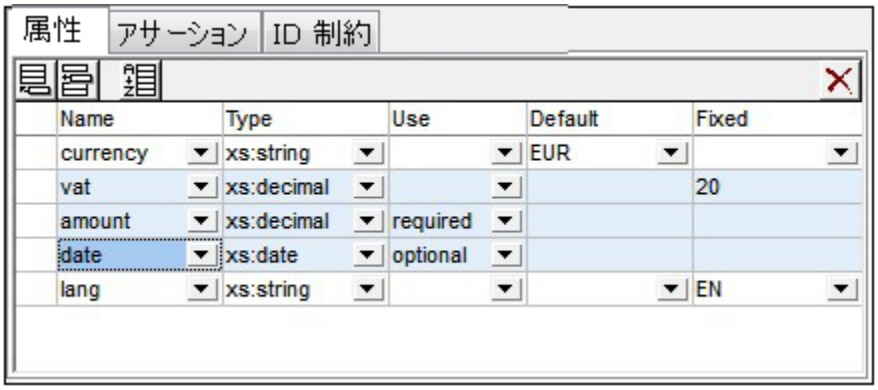


特定の複合型に対して defaultOpenContent 要素をオーバーライドするには、openContent 子要素をその複合型へ追加してください。以下のスクリーンショットでは、defaultOpenContent が適用された Desc 要素以下に openContent 子要素が定義されており、defaultOpenContent 要素がオーバーライドされています。



4.4.4 属性、アサーション、同一性制約

属性/アサーション/同一性制約（AAIDC）ペインは、スキーマ概要メイン ペインならびにコンテンツ モデルビューの下部に表示されます（以下のスクリーンショットを参照）。ペインとそのタブは固定されていますが、コンテンツ モデルビューでは各タブの表示内容を（AAIDC タブではなく）コンテンツ モデルビューに表示して、図の中で直接編集することができます。3つのタブの内容を図の中に表示するように指定すると、AAIDC ペインが表示されなくなります。



AAIDC ペインと図の中における表示は、スキーマ表示設定ダイアログ[1264]（「**スキーマ設計 | 表示設定 | 要素タブ**」）または（以下に示される）スキーマ設計ツール バーのアイコンから切り替えることができます。

	属性を図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツール バーで有効になります。属性の表示を図または属性タブで行うか指定します。
	アサーションを図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツール バーで有効になります。アサーションの表示を図またはアサーション タブで行うか指定します。.
	制約を図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツール バーで有効になります。同一性制約の表示を図または同一性制約タブで行うか指定します。.

タブの使用

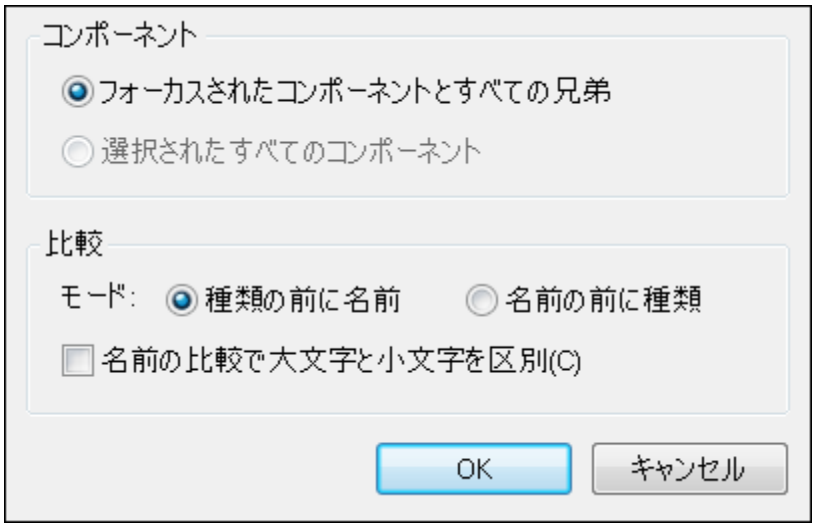
AAIDC ペイン内部のタブは、スキーマ概要またはコンテンツ モデル ビューにて選択されているコンポーネントの種類に対応しているタブが有効になります。例えば複合型に対して属性を追加することはできるため、メイン ペインにて複合型が選択されている時に属性タブが有効になります（タブが有効になるとは、そのタブの中でなんらかの操作ができることを意味します）。

各タブの使用方法については、このセクション以下にあるサブセクションを参照ください：

- 属性、属性グループ、属性ワイルドカード 196
- アサーション 199
- 同一性制約 202

属性と同一性制約の並べ替え

各タブのツール バーにある並べ替えアイコンをクリックすることで、属性と同一性制約の並べ替えを行うことができます。コンポーネントの並べ替えダイアログでは、選択されたコンポーネントとその兄弟、または選択されたコンポーネントだけを並べ替えの対象とするかを選択することができます。例えば上のスクリーンショットでは、3 つの属性が選択（ハイライト）されています。「Shift」 キーを押下しながらクリックすることで範囲の選択を、「Ctrl」 キーを押下しながらクリックすることで、コンポーネントを選択対象に加えることができます。



コンポーネントのセットを選択した後には最初にアルファベット順で並べ替えを行い、その次に種類による並べ替えを行うのか（種類の前に名前）、またはその逆（名前の前に種類）といった、並べ替えの優先順位を選択することができます。並べ替えの結果はスキーマドキュメントのテキストへ直ちに反映されます。

4.4.4.1 属性、属性グループ、属性ワイルドカード

属性/アサーション/同一性制約（AAIDC）ペインの属性タブでは、以下の操作を行うことができます:

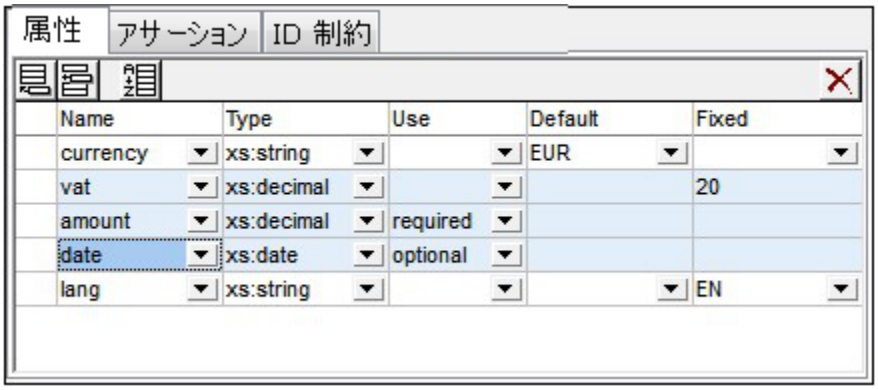
- 選択された複合型にて、ローカルの属性を宣言 197
- 選択された複合型で使用する属性グループの参照 198
- 選択された複合型にて属性ワイルドカードを定義 198

メモ　AAIDC ペインではなく、図の中に属性を表示するオプションを選択した場合（「**スキーマ設計 | 表示設定**」）、属性、属性グループ参照、属性ワイルドカードのプロパティを、図の中、そして詳細入力ヘルパーにて編集することができます。

	属性を図に表示: コンテンツモデルビューが表示されている際にスキーマ設計ツールバーで有効になります。属性の表示を図または属性タブで行うか指定します。

属性

属性/アサーション/同一性制約（AAIDC）ペインの属性タブでは、要素や複合型のローカル属性、または属性グループを構成する属性を宣言することができます（以下のスクリーンショットを参照）。



以下の操作により属性を作成することができます:

1. スキーマ概要にて複合型または属性を作成するための属性グループを選択します。
2. 属性タブの左上にある「**追加**」または「**挿入**」アイコンをクリックして、「**属性**」を選択します。
3. 属性に対して作成された行にて属性の詳細（Name、Type、Use、Default、Fixed の値）を入力します。Name プロパティは必須の項目で、`use` のデフォルト値はオプションです。データ型（Type）とDefault/Fixed 値はオプションです。

メモ　属性グループへの属性追加はスキーマ概要[159]でのみ行うことができますが、複合型への属性追加はスキーマ概要[159]とコンテンツモデルビュー[159]の両方で行うことができます。

メモ　属性が型定義を持つ場合、コンテキストメニューコマンド「**型の定義**」をクリックすると型定義に移動することができます。この場合、ビルトインの単純型の場合、単純型の情報を含むメッセージボックスが表示されます。

デフォルト値と固定値

属性宣言で宣言されたデフォルト値（Default）または固定値（Fixed）は、インスタンスドキュメント内にその属性が存在しない場合に（インスタンスドキュメントに対して）適用されます。指定できるのはデフォルト値と固定値の一方で、両方を指定することはできません（上のスクリーンショットを参照）。デフォルト値または固定値は属性のデータ型に対して妥当な値である必要があります。`use` に`required` がセットされている場合、デフォルト値も固定値も定義することはできません。

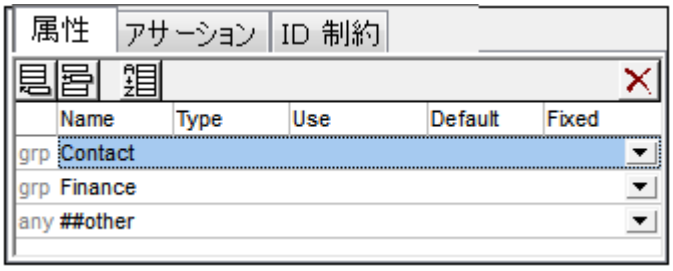
以下の点に留意してください:

- *デフォルトの値*: 属性が存在しない時だけデフォルト値は挿入されます。属性が存在しており、値が与えられている場合、デフォルト値は挿入されません。インスタンスドキュメント内の値が妥当ではない場合、エラーが返されます。
- *固定値*: 固定値は属性が存在しない場合だけではなく、インスタンスドキュメントの値が属性の宣言にて指定された固定値と異なる場合にも挿入されます。

メモ デフォルト値と固定値はローカルならびにグローバル属性に対しても指定することができます。ローカル属性の場合はAADC ペインの属性タブと詳細入力ヘルパーの両方で定義することができ、グローバル属性の場合、詳細入力ヘルパーにて定義することができます。

属性グループと参照

グローバルの属性グループが宣言されると、この属性グループに対する参照を複合型の定義内で追加することができます。スキーマ概要またはコンテンツ モデルビューにて複合型を選択し、AADC ペインの属性タブ左上にある「追加」または「挿入」アイコンをクリックして、「属性グループ」を選択してください。作成された属性グループ行にて参照される属性グループの名前を入力してください（以下のスクリーンショットには2 つの属性グループ参照が含まれています）。複数の属性グループを追加することができます。

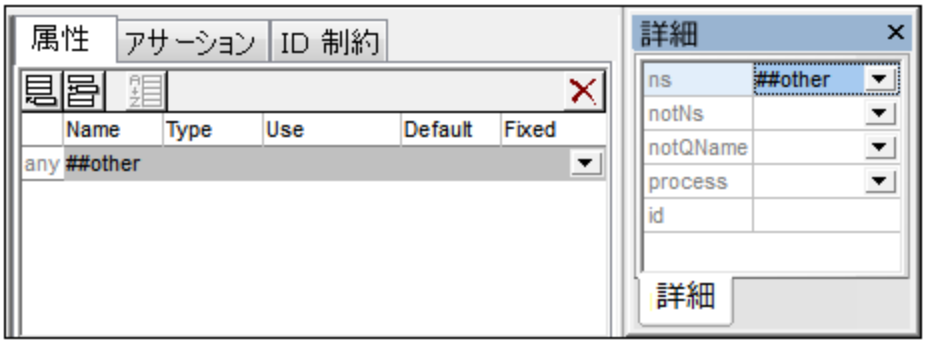


属性タブにて属性グループが選択されると、そのプロパティを詳細入力ヘルパーにて編集することができるようになります。

属性ワイルドカード: anyAttribute

属性ワイルドカードは、任意の属性を要素上で使用するのを許可するため複合型へ追加されます。属性ワイルドカードは単一の `anyAttribute` 要素により定義され、インスタンス ドキュメント内にある要素にて指定された名前空間から得られる任意の数の属性が出現するのを許可します。

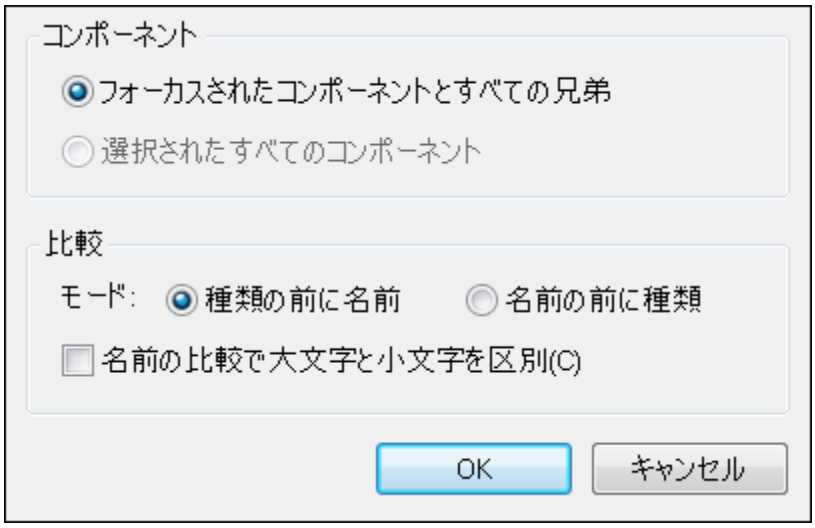
スキーマ概要またはコンテンツ モデルビューにて複合型コンポーネントを選択し、AADC ペインの属性タブ左上にある追加または挿入アイコンをクリックし、「任意属性」を選択することで属性ワイルドカードは作成されます。属性ワイルドカード `anyAttribute` に対する行が作成されます（以下のスクリーンショットを参照）。



属性タブでは、`anyAttribute` の `namespace` プロパティをセットすることができます。その他のプロパティは、属性タブにて属性ワイルドカードが選択されている状態で詳細入力ヘルパーにてセットすることができます（上のスクリーンショットを参照）。`notNamespace` ならびに `notQName` プロパティはXSD 1.1 の機能[161]で、XSD 1.0 モード[161]では利用できない点に注意してください。

属性と属性グループの並べ替え

タブのツール バーにある「並べ替え」アイコンをクリックすることで、属性タブに表示されている属性や属性グループを並べ替えることができます。コンポーネントの並べ替えダイアログでは、単一のコンポーネントとその兄弟コンポーネント、またはお選択されているコンポーネントのセットを並べ替えの対象とすることができます（以下のスクリーンショットを参照）。「Shift」または「Ctrl」キーを押下しながらコンポーネントをクリックすることで、複数のコンポーネントを選択することができます。



並べ替えを行う範囲を選択すると、属性すべてと属性グループをアルファベット順に並べ替え（種類の前に名前）、または属性と属性グループとアルファベット順で別々に並べ替えることができます（名前の前に種類）。

並べ替えの順序はスキーマドキュメントのテキストに直ちに反映されます。

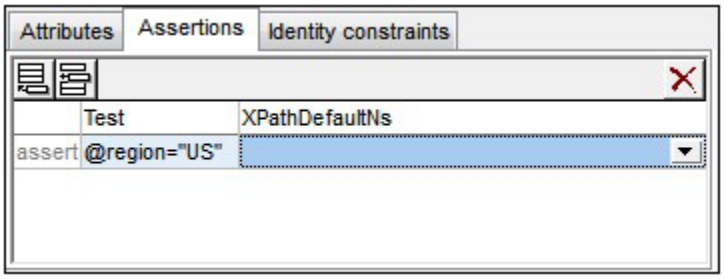
メモ　属性ワイルドカードは常に複合型宣言の末尾に出現する必要があり、複合型の定義に対して属性ワイルドカードはそれぞれ1つしか許されていないため、並べ替えの対象に含まれません。

4.4.4.2 アサーション

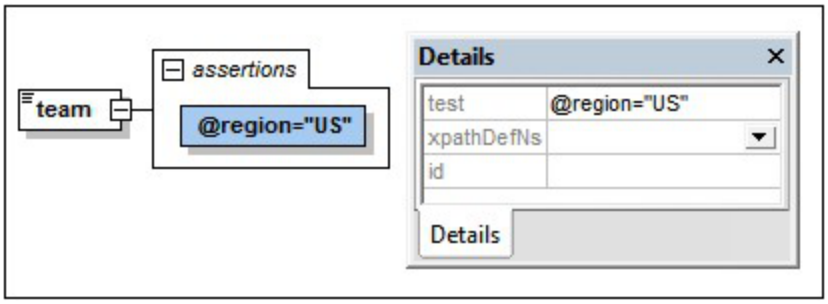
このセクションでは**複合型で使われるアサーション**について説明します。このようなアサーションは `xs:assert` 要素にて定義され、複合型の妥当性制約に使用されます（アサーションには `xs:assertion` 要素にて定義され、単純型の[ファセット入力ヘルパー](#)[215]にて作成ならびに編集されるアサーションも存在します）。

スキーマビューでは、以下の場所で複合型のアサーションを作成し、編集することができます:

- *スキーマ概要にて*: 属性/アサーション/同一性制約（AAIDC）ペインのアサーション タブにて（以下のスクリーンショットを参照）。



- *コンテンツモデルビューにて*: アサーションは図の中に表示されたアサーション タブにて編集することができます（以下のスクリーンショットを参照）。図の表示オプションを選択するには、スキーマ デザイン ツール バーにある「アサーションを図に表示アイコンをクリックしてください。複合型や複合型の要素にあるアサーションのボックスを選択し、詳細入力ヘルパーにてアサーションの定義を編集してください。



	図内でアサーションを表示する: コンテンツモデルビューで有効化されます。図とアサーションタブ間のアサーションの表示を切り替えます。

メモ アサーションはXSD 1.1 に搭載されている機能で、アサーションの編集機能は[XSD 1.1 モード](#) 161 でのみ有効になります。

アサーションのスコープ

アサーションの制約を定義するために使用されるXPath 条件式は、定義されている複合型のスコープ内にある必要があります。XPath 条件式が特定のノードへアクセスする必要がある場合、そのノードの祖先ノードにてアサーションを定義する必要があります。

アサーションの追加と削除

複合型には複数のアサーションを含めることができます。インスタンスドキュメント内にある要素に対してすべてのアサーションのXPath 条件式が`true` と評価されることで、ドキュメントも妥当なものと評価されます。以下の操作により、複合型に対してアサーションを追加することができます:

- **スキーマ概要にて**: 複合型を選択し、AADIC ペインにあるアサーション タブの左上にある**追加**または**挿入**アイコンをクリックします(上のスクリーンショットを参照)。追加することのできるアサーションの数に制限はありません。アサーションを選択し、タブの右上にある**削除**アイコンをクリックすることで、そのアサーションを削除することができます。
- **コンテンツ モデル ビュー**にて(上のスクリーンショットを参照): 複合型を右クリックして、「**子の追加 | アサーション**」を選択します。図に表示されている既存のアサーションを右クリックして、コンテキスト メニューから「**追加 | アサーション**」または「**挿入 | アサーション**」を選択することもできます。追加することのできるアサーションの数に制限はありません。アサーションを選択して、「**Delete**」キーを押下することで、そのアサーションを削除することができます。

アサーションのXPath 条件式を定義する

複合型アサーションのXPath 条件式により、インスタンスドキュメント内にある複合型要素に対して適用される妥当性制約が定義されます。例えば上のスクリーンショットでは、複合型要素の`team` とアサーションのXPath 条件式 `@region="US"` が定義されています。XML スキーマドキュメントで、アサーションは以下のように記述されます:

```
<xs:assert test='@region="US"'/>
```

アサーションにより、インスタンスドキュメント内にある`team` 要素には、`US` という値を持った`region` 属性を含めなければならなくなります。この条件が満たされない場合、ドキュメントの検証に失敗します。

以下の点に留意してください:

- XPath 条件式はXPath 2.0 に従ったかたちで記述される必要があります。
- XPath 条件式で評価されるノードは、アサーションのスコープ内にある必要があります(上記を参照ください)。
- 条件式が`boolean` の`true/false` を評価しない場合、返された値が`boolean` 値に変換されます。空でないシーケンスは`true` に変換され、空のシーケンスは`false` に変換されます。

- 条件式における構文エラーは赤色で示されます。コンテキスト エラーは表示されません。例えばXPath 条件式にてある属性が評価され、その属性がスキーマ内部で定義されていなかったとしても、そのエラーは表示されません。

アサーションメッセージ

アサーションの詳細がこの定義と共に提供される場合とても役に立ちます。XML インスタンスドキュメントが検証される際、アサーションが満たされていない場合、適切なメッセージが表示されます。XML スキーマ仕様は、このようなメッセージのために対応をしないため、XMLSpy は、アサーションの定義と共にXML インスタンスドキュメントの検証内で使用される `http://www.altova.com/ja/xml-schema-extensions`（または他の名前空間）内でメッセージを許可します。例えば

```
<xs:assert test="count(//MyNode) ge 1" altova:message="There must be at least one
MyNode element"/> or
<xs:assertion test="count(//MyNode) ge 1" altova:message="There must be at least one
MyNode element"/>
```

アサーション内で指定された制約が満たされない場合、XMLSpy の検証エンジンは、検証-エラーメッセージと共にヒントとしてのアサーションと関連したメッセージを表示します。バリデーターは、`assert/@message` 属性の値または、`message` 属性が存在する名前空間に関わらず、`assert/@message` 属性 を報告します。ですが、スキーマビュー内では、xml-schema-extensions 名前空間にある `message` 属性 のみを編集することができます。他の名前空間内の`message` 属性 を編集するには、テキストビューを使用します。

詳細に関しては、アサーションメッセージ[220]を参照してください。

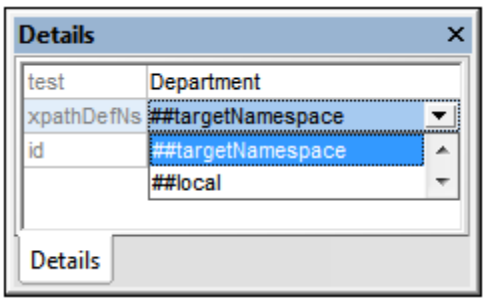
xpathDefaultNamespace の使用

XML スキーマドキュメントで宣言されたデフォルト名前空間は、そのXML スキーマドキュメントのデフォルト名前空間です。スキーマドキュメント内の接頭辞のない要素名に対して適用されますが、スキーマドキュメント内の接頭辞のないXPath 条件式に対しては適用されません。

XSD 1.1 にて新たに追加された`xpathDefaultNamespace` 属性（ XSD 1.1 の新規機能）は、XPath 条件式にて接頭辞のない要素の名前空間を指定するために使用されます。XPath デフォルト名前空間は、それらが宣言されたXML スキーマ要素上でスコープされます。`xpathDefaultNamespace` 属性は、以下のXML Schema 1.1 要素内に記述することができます:

- `xs:schema`
- `xs:assert` と`xs:assertion`
- `xs:alternative`
- `xs:selector` と`xs:field`（同一性制約にて）

XSD 1.1 モードでは、`xs:schema` 上の`xpathDefaultNamespace` がスキーマ設定ダイアログ（「スキーマ設計 | スキーマ設定」）にてセットされます。上に示されるその他の要素の場合、xpathDefaultNamespace 属性が詳細入力ヘルパーにてセットされます（以下のスクリーンショットを参照）。xs:schema 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、`xpathDefaultNamespace` 属性が許されている属性でオーバーライドすることができます（上のリストを参照ください）。



Xs:schema 上でXPath デフォルト名前空間を宣言することで、スキーマ全体に対するXPath デフォルト名前空間が宣言されます。この宣言は、xpathDefaultNamespace 属性が許されている属性でオーバーライドすることができます（*上のリストを参照ください*）。

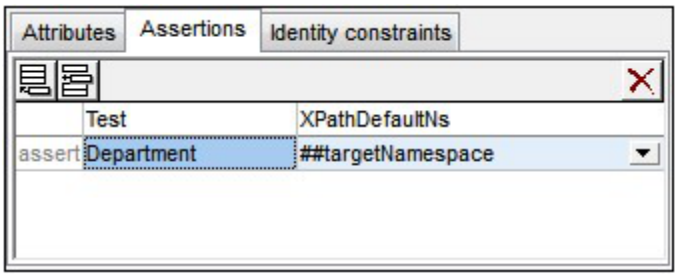
実際の名前空間を含める代わりにxpathDefaultNamespace 属性には、以下のうちのどれかを含めることができます:

- ##targetNamespace: スキーマのターゲット名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- ##defaultNamespace: スキーマの名前空間と同じ名前空間がXPath デフォルト名前空間として使用されます。
- ##local: XPath デフォルト名前空間は指定されません。

ドキュメントに対してXPath デフォルト名前空間が宣言されなかった場合、XPath 条件式における接頭辞なしの要素は名前空間を持たない要素として扱われます。

メモ XPath デフォルト名前空間宣言は属性に対して適用されません。

アサーション内にあるXPath 条件式アサーションの定義にてXPath デフォルト名前空間を定義することもできます。属性/アサーション/同一性制約（AAIDC）ペインのアサーション タブにて、XPathDefaultNs フィールドのドロップダウン リストから目的のキーワードを選択してください。



選択された名前空間が、*そのアサーションのスコープ*となります。

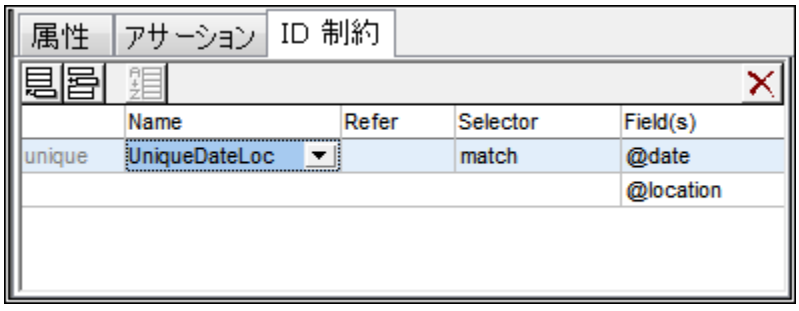
4.4.4.3 同一性制約

同一性制約（IDC）は*グローバル*要素宣言にて定義されます。同一性制約によりノードの一意性が指定され、ユニークなノード間の正確な参照を行うことができます。

宣言のメカニズム

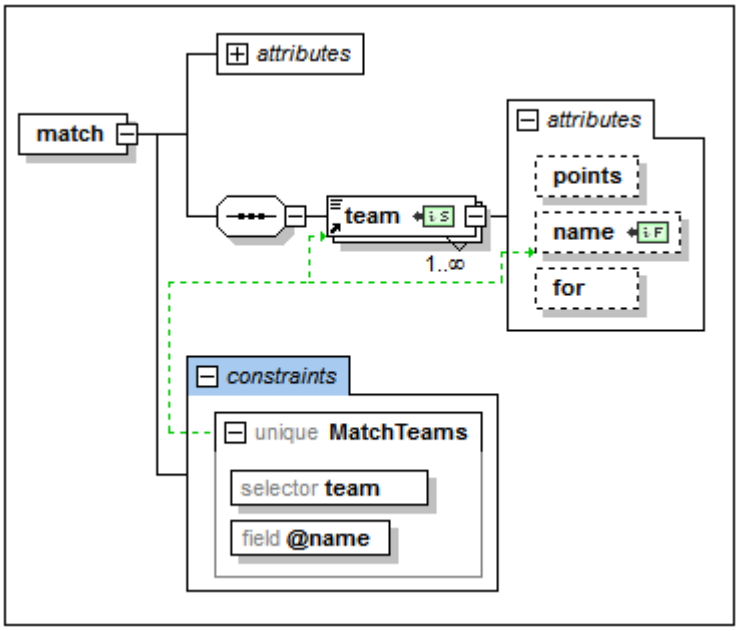
同一性制約（unique、key、keyref）を定義するために以下のメカニズムを使用することができます:

- スキーマ概要[164]に表示されたグローバル要素にて同一性制約を宣言することができます。グローバル要素を選択し、属性/アサーション/同一性制約（AAIDC）ペインにある同一性制約タブをクリックします（以下のスクリーンショットを参照）。



同一性制約タブにある挿入または追加アイコンにより、同一性制約（unique、key、keyref）を追加します。選択された同一性制約に対して field を追加することもできます。選択された field や同一性制約を削除するには、削除アイコンを使用してください。

- グローバル要素のコンテンツモデルビュー[176] では、グローバル要素やローカルの子孫要素にて同一性制約を定義することができます。このビューでは、同一性制約のタブと図の中に表示された要素の制約ボックスで同一性制約を編集することができます（以下のスクリーンショットでは、match 要素に team セレクターが含まれるというユニークな制約が示されています）。「スキーマ設計 | 表示設定」により表示されるスキーマ表示設定にて、またはスキーマ設計ツールバーにある「制約を図に表示」アイコンをクリックすることで、表示方法を切り替えることができます。図に示された同一性制約に対してドラッグ アンド ドロップ編集機能を使用することもできます。



同一性制約（unique、key、keyref）を図の中に追加するには、制約を行う要素を右クリックして、コンテキスト メニューから「子の追加 | <同一性制約の種類>」を選択してください。同一性制約が図の中で選択されていると、フィールド アイテムが選択可能になります。「Delete」キーを押下することで、選択された field または同一性制約を削除することができます。

以下にあるいずれかの方法で、selector ならびに field ボックスに XPath 条件式を入力することができます: (i) キーボードから入力する、(ii) selector または field ボックスをクリックすると表示されるドロップダウン リストから目的のノードを選択する、または (iii) ターゲット ノードを selector または field ボックスへドラッグして、ボックスがハイライトされたところでドロップする（XPath 条件式が自動的に作成されます）。

メモ　コンポーネント入力ヘルパーにある同一性制約のタブではスキーマ内にあるすべての同一性制約の概要[209] を確認することができます。詳細については、以下のコンポーネント入力ヘルパーを参照ください。

同一性制約アイコン

	制約を図に表示: コンテンツ モデルビューが表示されている際にスキーマ設計ツールバーで有効になります。同一性制約の表示を図または同一性制約タブで行うか指定します。
	同一性制約の視覚化: コンテンツ モデルビューにて選択可能になります。同一性制約の情報の表示/非表示を切り替えます。

	セレクター ノード、フィールド ノード: ノードの図にて表示され、これら 2 つのアイコンにより、`selector` ならびに `field` の XPath 条件式により選択されたノードがそれぞれ表示されます。

同一性制約の可視化

同一性制約の可視化アイコン が有効になっている場合、同一性制約の情報が図の中で可視化されます。可視化が有効になっている場合、`selector` と `field` の XPath 条件式により選択されたノードがボックス内のアイコンにより示され、緑色の線によりセレクターとフィールドのボックスが接続されます（上のスクリーンショットを参照ください）。

同一性制約の可視化アイコンにより、スキーマビューにおける同一性制約の検証機能も切り替えられます。XPath 条件式または同一性制約が間違っている場合、エラーが赤色で、警告がオレンジ色で示されます。XML スキーマドキュメントの検証を行うと、エラーと警告がメッセージ ウィンドウに表示されます。

同一性制約の可視化アイコン を無効にすることで、検証を無効にすることができます。

XML の内容

このセクションの下部に示される同一性制約の例は、以下の妥当なインスタンスドキュメントをベースにしています:

```
<results xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="Scores.xsd">
   <!----- Groups ----->
   <group id="A">
          <team name="Brazil"/>
          <team name="Germany"/>
          <team name="Italy"/>
          <team name="Holland"/>
   </group>
   <group id="B">
          <team name="Argentina"/>
          <team name="France"/>
          <team name="England"/>
          <team name="Spain"/>
   </group>
   <!----- matches ----->
   <match group="A" date="2012-06-12" location="Munich">
          <team name="Brazil" for="2" points="3"/>
          <team name="Germany" for="1" points="0"/>
   </match>
   <match group="A" date="2012-06-12" location="Frankfurt">
          <team name="Italy" for="2" points="1"/>
          <team name="Holland" for="2" points="1"/>
   </match>
   <match group="B" date="2012-06-13" location="Munich">
          <team name="Argentina" for="2" points="3"/>
          <team name="France" for="0" points="0"/>
   </match>
   <match group="B" date="2012-06-13" location="Berlin">
          <team name="England" for="0" points="1"/>
          <team name="Spain" for="0" points="1"/>
   </match>
</results>
```


(C) 2015-2021 Altova GmbH

ユニーク制約 (unique)

ユニーク制約により、要素または属性（要素や属性のセット）の値が定義されたスコープ内でユニーク（一意）でなければならないと指定されます。なんらかのリーグ戦を表している上のXML テキストでは、同じ名前のチーム同士が試合を行わないような制約が必要となるため、各 `match` 要素のスコープにて `team/@name` ノードの値がユニークであるような制約を作成します。以下の操作により制約を作成します:

1. スキーマ概要にて `match` 要素を選択します。`match` 要素が同一性制約のスコープとなります。
2. 同一性制約タブの左上にある「**追加**」または「**挿入**」アイコンをクリックして、メニューから「**ユニーク**」をクリックします。ユニーク制約の行が追加されます（以下のスクリーンショットを参照）。

3. 同一性制約の名前を入力します（上のスクリーンショットでは、`MatchTeams` という名前が与えられています）。
4. `team` 要素を選択するXPath 条件式を `Selector` 列に入力します。`match` 要素がコンテキスト ノードだということご注意してください。これで `team` 要素が同一性制約のセレクターとなり、ユニーク制約が適用されるノードとなります。
5. ユニークでなければならない `@name` ノードを Field 列に入力します。ユニーク制約が入力されたノードの値に対して適用されます。

上記のユニーク制約により、各 `match` 要素のスコープ内にてすべての `team` 要素にユニークな `@name` 属性値が必要になります。

更にフィールドを追加して、ユニーク制約をチェックすることもできます。例えば `result` 要素にて定義されたユニーク制約により、ユニークな日付と場所の組み合わせがすべての `match` に含まれていることをチェックすることができます（同じ日付に同じ場所で複数の試合を行うことができなくなります）。ユニーク制約により、各 `match` 要素（セレクター）に対して `@date` と `@location` の値を組み合わせた値が `result` 要素のスコープ内部でユニークとなる必要があります。

上記の方法により `result` 要素に対してユニーク制約を定義します。`match` がセレクターとなり、`@date` と `@location` がフィールドとなります（以下のスクリーンショットを参照）。追加アイコンをクリックして「**フィールド**」を選択することで、2 番目のフィールドが追加されます。

メモ　同一性制約タブのRefer 列はキー参照（`keyref`）制約でのみ有効になります。

キー制約 (key)

キー制約により、以下の制約が指定されます: (i) 要素や属性（または属性やスキーマのセット）の値が定義されたスコープ内でユニークであり、(ii) これらフィールド要素や属性がインスタンスXML ドキュメントに含まれている（従って省略可能な要素や属性をキー制約のフィールド

として選択すべきではありません)。キー制約により、ユニーク制約により与えられる制約に加え、指定されたフィールド要素/属性をXMLドキュメント内に含める必要があります。

既に登場した最初のユニーク制約と同様に、match 要素にて定義されたキー制約を以下のスクリーンショットに示します:

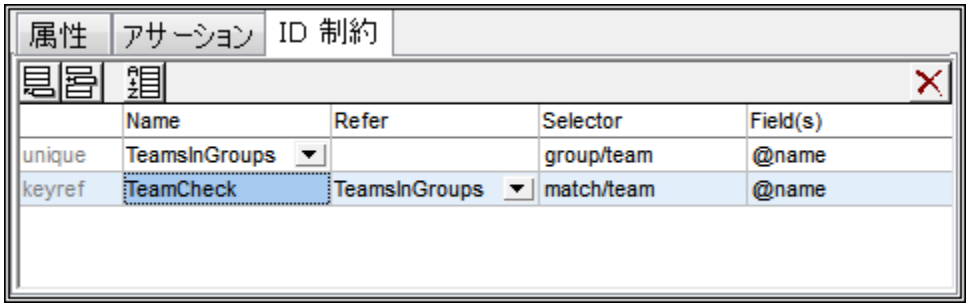
このキー制約により、各 match 要素のスコープ内部ですべての team 要素にユニークな @name 値が必要になります。更に、@name 属性がmatch/team 要素上に存在しなければならないとも指定されます。

メモ 同一性制約タブのRefer 列はキー参照(keyref) 制約でのみ有効になります。

キー参照(keyref)

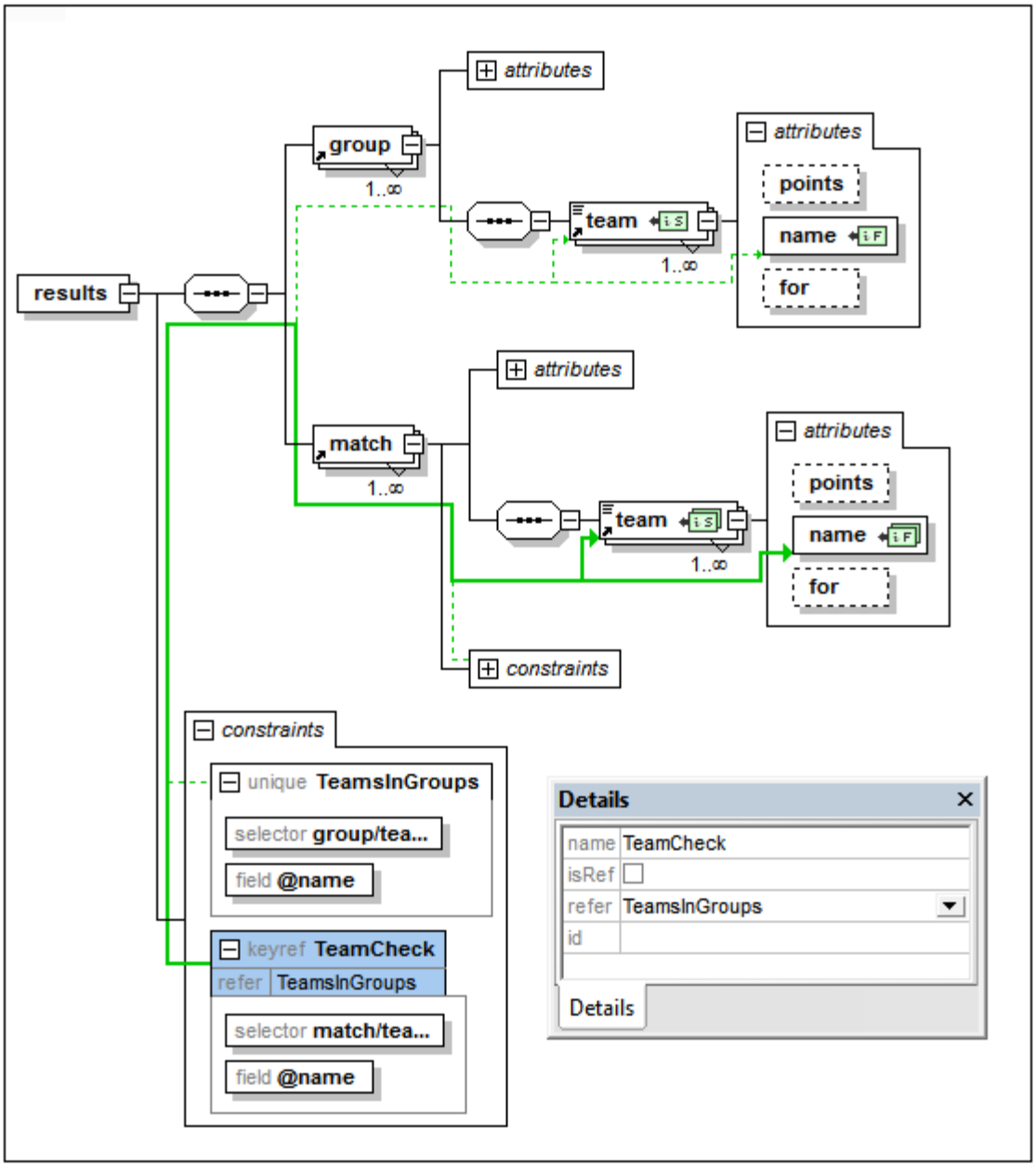
キー参照により、インスタンスドキュメント内にある値のセットがチェックされます。例えば上に示されるXML のテキストでは、キー参照を使うことにより同じグループに含まれているチーム同士で試合が行われるかをチェックすることができます。条件が満たされない場合、XMLドキュメントの検証に失敗します。

最初にユニーク制約またはキー制約を作成します。以下のスクリーンショットではTeamInGroups という名前のユニーク制約がresults 要素に対して作成されています。この制約により、group 要素内にある各 team にて、ユニークな @name 属性値が必要になります。



次に、match の子要素である team を選択し、@name の属性値がTeamInGroups により返された値に含まれるかどうかをチェックする、TeamCheck という名前のキー参照(keyref) を作成します。

以下のスクリーンショットでは、(水色でハイライトされている) このキー参照と、詳細入力ヘルパーの内容が示されています。選択された同一性制約との関係は緑色の実線で示されており、選択されていない同一性制約は緑色の破線により示されています。それぞれの同一性制約にてselector とfield のXPath 条件式にて選択されたノードが、それぞれ と アイコンにより示されます。ノードが縮退されると、関係を表す線が破線状態で表示されます。



xpathDefaultNamespace の使用

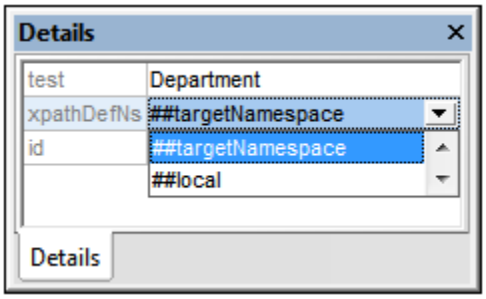
XML スキーマドキュメントで宣言されたデフォルト名前空間は、そのXML スキーマドキュメントのデフォルト名前空間です。スキーマドキュメント内の接頭辞のない要素名に対して適用されますが、スキーマドキュメント内の接頭辞のないXPath 条件式に対しては適用されません。

XSD 1.1 にて新たに追加された `xpathDefaultNamespace` 属性（ XSD 1.1 の新規機能）は、XPath 条件式にて接頭辞のない要素の名前空間を指定するために使用されます。XPath デフォルト名前空間は、それらが宣言されたXML スキーマ要素上でスコープされます。`xpathDefaultNamespace` 属性は、以下のXML Schema 1.1 要素内に記述することができます:

- `xs:schema`
- `xs:assert` と`xs:assertion`

- xs:alternative
- xs:selector と xs:field (同一性制約にて)

XSD 1.1 モードでは、xs:schema 上の xpathDefaultNamespace がスキーマ設定ダイアログ (「スキーマ設計 | スキーマ設定」) にてセットされます。上に示されるその他の要素の場合、xpathDefaultNamespace 属性が詳細入力ヘルパーにてセットされます (以下のスクリーンショットを参照)。xs:schema 上で XPath デフォルト名前空間を宣言することで、スキーマ全体に対する XPath デフォルト名前空間が宣言されます。この宣言は、xpathDefaultNamespace 属性が許されている属性でオーバーライドすることができます (*上のリストを参照ください*)。



Xs:schema 上で XPath デフォルト名前空間を宣言することで、スキーマ全体に対する XPath デフォルト名前空間が宣言されます。この宣言は、xpathDefaultNamespace 属性が許されている属性でオーバーライドすることができます (*上のリストを参照ください*)。

実際の名前空間を含める代わりに xpathDefaultNamespace 属性には、以下のうちのどれかを含めることができます:

- ##targetNamespace: スキーマのターゲット名前空間と同じ名前空間が XPath デフォルト名前空間として使用されます。
- ##defaultNamespace: スキーマの名前空間と同じ名前空間が XPath デフォルト名前空間として使用されます。
- ##local: XPath デフォルト名前空間は指定されません。

ドキュメントに対して XPath デフォルト名前空間が宣言されなかった場合、XPath 条件式における接頭辞なしの要素は名前空間を持たない要素として扱われます。

メモ XPath デフォルト名前空間宣言は属性に対して適用されません。

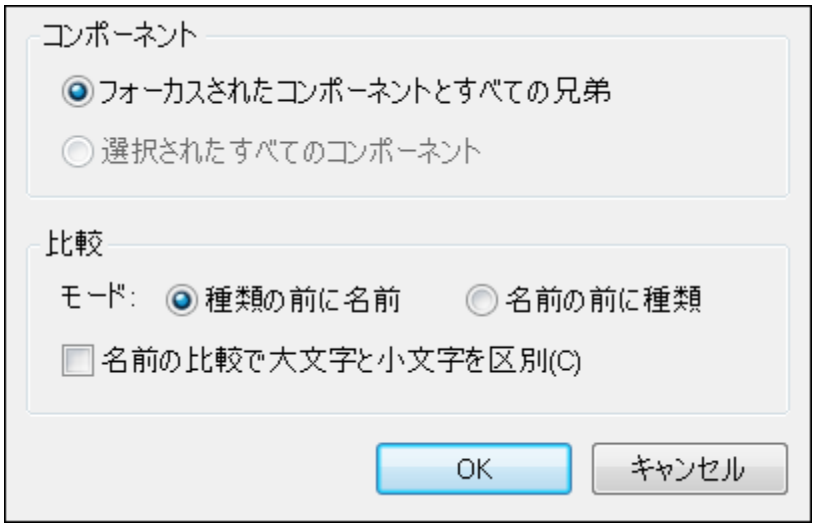
同一性制約の ID

同一性制約とそのセレクター、フィールドに対して ID を与えることができます。ID を割り当てるには必要なコンポーネントを選択し、詳細入力ヘルパーの ID 行にて ID を入力してください。

同一性制約の並べ替え

同一性制約タブのツールバーにある「並べ替え」アイコンをクリックすることで、同一性制約の並べ替えを行うことができます。コンポーネントの並べ替えダイアログでは、選択されたコンポーネントとその兄弟、または選択されたコンポーネントのセットを並べ替えの対象とすることができます。「Shift」キーを押下しながらクリックすることで範囲の選択を、「Ctrl」キーを押下しながらクリックすることで新たなコンポーネントを選択対象に含めることができます。

(C) 2015-2021 Altova GmbH



並べ替えの範囲を選択した後には、すべての対象をアルファベット順で並べ替えるのか(種類の前に名前)、または同一性制約の種類ごと(ユニーク制約、キー制約、キー参照の順)でアルファベット順に並べ替えるのかを選択することができます。

並べ替えの順序はスキーマのテキストに反映されます。

4.4.5 スキーマ ビューにおける入力ヘルパー

スキーマビューにて 3 つの入力ヘルパーが表示されます。このセクション以下にあるサブセクションでは、これら入力ヘルパーの詳細について説明されます:

- コンポーネント入力ヘルパー 209
- 詳細入力ヘルパー 213
- ファセット入力ヘルパー 215

表示される入力ヘルパーはスキーマ概要とコンテンツ モデルビューで変わらず、スキーマ コンポーネントの定義を視覚的に追加ならびに編集することができます。入力ヘルパーからコンポーネントをドラッグすることができるほか、デザイン上でコンポーネントを選択して、そのプロパティを入力ヘルパーで定義することができます。

4.4.5.1 コンポーネント

スキーマビューにおけるコンポーネント入力ヘルパーでは、3 つの操作を行うことができます:

- コンポーネントの種類と名前空間により、ツリー内にあるグローバル コンポーネントを管理する(以下のスクリーンショットを参照): これにより、名前空間に従うかたちですべてのグローバル コンポーネントの概要が表示されます。
- グローバル コンポーネントのコンテンツ モデルビューへのナビゲーションを行う(コンポーネントにコンテンツ モデルが含まれている場合): コンポーネントにコンテンツ モデルが含まれない場合、スキーマ概要にてコンポーネントがハイライトされます。他のスキーマからインクルードまたはインポートされたグローバル コンポーネントもコンポーネント入力ヘルパーに表示されます。
- スキーマドキュメント内で定義された同一性制約の概要を表示する 同一性制約タブの詳細については、「同一性制約」 202 のセクションを参照ください。

メモ 表示されるのがXSD 1.0 なのか1.1 なのかは、選択されているXSD モード（XSD 1.0 または1.1）により決定されます。

グローバルタブ

グローバルタブでは、グローバルコンポーネントがツリー構造でグループ化され表示されます（以下のスクリーンショットを参照）。スキーマ内にある各グローバルコンポーネントの種類の数が、各種コンポーネントの隣に表示されます。

ツリー構造では、グローバルコンポーネントが以下の7 つの種類に分類されます。コンポーネントの種類がツリーに表示されるには、その種類のコンポーネントがスキーマ内部に最低 1 つ必要になります。

- 要素宣言（要素）
- モデルグループ（グループ）
- 複合型
- 単純型
- 属性宣言（属性）
- 属性グループ
- 記法

名前空間タブ

名前空間タブでは、名前空間とコンポーネントの種類により分類されたかたちでコンポーネントが表示されます（以下のスクリーンショットを参照）。

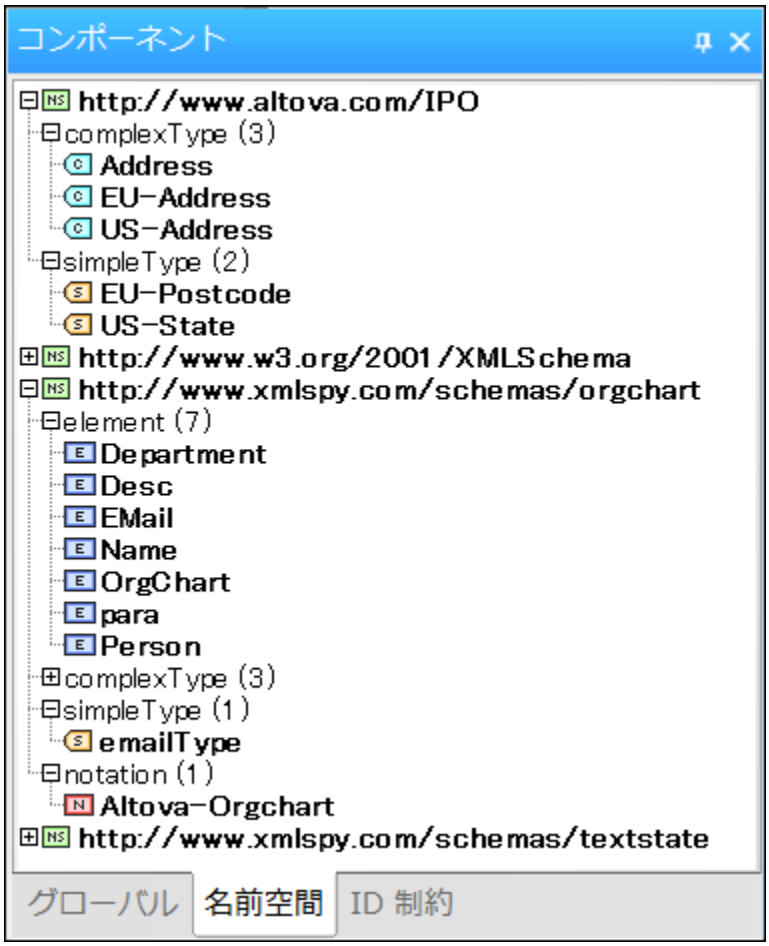
(C) 2015-2021 Altova GmbH

ツリーの表示では、グローバルコンポーネントが以下にある 7 つのグループに分類されます。コンポーネントの種類がツリーに表示されるには、その種類のコンポーネントがスキーマ内部に最低 1 つ必要になります。

- 要素宣言 (要素)
- モデルグループ (グループ)
- 複合型
- 単純型
- 属性宣言 (属性)
- 属性グループ
- 記法

グローバルと名前空間タブに表示されるコンポーネントのグループ

グローバルまたは名前空間タブにてコンポーネントのグループを展開すると、そのグループに含まれるすべてのコンポーネントが表示されます。これにより、ユーザー定義のコンポーネントへ簡単にアクセスすることができます。コンポーネント タブに表示されたコンポーネントをダブルクリックすると、その定義がメイン ウィンドウに表示されます。

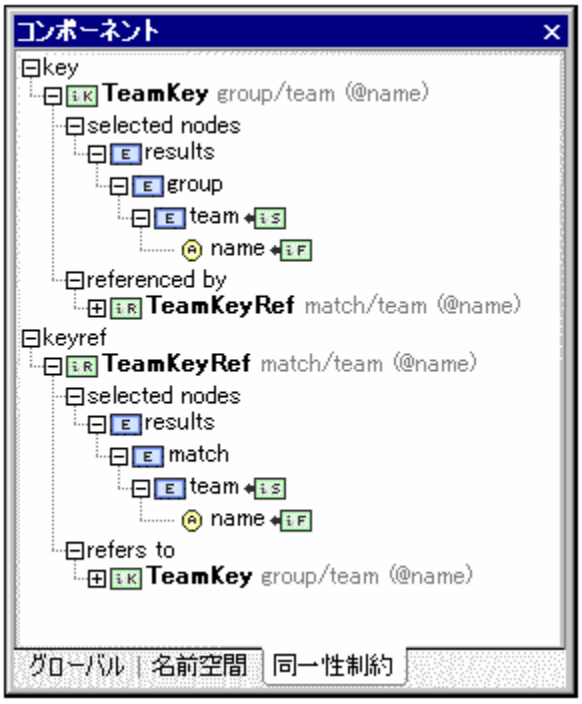


コンポーネントにコンテンツ モデルが含まれている場合（つまり、コンポーネントが要素、グループ、または複合型の場合）、コンポーネント名をダブルクリックすることで、コンポーネントのコンテンツ モデルが（メイン ウィンドウにある）コンテンツ モデル ビューに表示されます。コンテンツ モデルを含まないコンポーネントがダブルクリックされた場合、そのコンポーネントが（メイン ウィンドウにある）スキーマ概要でハイライトされます。

注意: インクルードまたはインポートされたスキーマの中にコンポーネントが含まれる場合、インクルード/インポートされたスキーマが開かれ、コンポーネントのコンテンツ モデルがコンテンツ モデル ビューに表示されるか、コンポーネントがスキーマ概要でハイライトされます。

同一性制約

コンポーネント入力ヘルパーの同一性制約タブでは、ドキュメントにある同一性制約の概要が表示されます（以下のスクリーンショットを参照）。このタブでは、同一性制約の種類（ユニーク（unique）、キー（key）、キー参照（keyref））により分類されたかたちで同一性制約が表示されており、展開/縮退可能なツリーにより同一性制約が表示されます。



太字で表示されているエントリーはその同一性制約が現在アクティブなスキーマ内部にあることを表しており、そうでないエントリーはサブスキーマ内に含まれることを表しています。同一性制約タブに表示されたエントリーをダブルクリックすることで、そのスキーマ コンポーネントがコンテンツ モデルビュー[202]にて表示されます。

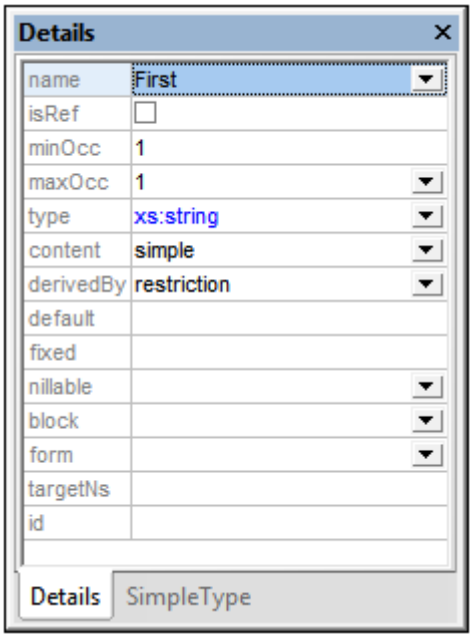
同一性制約タブにてアイテムが選択されている状態では、以下のコンテキスト メニュー コマンドを利用することができます:

- **図に表示**: スキーマ コンポーネントをコンテンツ モデルビュー[202]に表示します。
- **図にセレクター/フィールド ターゲットを表示**: 同一性制約のセレクター(selector) またはフィールド (field) にて指定されているスキーマ コンポーネントをコンテンツ モデルビュー[202]に表示します。複数のフィールドがある場合、目的のフィールドを選択するためのダイアログが表示されます。
- **同一性制約へ移動**: 同一性制約をスキーマ概要[202]に表示します。
- **すべての子を展開/縮退**: ツリーを展開/縮退します。

同一性制約タブの詳細については、「同一性制約」[202] のセクションを参照ください。

4.4.5.2 詳細

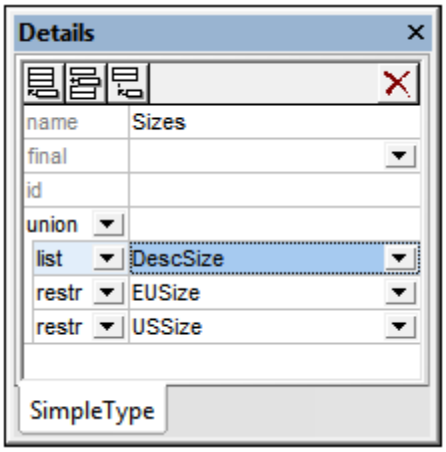
スキーマ ビューの詳細入力ヘルプ パネルは、メイン ウィンドウにて現在選択されているコンポーネントやコンポジッターに関する編集可能な情報が表示されます。データベース拡張を含んでいるスキーマ ファイルを編集している場合、DB 拡張に関する情報を表示するためのタブも表示されます。



現在選択されているコンポーネントやコンポジッターのプロパティを変更するには、編集するフィールドをダブルクリックして、値を選択するか直接入力してください。編集するフィールドにてコンボボックスが利用できる場合、ドロップダウンリストから目的の値を選択してください。詳細入力ヘルパーにて行った変更は、直ちにデザインへ反映されます。

単純型の派生

詳細入力ヘルパーにて派生した単純型（restriction、list、union）を素早く正確に作成することができます。デザインにて単純型が選択されると、詳細入力ヘルパーに単純型タブが表示されます（以下のスクリーンショットを参照）。



単純型タブにある派生型のコンボボックスにて派生型（restriction、list、union）を選択し、右側にある対応するメンバーの型で利用可能な単純型から目的の単純型を選択してください。ツールバーにあるアイコンを使用することで、同じレベルに型を追加または挿入し、サブレベルに別の派生を追加し、派生型を削除することができます。型の定義に移動するには、「型の定義」をクリックします。この場合、ビルトインの単純型の場合、単純型の情報を含むメッセージボックスが表示されます。

4.4.5.3 ファセット

単純型の基底型（既存の単純型）に制約を課すことで新たな単純型（名前付き、または匿名型）が作成されます。このような制約は、基底型の値にファセットを追加することで達成されます。スキーマ ビューにあるファセット入力ヘルパーでは、単純型のファセットを簡単かつ視覚的に編集することができます（以下のスクリーンショットを参照）。利用可能なファセットは、以下のテーブルで示されるようにファセット入力ヘルパーのタブにて管理されます。

タブ	利用可能なファセット
ファセット 215	`minInclusive`、`maxInclusive`、`minExclusive`、`maxExclusive`、`length`、`minLength`、`maxLength`、`totalDigits`、`fractionDigits`、`whiteSpace`、`explicitTimezone`
パターン 216	`pattern`
列挙 216	`enumeration`
アサーション 217	`assertion`
サンプル 218	`altova:exampleValues` はサンプルであり、ファセットではありません。サンプルは XMLSpy にて XML スキーマからインスタンス XML ドキュメントを生成する際のサンプル値として使用されます。

以下に、これらタブの詳細について記述します。

デザイン内にある単純型の選択

単純型（名前付きまたは匿名型）は、以下のデザイン環境にて選択することができます:

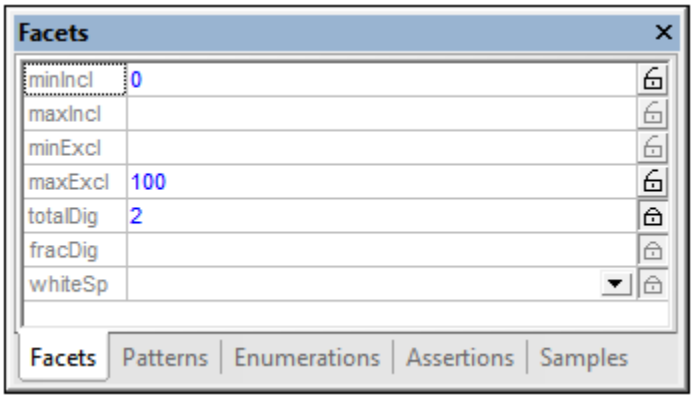
- スキーマ概要 164 にて: グローバルコンポーネントのリスト、またはグローバルコンポーネントのリスト以下にある属性タブから。
- コンテンツ モデル ビュー 176 にて: 図の中、またはデザイン下部にある属性タブから。

上記のデザイン環境にて単純型が選択されると、定義されているファセットがファセット入力ヘルパーに表示され、編集することができるようになります。

ファセット タブ

ファセット タブではデザインにて選択されている型に対して利用可能なファセットだけが表示されます。例えば制約を課されているのが `xs:string` 型の場合、`totalDigits` などの利用できないファセットは表示されません。

- 数値に関するファセット（`minInclusive`、`maxInclusive`、`minExclusive`、`maxExclusive`）は数値型と日付/時間型、そしてこれらの型から派生した型に対してだけ使用することができます。
- 長さに関するファセットは `string` を基底とした型、バイナリ型、そして `anyURI` に対してだけ使用することができます。
- `totalDigits` ファセットは `xs:decimal` ならびに整数型と、これらの型から派生したすべての型に対して使用することができます。`fractionDigits` ファセットは `xs:decimal` に対してのみ適用されます。

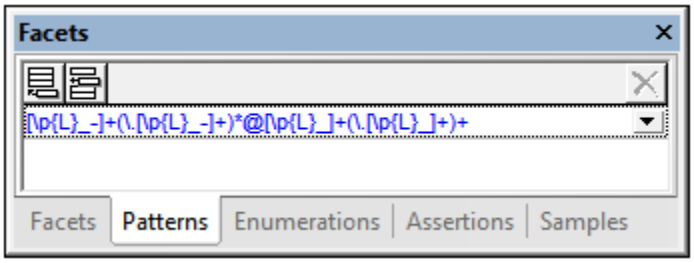


値を入力するには、(それが存在する場合には) コンボボックスのドロップダウンリストから値を選択、または値フィールドをダブルクリックして値を入力してください。無効な値や他の値と相反する値入力された場合、その値が赤色で表示され、妥当な値は青色で表示されます。例えば、`minInclusive` ファセットと`maxInclusive` ファセットは同時に定義することができないため、これらのうち後に入力された値が赤色で表示されます。

固定ファセット (fixed="true" の属性を持つファセット) を指定するには、ファセットの右側にある鍵のシンボルをクリックして、シンボルを閉じた鍵に変更してください。上のスクリーンショットでは、`totalDigits` ファセットが固定ファセットとして定義されています。固定することのできるファセットの数に制限はありません。ファセットの固定を解除するには、閉じた鍵のシンボルをクリックしてください。

パターン タブ

パターン タブでは、1 つ以上の`pattern` ファセットを制約に追加することができます。`pattern` ファセットのパターンは正規表現の構文により指定されます。以下のスクリーンショットでは、電子メールアドレスのパターンが示されます。

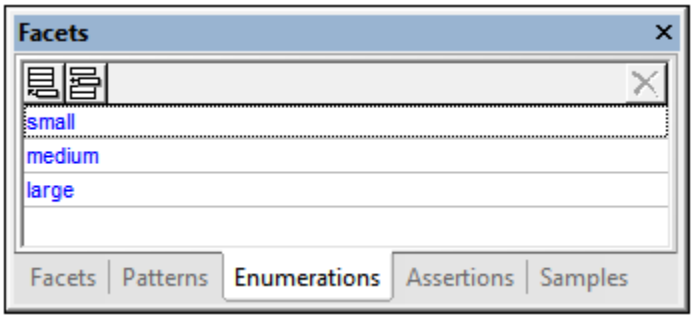


複数の`pattern` ファセットが指定されている場合、XML インスタンスの値が指定されたパターンの 1 つにマッチする必要があります。例えば、米国と欧州の郵便番号にマッチするパターンが与えられている場合、XML インスタンスの値を、そのうち 1 つに対してマッチさせる必要があります。

左上にある「**追加**」または「**挿入**」アイコンをクリックして、目的のパターンを定義する正規表現を入力することで、`pattern` アセットを追加することができます。パターンを選択して、右上にある「**削除**」アイコンをクリックすることで、そのパターンを削除することができます。

列挙タブ

列挙タブでは、`enumeration` ファセットを制約に加えることができます (以下のスクリーンショットを参照)。各 `enumeration` ファセットにより、その型に対して妥当となる値が指定されます。`enumeration` ファセットのセットにより、許されている値のセットを指定することができます。以下のスクリーンショットでは、`enumeration` ファセットにより許されている文字列のセットを確認することができます。



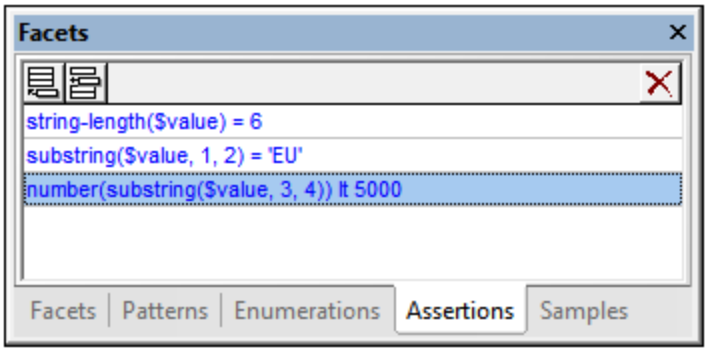
左上にある「**追加**」 または「**挿入**」アイコンをクリックして `enumeration` の値を入力することで、`enumeration` ファセットを追加することができます。`enumeration` ファセットを選択し、右上にある「**削除**」アイコンをクリックすることで、その `enumearation` ファセットを削除することができます。

アサーション タブ

アサーションは XSD 1.1 にて新たに追加された機能で、XSD 1.1 モード[161] でのみ表示されます。ファセット入力ヘルパーのアサーション タブにて定義されたアサーション ファセットは、**単純型に対するアサーション**です（複合型に対するアサーションは、ファセット入力ヘルパーではなくスキーマ概要やコンテンツ モデル ビューにて定義ならびに編集[199] されます）。

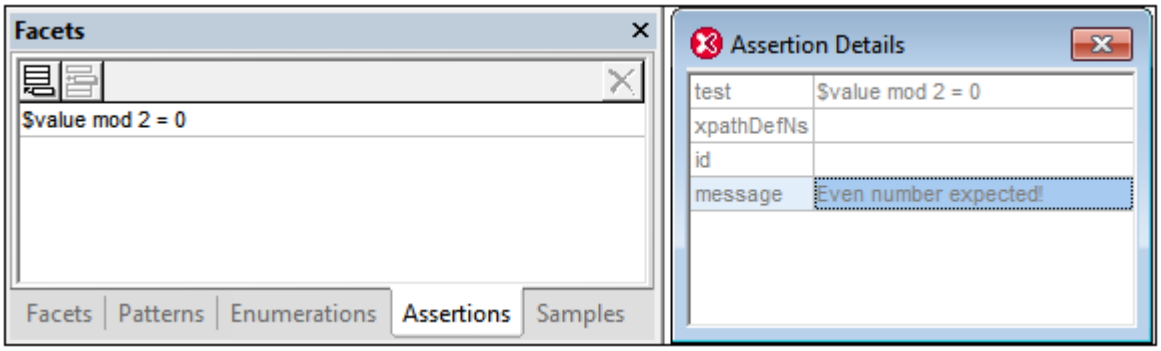
単純型（要素または単純コンテンツの属性）がデザインで選択されている際にアサーション タブへ切り替え、左上にある「**追加**」または「**挿入**」アイコンをクリックし、`assertion` を定義するのに使われる XPath 2.0 条件式を入力することで、アサーションの指定を行うことができます。XPath 条件式では、`$value` という特別な変数を使用することで、単純型の値を参照することができます（1 つの値以外に評価する要素はないため、通常使用される `self::node()` パス（またはその省略形であるピリオド '.'）を XPath 条件式内部で使用することはできません）。

例えば、`string-length($value) = 6` という XPath 条件式により、単純型に含まれている値が 6 文字から構成されるかをチェックすることができます。インスタンス ドキュメント内に含まれている要素や属性の値が 6 文字から構成される場合、アサーションに従った妥当な値ということになります。



メモ XPath 条件式における構文エラーは赤色により示されます。しかしデータ型はランタイムにて決定されるため、型に関するエラーは XPath 条件式の入力中に報告されません。そのため期待されたとおりに型の構築を行う必要があります。例えば上のスクリーンショットにある 3 番目の条件式では、数値の比較を行う前に、文字列の値（アサーションが `xs:string` 単純型にて定義されたと仮定した場合）を数値型へ変換しています。

上のスクリーンショットの例にあるように、複数のアサーションを同じ単純型へ指定することもできます。この場合、指定されているすべてのアサーションに要素または属性の値をマッチさせる必要があります。上のスクリーンショットにあるアサーションでは、インスタンス ドキュメントの値が 6 文字から構成され、`EU` という文字から開始され、最後の 4 文字を `0000` から `4999` までの数値に収める必要があります。



アサーションの詳細がこの定義と共に提供される場合とても役に立ちます。XML インスタンスドキュメントが検証される際、アサーションが満たされていない場合、適切なメッセージが表示されます。XML スキーマ仕様は、このようなメッセージのために対応をしないため、XMLSpy は、アサーションの定義と共にXML インスタンスドキュメントの検証内で使用される `http://www.altova.com/ja/xml-schema-extensions`（または他の名前空間）内でメッセージを許可します。例えば

```
<xs:assert test="count(//MyNode) ge 1" altova:message="There must be at least one
MyNode element"/> or
<xs:assertion test="count(//MyNode) ge 1" altova:message="There must be at least one
MyNode element"/>
```

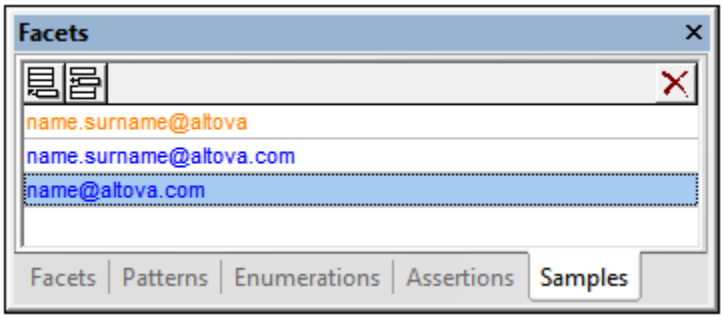
アサーション内で指定された制約が満たされない場合、XMLSpy の検証エンジンは、検証エラーメッセージと共にヒントとしてのアサーションと関連したメッセージを表示します。バリデーターは、`assert/@message` 属性の値または、`message` 属性が存在する名前空間に関わらず、`assert/@message` 属性 を報告します。ですが、スキーマビュー内では、xml-schema-extensions 名前空間にある `message` 属性 のみを編集することができます。他の名前空間内の`message` 属性 を編集するには、テキストビューを使用します。

詳細に関しては、アサーションメッセージ 220 を参照してください。

メモ 可能な場合はアサーションよりもその他の種類のファセット使用する方が推奨されます。例えば、上のスクリーンショットにある最初のアサーションにより指定された制約は、（ファセット タブにある）`length` ファセットにより指定された方が良いでしょう。

サンプル タブ

サンプル タブでは、XML スキーマから XML インスタンス ファイルを生成する際のサンプル値として使用される値を指定することができます（以下のスクリーンショットを参照）。XML インスタンス ファイルの生成は、メニューオプションの「DTD/スキーマ | サンプル XML ファイルの生成」から行うことができます。サンプル値が無効の場合、警告としてサンプル値がオレンジ色で表示されます。以下のスクリーンショットでは最初のサンプルが電子メールの`pattern` ファセットにマッチしないため、無効な値となります（上記の パターン タブを参照ください）。



メモ ツールバーにある「**検証の警告を表示**」アイコン をクリックすることで、無効なサンプル値の警告を確認することができます。無効なサンプル値によりXSD ファイルの検証に失敗することはありません。

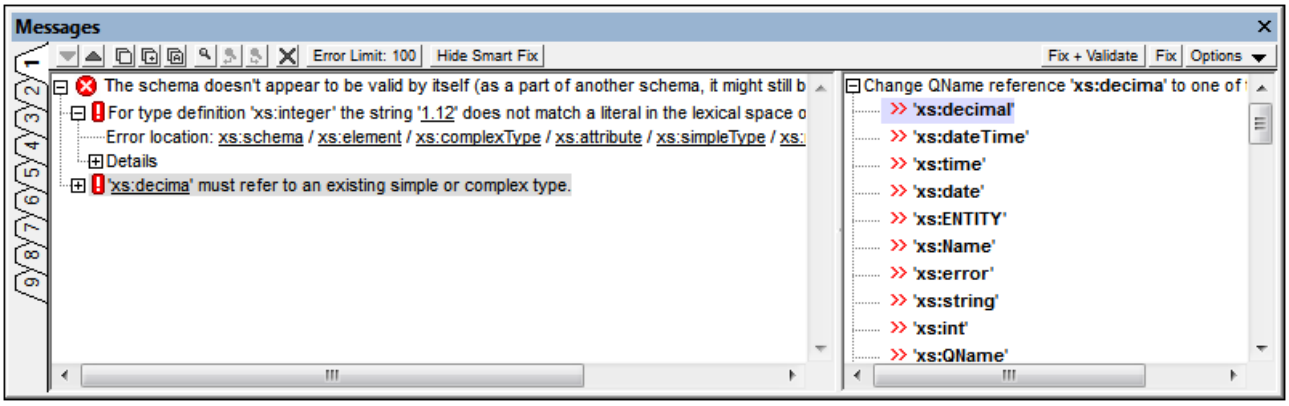
左上にある「追加」または「挿入」アイコンをクリックしてhttp://www.altova.com/ja/xml-schema-extensions内のサンプル値を入力することで、`altova:example` へサンプル値が入力されます。サンプル値を選択し、右上にある「削除」アイコンをクリックすることで、`altova:example` の値が削除されます。

4.4.6 検証とスマートフィックス

XML スキーマドキュメントの正確性の確認のために検証を行うことができます。これはメニューコマンド「XML | XML の検証 (F8) 」をクリックして行います。

ドキュメントが有効な場合、この効果についてのメッセージがメッセージウィンドウに表示されます。

ドキュメントが無効な場合、メッセージ ウィンドウは、2 つのペインの表示をするように変更されます (*下のスクリーンショット参照*)。左側のペイン (エラー ペイン) は、最初の x エラーまたはすべてのエラーをリストします。右側のペインは、スマートフィックス ペインです。このペインは左側のペインで選択されたえらのためのフィックスの選択肢のリストを含みます。下のスクリーンショットの例では、エラー ペイン内の2 番目のエラーの選択は、右側の スマートフィックス ペインにこのエラーのフィックスの選択肢の表示をします。フィックスの 1 つを選択してを「フィックス+ 検証」またはフィックス選択すると、ドキュメント内のエラーはこの特定のフィックスと共に修正されます。



エラー ペイン

ウィンドウのツール バーには、次の機能があります :

- 「上へ」および「下へ」矢印を使用して、エラーをスクロールします。
- メッセージ、またはメッセージとその子孫、またはすべてのメッセージをクリップボードにコピーします。
- 検索を使用して、用語を検索します。複数のエラーが報告されている場合、次を検索、および前を検索 機能は、役に立ちます。
- エラー ペイン空すべてのエラーをクリアします。
- 検索され表示されるエラー数の制限を設定します (`1` から `999`)。デフォルトの値は `100` です。ボタンをクリックして制限を編集します。
- 表示/非表示 スマートフィックス ペイン. スマートフィックス ペインが非表示の場合、「スマートフィックスを表示」ボタンがツール バーに表示されます。クリックするとスマートフィックスペインが表示され、ボタンが「スマートフィックスを非表示」に変更されます。「表示/スマートフィックスを非表示」ボタンが、有効化されていない場合、スマートフィックスを使用することができません。

スマートフィックス ペイン

ウィンドウのツール バーで次の機能を使用することができます:

- 「フィックス+検証」ボタンは、選択されたエラーを選択されたスマートフィックスで修正します。そしてドキュメントを再検証します。他のエラーは、エラーペインで再度開かれます。
- フィックスボタンをクリックすると、エラーはフィックスされますが、再検証は行われません。
- オプションボタンは、スマートフィックスをダブルクリックすることにより作動する振る舞いの選択を含むリストをドロップします: ダブルクリックにより「フィックス+検証」または「フィックス」を実行するかどうか。

4.4.7 アサーションメッセージ

XML Schema 1.1 内では、 複合型 (xs:assert 要素を使用して) および単純型 (xs:assertion 要素を使用して) のためにアサーションは、定義されることができます。

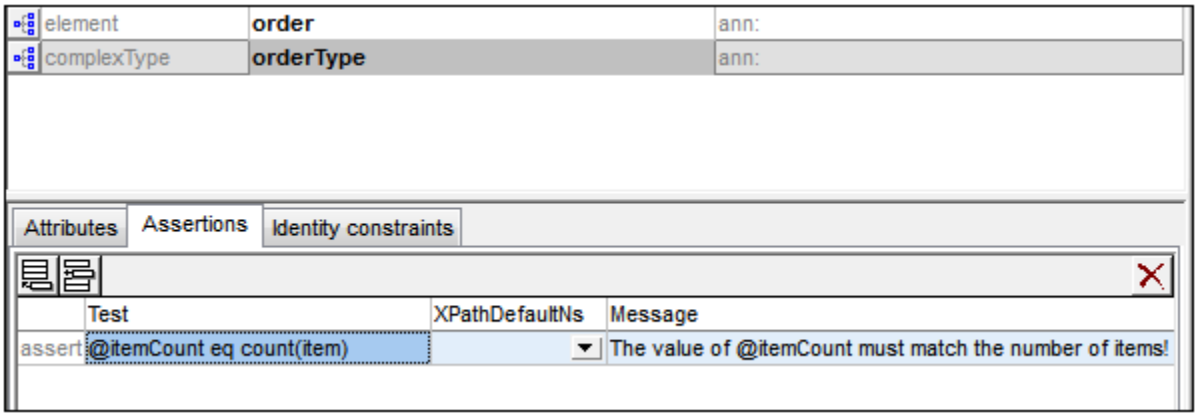
アサーションの詳細がこの定義と共に提供される場合とても役に立ちます。XML インスタンスドキュメントが検証される際、アサーションが満たされていない場合、適切なメッセージが表示されます。XML スキーマ仕様は、このようなメッセージのために対応をしないため、XMLSpy は、アサーションの定義と共にXML インスタンスドキュメントの検証内で使用される http://www.altova.com/ja/xml-schema-extensions (または他の名前空間) 内でメッセージを許可します。例えば

```
<xs:assert test="count(//MyNode) ge 1" altova:message="There must be at least one
MyNode element"/> or
<xs:assertion test="count(//MyNode) ge 1" altova:message="There must be at least one
MyNode element"/>
```

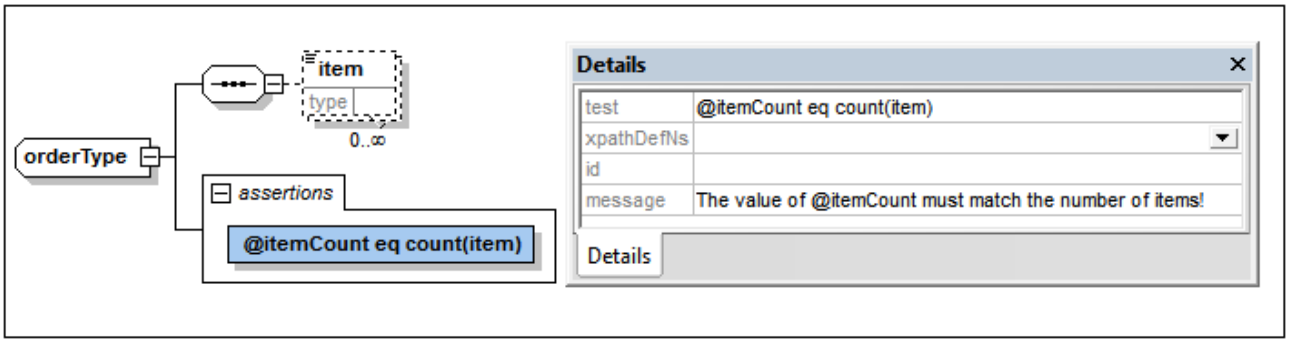
アサーション内で指定された制約が満たされない場合、XMLSpy の検証エンジンは、検証-エラーメッセージと共にヒントとしてのアサーションと関連したメッセージを表示します。バリデーターは、assert/@message 属性の値または、message 属性が存在する名前空間に関わらず、assert/@message 属性 を報告します。ですが、スキーマビュー内では、xml-schema-extensions 名前空間にある message 属性 のみを編集することができます。他の名前空間内の message 属性 を編集するには、テキストビューを使用します。

xs:assert メッセージの編集

スキーマビュー内では、(複合型のための) xs:assert 要素 を作成して、属性/アサーションID 制約 (AAIDC) ペイン[199] または関連する複合型の詳細入力ヘルパー[199] 内で編集することができます。下のスクリーンショットは、複合型 orderType のためのアサーションを表示します。 (この場合は、xs:assert) は、(スキーマ概要の) アサーション タブ でアサーション メッセージと共に定義されています。

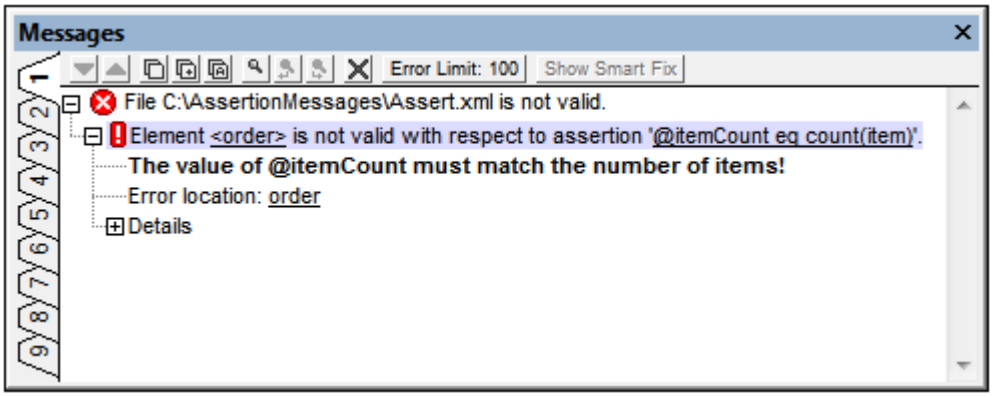


ダイアグラム内でアサーションを表示する[181] オプション が選択された場合、複合型 のアサーションが作成され、コンテンツモデルビュー内で編集されます。アサーション メッセージを追加または編集するには、アサーションを選択して、詳細入力ヘルパー内のアサーションメッセージに入力してください (*下のスクリーンショット参照*)。



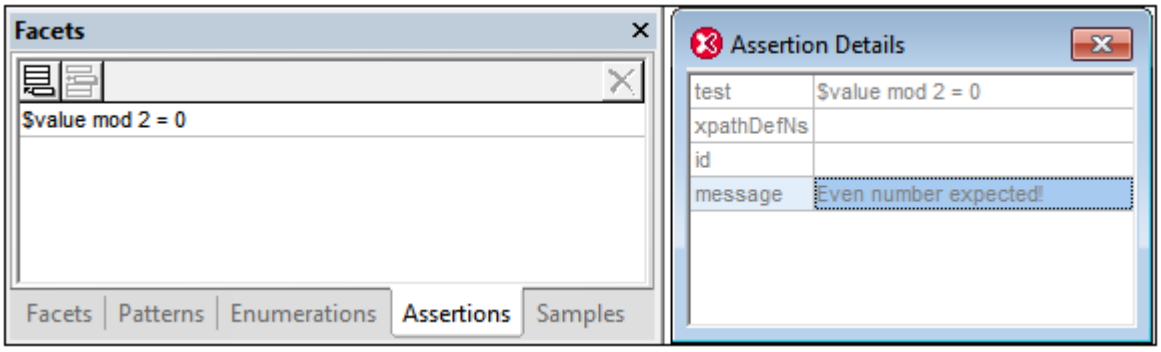
この方法で作成されるアサーションメッセージは、Altova xml-schema-extension 名前空間 `http://www.altova.com/ja/xml-schema-extensions` 内にあります。 XML スキーマドキュメント内に最初のアサーションメッセージを、AADC ペイン[199] または詳細入力ヘルパー[199]を介して追加すると、Altova xml-schema-extension 名前空間が自動的に`xs:schema` 要素に対して宣言されます。

XML ファイルが検証されてアサーションテストが満たされない場合、アサーションのために定義された、エラーメッセージと共に表示されます (*下のスクリーンショット参照*)。

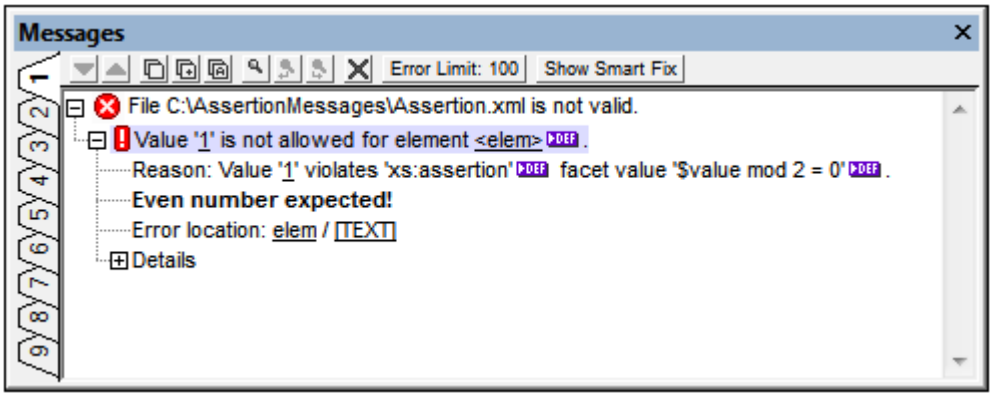


xs:assertion メッセージの編集

スキーマビューの`xs:assertion` 要素 (単純型) は、関連する単純型のファセット 入力ヘルパー[215] 内で作成および編集されることができます。アサーションメッセージを編集するには、ファセット 入力ヘルパー (*下のスクリーンショット参照*) 内のアサーションを右クリックして、ポップアップするメニューの「詳細」をクリックし、アサーション詳細モーダルウィンドウのメッセージを編集します (*下のスクリーンショット参照*)。メモ この方法で作成される、アサーションメッセージは、Altova xml-schema-extension 名前空間 `http://www.altova.com/ja/xml-schema-extensions` 内にあります。アサーション詳細モーダルウィンドウを介して、XML スキーマドキュメント内のメッセージに最初のアサーションを追加する際には、名前空間は、自動的に`xs:schema` 要素 に対して宣言されます。



XML ファイルが検証されて、アサーションテストが満たされない場合、アサーションのために定義されたメッセージがエラーメッセージと共に表示されます（*下のスクリーンショット参照*）。

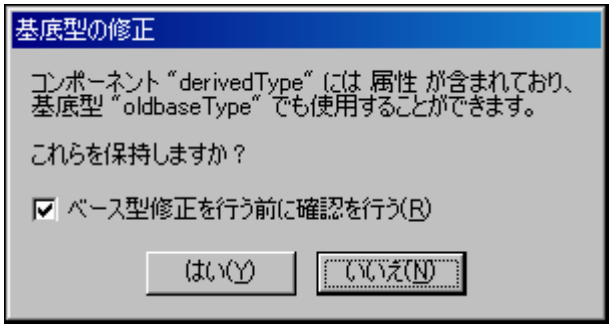


4.4.8 基底型の修正

派生した型の基底型がスキーマビューで変更されると、派生型にあるコンテンツ、属性、ファセット、サンプル値を、以下のいずれかの方法で処理することができます:

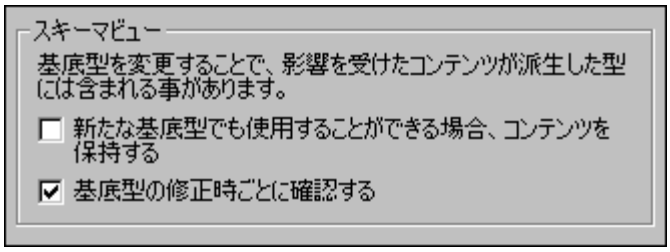
- 新たな基底型でも使用できる場合、これらの値を保持する。
- 新たな基底型でも使用できるかどうかにかかわらず、これらの値を自動的に削除する。

コンテンツ、属性、ファセット、またはサンプル値が含まれる派生型の基底型を切り替えると、基底型の修正ダイアログが表示されます（以下のスクリーンショットを参照）。



「**基底型の修正を行う前に確認**」チェック ボックスを無効にした場合、オプション ダイアログのビューのセクション（「ツール | オプション | ビュー[1466]」）にてこの設定を再度有効にすることができる旨のメッセージが表示されます。

オプション ダイアログのビューのセクション（「ツール | オプション | ビュー[1466]」）にあるスキーマ ビュー ペインでは、コンテンツを保持するか、または基底型の修正を行うたびにユーザーへの確認を行うかを指定することができます（以下のスクリーンショットを参照）。



コンテンツを保持して、確認を行うチェック ボックスを有効にすることで、これらの動作がデフォルトのオプションになります。

4.4.9 スマート制約

複合型に対して制約を課す場合、基底型に含まれるコンテンツ モデルの一部が派生型にて書き換えられることになります。派生型の編集を行いながらその基底型のコンテンツ モデルを正確に把握するのは大変な作業です。

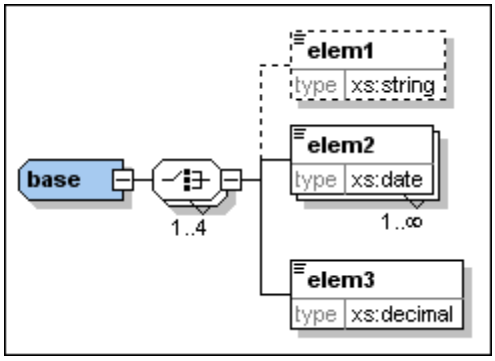
スマート制約により、（基底型と派生型の）2 つのコンテンツ モデルが派生した型のコンテンツ モデルに表示されます。派生した複合型では、基底複合型にあるすべての項目と、それらが派生した型とどのように関連付けられているのかを確認することができます。更にスマート制約では、基底型のコンテンツ モデルへ課すことのできるすべての制約が表示されます。これで派生した型を簡単かつ正確に制約することが可能になります。

以下の操作によりスマート制約を有効にすることができます:

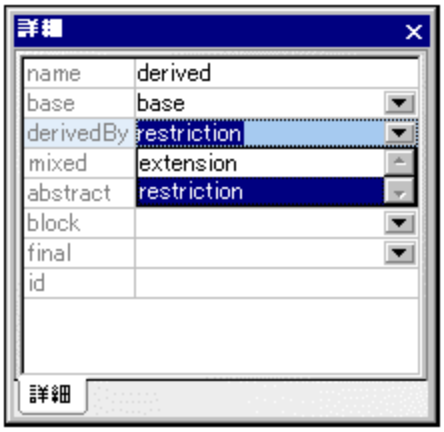
- スキーマ デザイン ツールバーにあるスマート制約アイコン をクリックします。

以下のサンプルはスマート制約の機能を説明しています。

この例では以下に示される複合型が基底型として使用されます

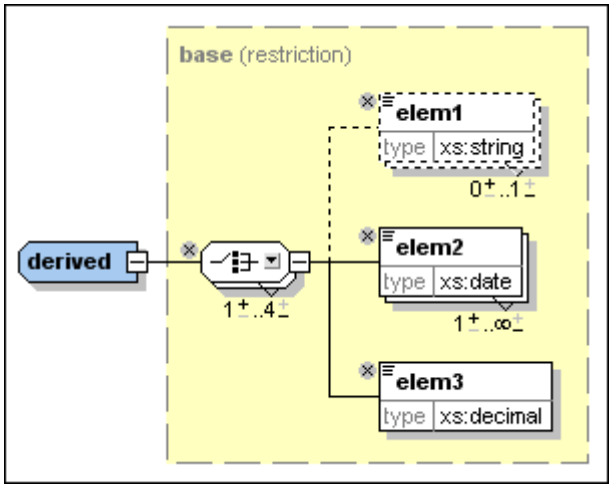


上に示される基底型のbase から、以下のように複合型のderived を派生させることができます:
1. スキーマにて derived という名前の複合型を新たに作成します。
2. 詳細入力ヘルパーにて base ドロップダウンリストから base を選択し、derivedBy ドロップダウンリストから restriction を選択します。



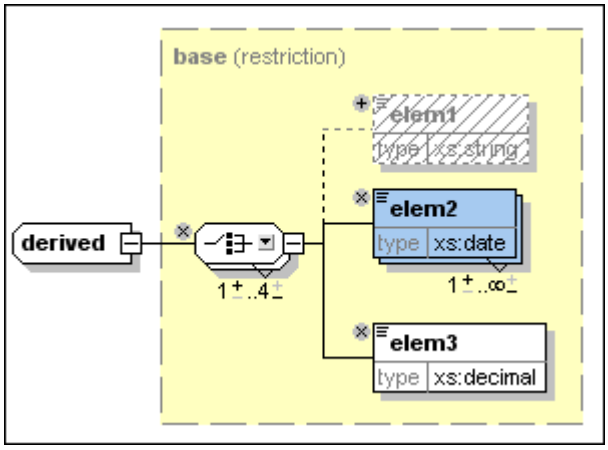
スマート制約が有効になっている状態で、派生型が以下のように表示されます

(C) 2015-2021 Altova GmbH

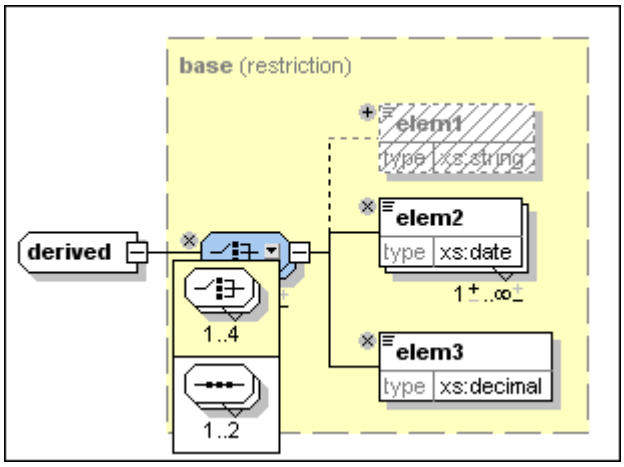


この例で派生型の制約に使用されるコントロールを以下に示します:

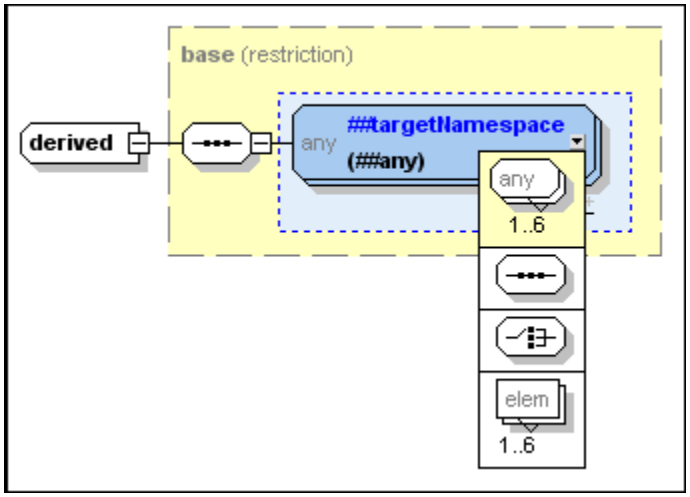
- このアイコン により、基底型にある要素を削除することができます。ここでは elem1 が削除されます。要素を追加するには、アイコン をクリックしてください。



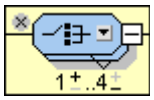
- 選択コンポジッターに表示された矢印 ▼ をクリックすることで、以下のスクリーンショットに示されるリストが表示されます。選択モデル グループをシーケンス モデル グループへ変更することができます。



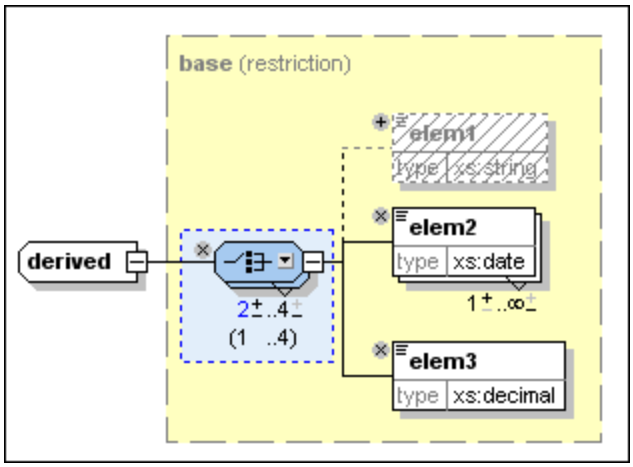
同様に、ワイルドカードを変更することもできます



どの項目をどの項目により置き換えることができるかの詳細については、XML スキーマの仕様 (英文) を参照ください。

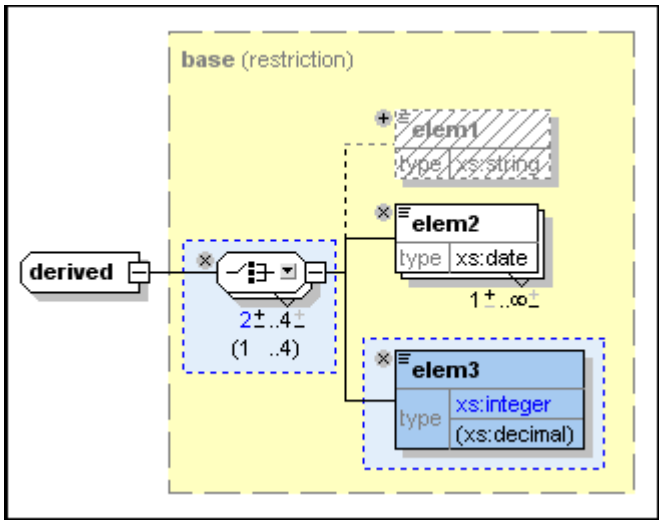
- モデル グループの出現回数は、  にある 1 の隣にあるプラス記号により最低出現回数を増やし、4 の隣にあるマイナス記号により最大出現回数を減らすことができます。基底型にて定義された出現の回数が特定の数 (例えば 4) ではなく範囲の場合 (例えば 2-5) の場合にこれらのコントロールが表示されます。出現の範囲が間違っていてもこれらのコントロールは表示されます。

(C) 2015-2021 Altova GmbH

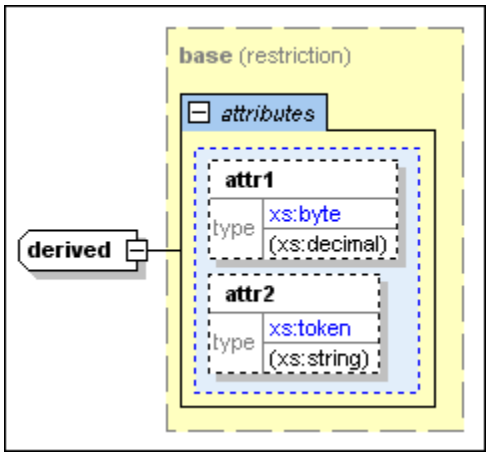


上のスクリーンショットでは、要素の最低出現回数が 2 に変更されています。モデル グループの背景が青色に変更され、基底複合型にあるモデル グループとは違うものであることを表している点に注目してください。基底型にて定義された出現範囲も表示されます。

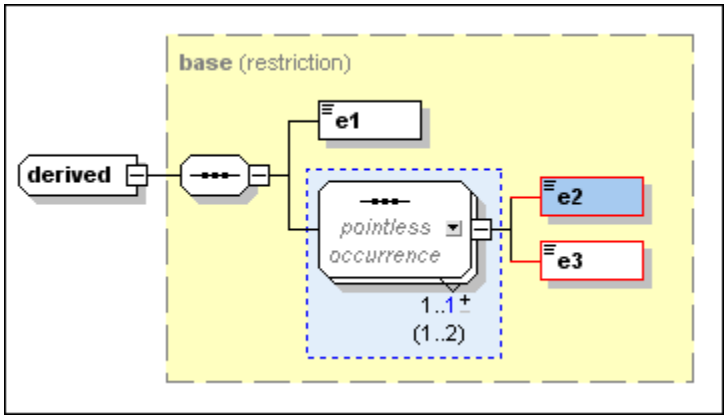
- それが XML スキーマの仕様で定義された基底型の制約に対して妥当な変換である場合、属性または要素のデータ型を変更することができます。例えば、derived 複合型にある elem3 データ型を decimal から integer へ変更することができます。これにより、要素の背景が青色に変わり、基底型のデータ型とは違っている点が示され、基底型で定義されている要素のデータ型がかっこの中に表示されます:



派生した複合型にて値のデータ型が制限された属性を以下のスクリーンショットに示します



- スマート制約はコンテンツ モデルで**意味のない出現**の警告を行います。意味のない出現とは、例えばコンテンツ モデルで表示されている、実際には必要ないシーケンスのことです。以下の図に意味のない出現の例を示します:



メモ: 意味のない出現は、コンテンツ モデルがエラーを含んでいる時にだけ表示されます。コンテンツ モデルが意味のない出現を含んでいながら妥当である場合もあり、その場合は混乱を避けるために意味のない出現は明示的に表示されません。

意味をなさない表示の詳細については、XML スキーマの仕様を参照ください。

4.4.10 xml:base、xml:id、xml:lang、xml:space

名前空間 `http://www.w3.org/XML/1998/namespace` はXML 名前区間の仕様により与えられたもので、`xml:` 接頭辞に紐付けられています。これにより、`xml:` 接頭辞とともにこの名前空間が使用されることになります。この名前空間では4 つの属性が定義されており、任意のXML ドキュメント（スキーマかインスタンスかを問わず）にあるすべてのXML 要素にて使用することができます:

- `xml:base`: 要素のベースURI をセットするのに使用
- `xml:id`: 要素のユニークなID を指定するのに使用
- `xml:lang`: その要素内にあるコンテンツが記述されている言語を識別するために使用
- `xml:space`: 要素内にある空白スペースをどのように処理するかを指定するために使用

スキーマビューにて、XML スキーマドキュメントへこのXML 名前空間がインポートされると、これら4 つのxml: 属性をスキーマ内にあるすべての要素にて使用することができます。

以下の操作により、これら属性のうち1 つを要素にて宣言することができます:

1. そのスキーマドキュメントに対するXML Namespace 名前空間を宣言し、名前空間をxml: 接頭辞に割り当てます。4 つあるxml: 属性のうちのどれかがドキュメントにて使用されると、その名前が正しい名前空間を含めるように展開されます。
2. XML Namespaces 名前空間をインポートします。XMLSpy のバリデーターが名前空間を認識し、4 つのxml: 属性がスキーマ内部から参照可能なグローバル属性で利用できるようになります。
3. 目的のxml: 属性を要素の子として挿入します。インポートされたグローバル属性への参照として属性は宣言されます。

XML Namespaces 名前空間の宣言

そのスキーマに対して宣言されているすべてのスキーマを確認ならびに編集することができるスキーマ設定ダイアログでは、XML Namespaces 名前空間(http://www.w3.org/XML/1998) を入力することができます。名前空間はxml: 接頭辞へ関連付ける必要があります(xml: 接頭辞とともに、テキスト ビューで表示されたxs:schema 要素にて名前空間を宣言することもできます)。

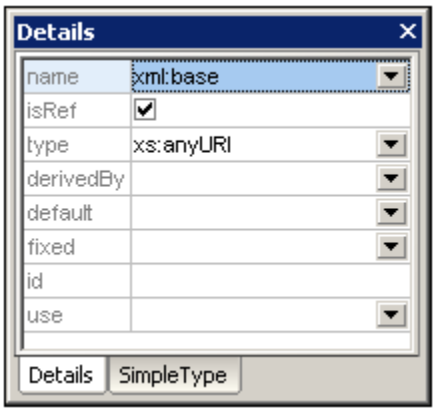
XML Namespaces 名前空間のインポート

スキーマ概要にてXML Namespaces 名前空間に対するグローバル要素宣言を作成します。スキーマ概要ウィンドウの上部にある挿入 または追加 アイコンをクリクし、メニューから「インポート」を選択します。XML Namespaces 名前空間をインポートする名前空間として入力します。テキスト ビューではインポート宣言が以下のように表示されているはずです:

```
<xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/XML/1998/namespace"/>.
```

xml: 属性の追加

スキーマ概要にてxml: 属性を追加する要素を選択し、その要素に対して属性を追加します。詳細ウィンドウにてname コンボボックスにある矢印ボタンをクリックし、目的のxml: 属性(例えばxml:base) を選択します(以下のスクリーンショットを参照)。グローバル属性を参照するか尋ねられた場合、「はい」を選択します。属性が参照として追加されます。



XInclude とxml:base

XInclude のinclude 要素がインクルード要素のhref 属性にて指定されたXML ファイルにより置き換えられた場合、解析されたXML ドキュメントの最上レベルにある要素にxml:base 属性が与えられます。このXML ドキュメントを検証する必要がある場合、xml:base 属性を関連する要素に対して定義する必要があります。

4.4.11 戻ると進む: 位置の移動

スキーマビューの「戻ると進む」コマンドにより、スキーマビューで直前に閲覧した位置に戻ることができます。スキーマビューにてスキーマコンポーネント間を移動しながら、既に見たコンポーネントを再度閲覧するのに便利な機能です。ツールバーにある「戻る」ボタンを一度クリックすることで、直前に閲覧した位置に移動します。「戻る」ボタンを繰り返し押すことで、既に閲覧したポジション（最大 500）まで移動することができます。既に閲覧した位置に移動した後、「進む」ボタンをクリックすることで先の位置に移動することができます。

これら 2 つのショートカット キーを以下に示します:

- 戻る: Alt + 左矢印
- 進む: Alt + 右矢印

戻る/進むと元に戻す/やり直し

戻ると進むコマンドは元に戻す（「Ctrl」 + Z）ややり直し（「Ctrl」 + Y）とお違うものだということご注意してください。「戻る」コマンドをクリックすることで、最後に見た状態のまま直前に閲覧したコンポーネントに戻ります。元に戻すコマンドを一度選択すると、最後に行った変更を（その変更がどこで行われたかに関わらず）取り消します。

追加メモ

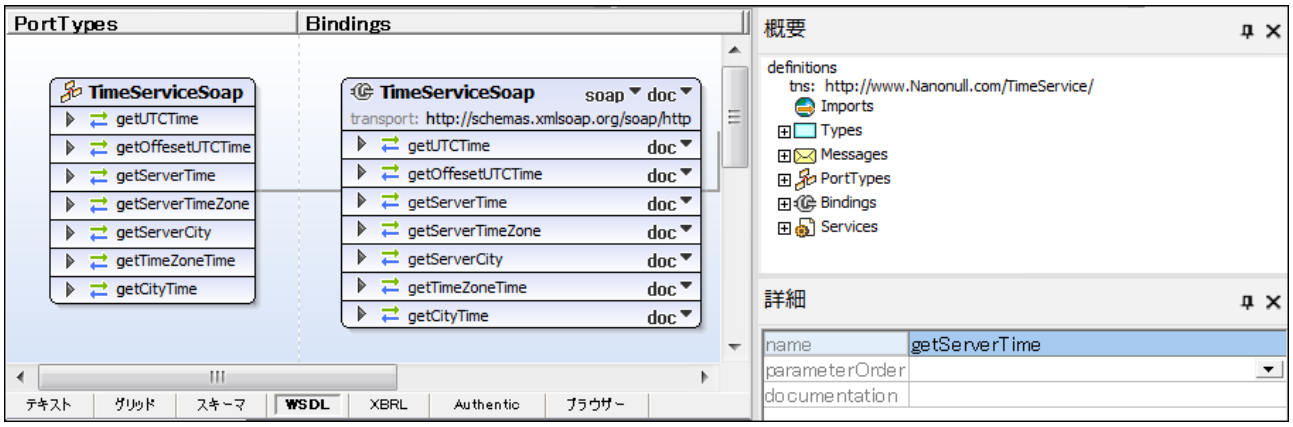
以下の点に注意してください:

- 「戻る」ボタンにより、最大 500 回まで最後に閲覧した場所に移動できます。
- 戻る/進む機能はスキーマ全体に対して有効になります。スキーマが閉じられ、他のビューで開かれていたとしても、スキーマビューにて再度開かれます。
- 前に閲覧したコンポーネントが削除されていた場合、コンポーネントを閲覧することはできません。そのようなコンポーネントが前の位置の一部であった場合、削除されていないコンポーネントとともに以前の画面が表示されます。削除されたコンポーネントが以前閲覧した位置のすべてを含んでいた場合、その位置を表示することはできず、「戻る」ボタンをクリックしても、削除されたコンポーネントのさらに 1 つ前の位置に移動します。

(C) 2015-2021 Altova GmbH

4.5 WSDL ビュー

WSDL ビュー(下のスクリーンショット)はWSDL 1.1 とWSDL 2.0 ドキュメントをグラフィカルに編集するインターフェースを提供します。WSDL ビューはWSDL ドキュメントがアクティブになっているときにWSDL ビュータブにて有効になります。グラフィカルなWSDL ドキュメントの構造とコンポーネントがメインウィンドウ[232]で作成され、入力ヘルパー[250]には特別な編集機能が有効になります。



メインウィンドウ[232](ポートタイプ(WSDL 1.1)またはインターフェース(WSDL 2.0)、バインディングとサービス)と入力ヘルパー(概要[236]と詳細[242])についてはこのセクションのサブセクションにて説明されます(プロジェクトに関する詳しい情報についてはリファレンスセクションのプロジェクトメニュー[1168]を参照ください)。

WSDL ビューで使用することのできる機能

WSDL ビューでは以下の機能を使用することのできます:

- メインウィンドウ内ですべてのWSDL 要素(ポートタイプ(WSDL 1.1)またはインターフェース(WSDL 2.0)、バインディング、サービス)のグラフィカルな表示
- ドラッグアンドドロップによるWSDL 要素の直接操作
- 動的なグラフィカルビューによる任意のWSDL 要素の追加、操作、削除機能
- 詳細入力ヘルパーによる値の入力と編集
- W3C Working Draft に従ったWSDL の検証機能
- 全てのWSDL 要素を、ポートタイプ(WSDL 1.1) またはインターフェース(WSDL 2.0)、バインディング、サービスによりグループ化されたかたちでメインウィンドウに表示
- ドラッグアンドドロップによりWSDL 要素を直接操作
- 参照またインポートしたスキーマが不正である場合のスキーマビューへの自動的な切り替え
- WSDL ビュー内からスキーマ型の編集
- MS Word またはHTML 形式でのWSDL ドキュメンテーション生成
- メインウィンドウのWSDL ドキュメントのダイアグラム(PNG イメージ)生成
- WSDL ウィンドウのビューを印刷

ファイル閲覧

ファイルの閲覧に関する以下の点に注意してください:

- WSDL ファイルを開いたときには、自動的にWSDL ビューで開かれます。
- WSDL ドキュメントをテキストビューや拡張グリッドビューで閲覧することも可能です。対応するタブをクリックしてください。
- WSDL ファイルがXML スキーマに対する参照を含んでいる場合、メニューコマンド「WSDL | 型 | スキーマビューでスキーマを編集」によりそのスキーマを閲覧、編集することができます。

- 関連付けられているスキーマファイルが開かれている場合、W SDL ファイルのビューを変更することはできません(例えば、W SDL ビューからテキストビューへの変更)。W SD Lファイルのビューを変更する前に、スキーマファイルを保存して閉じたことを確認してください。

W SDLドキュメント編集の助けになる2つの入力ヘルパーがあります: 概要[236]と詳細[242]入力ヘルパーです。両方の入力ヘルパーともタイトルバーをダブルクリックすることにより切り離しドッキングを行うことができます。ドッキングされているときには、タイトルバーにあるピンアイコンをクリックすることで自動的に隠す機能を有効にできます。自動的に隠す機能が有効になっていれば、入力ヘルパーはアプリケーションウィンドウの橋にあるタブとして最小化されます。隠れた入力ヘルパーはマウスポインターをタブに移動して、タイトルバーのピンアイコンを再度クリックすることで元の状態に戻すことができます。

関連項目: W SDLドキュメントの使用に関する詳細については、W SDL チュートリアル[684]ならびにユーザーリファレンスのW SDL メニュー[1367]を参照ください。

4.5.1 メインウィンドウ

メインウィンドウによりW SDLドキュメントの編集を行います。メインウィンドウには垂直の点線で分けられた3つのセクションが存在します: (i) ポートタイプ(W SDL 1.1)またはインターフェース(W SDL 2.0)[233]; (ii) バインディング[233]; (iii) サービス[235]。ポートタイプとバインディングの関係、またバインディングとサービスの関係はそれぞれ接続線で示されます。これら3つのセクションについては以下で説明します。

メインウィンドウのシンボル

以下のシンボルがメインウィンドウでは使用されます:

W SDL 1.1 ではポートタイプ、W SDL 2.0 ではインターフェース
バインディング
サービス
フォールト
 オペレーション。緑の矢印はインプットを表し、青の矢印はアウトプットを表します。オペレーションの種類により適切なシンボルが使用されます。
メッセージ
メッセージパート(パラメタ)
XSD 要素
XSD 単純型または複合型
ポート

新しいポートタイプ、インターフェース、バインディング、サービスを追加する

新しいポートタイプ(W SDL 1.1 ドキュメント)、インターフェース(W SDL 2.0 ドキュメント)、バインディング、またはサービスを追加するには、メインウィンドウのどこか(ただしコンポーネントボックスの外)を右クリックして、現れるコンテキストメニューから適切なコマンドを選択します。

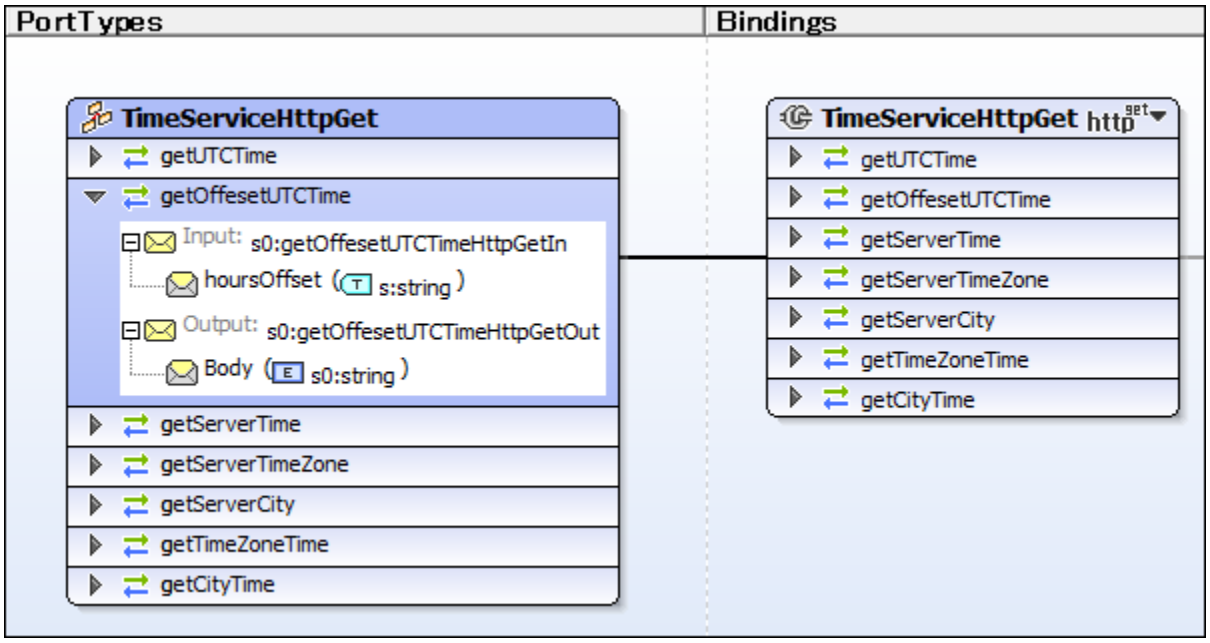
ドラッグアンドドロップ機能

以下のドラッグアンドドロップ機能が使用可能です:

- メインウィンドウにて、ドラッグアンドドロップによりポートタイプ(W SDL 1.1)またはインターフェース(W SDL 2.0)、バインディング、そしてサービス間の関係を記述することができます。
- W SDL 2.0 ドキュメントでは、概要入力ヘルパーの要素をメインウィンドウと概要入力ヘルパーのインターフェースfault にドラッグすることができます。

ポートタイプ(WSDL 1.1)、インターフェース(WSDL 2.0)

ポートタイプセクション(WSDL 1.1 ドキュメント)にはWSDLドキュメントで定義された全てのポートタイプが含まれます(下のスクリーンショットではポートタイプセクションのポートタイプ1つだけを表示している)。インターフェースセクション(WSDL 2.0 ドキュメント)にはWSDLドキュメントで定義された全てのインターフェースが含まれます。



各ポートタイプまたはインターフェースは定義されたオペレーション を含むボックス として表現されます。コンポーネントはボックス内で直接編集することができます。ポートタイプとインターフェースボックスの主な機能は以下の通りです:

- オペレーションはオペレーション名の左にある アイコンをクリックすることで、メッセージ を閲覧できるよう展開することができます。
- WSDL 1.1 では、メッセージにメッセージパート が含まれることがあります。このようなメッセージはメッセージパートを閲覧できるよう展開することができます。
- ポートタイプボックス(ポートタイプ、メッセージ、またはメッセージパート)のコンポーネントを右クリックすることで、コンテキストメニューが表示され、そこから関連するアクションを選択することができます。例えば、ポートタイプ名を右クリックすることで、新しいポートタイプや追加、選択したポートタイプに対してオペレーションを追加し、バインディングを作成することができます。
- 任意のWSDL 2.0 インターフェイスプロパティである *extends*、*styleDefault*、および *documentation* は、空の場合表示されません。インターフェイスのコンテキストメニュー内の「**編集**」コマンドにより編集することができます。
- WSDL 2.0 ドキュメントにて、オペレーションのプロパティーはオペレーションのコンテキストメニューの編集コマンドにより編集することができます。
- コンポーネントが選択されたとき、その詳細を詳細入力ヘルパーにて編集することができます。
- ポートタイプとインターフェースのドキュメンテーションは個々のボックスの下部に表示されます。

ポートタイプまたはインターフェースとバインディングの関連性はメインウィンドウに表示される(ポートタイプボックスまたはインターフェースボックスとバインディングボックスを繋ぐ)黒色の接続線で表されます。バインディングボックスはメインウィンドウのバインディングセクションに表示されます。

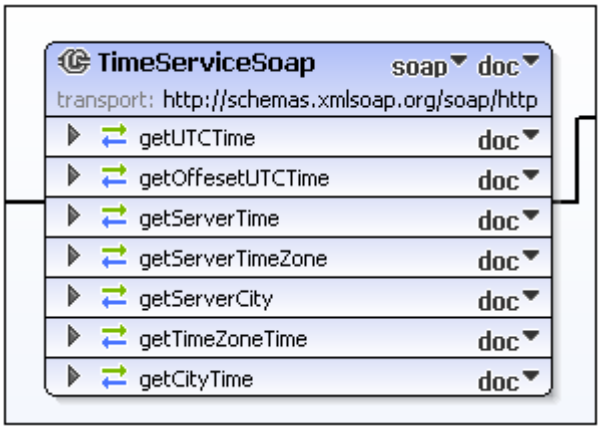
バインディング

バインディングは以下のメッセージフォーマットとプロトコルの詳細を定義します:

- 特定のポートタイプ(WSDL 1.1)で定義されたオペレーション、または

- 特定のインターフェース(WSDL 2.0)で定義されたオペレーション

WSDL 1.1 では、SOAP 1.1 またはSOAP 1.2 のエンドポイント、またはHTTP 1.1 のGET とPOST verb 属性に対してバインディングが作成されます。WSDL 2.0 において、バインディングはSOAP 1.1 またはSOAP 1.2 エンドポイント、またはHTTP に対して作成されます。各バインディングはメインウィンドウのバインディングセクションにあるバインディングボックスにより表示されます(下のスクリーンショットを参照)。バインディングボックスには全てのオペレーションと、関連したポートタイプまたはインターフェースが表示されます(下のスクリーンショットを参照)。



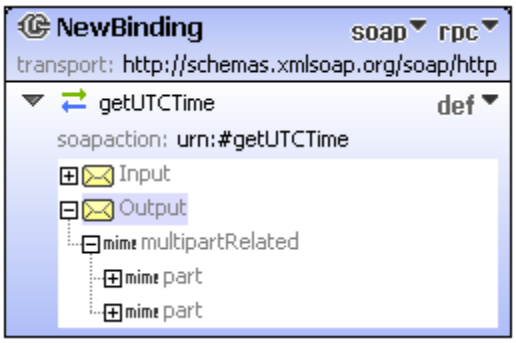
バインディングは以下のどれかの方法でポートタイプまたはインターフェースと関連付けられます:

- ポートタイプまたはインターフェースを右クリックして、「**バインディングのポートを作成**」または「**バインディングのインターフェースを作成**」を選択します。
- WSDL 1.1 バインディングを右クリックして、ポートタイププロパティーを編集します。
- WSDL 2.0 バインディングを右クリックして、「**編集 | インターフェース**」を選択します。

バインディングを定義するには、バインディング名の隣にあるコンボボックス(下のスクリーンショット)から必要なプロトコルを選択します。WSDL 1.1 では、soap 1.1、soap 1.2、http-get、またはhttp-post によりバインディングの種類を決定します。SOAP プロトコルを選択した場合、他にも(2 つ目のコンボボックスを使用して)スタイルを doc かrpc に設定できます。WSDL 2.0 ドキュメントの場合、wsoap:protocol プロパティーをバインディングのコンテキストメニューから「**編集**」コマンドにより追加することができます。

WSDL 1.1 では、MIME エンコーディング(MIME バインディング) がメッセージレベルで定義されます。MIME エンコーディングを定義するには、メッセージを右クリック(下のスクリーンショット)して、適切な MIME 定義を追加します。下のスクリーンショットでは、`Output` メッセージのMIME 宣言が作成されています。

(C) 2015-2021 Altova GmbH



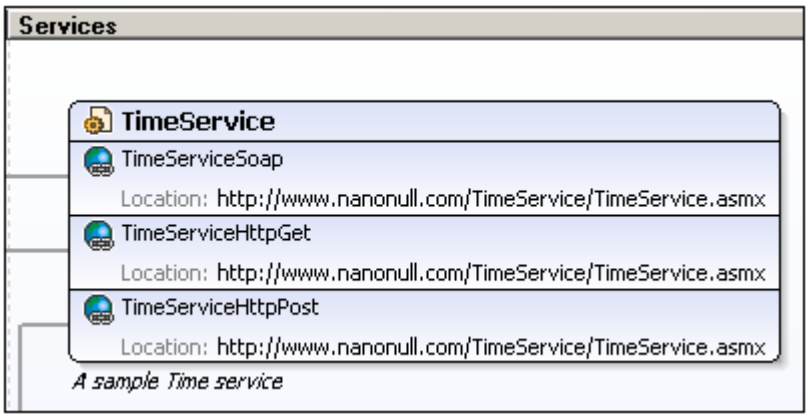
バインディングボックスにある特定のアイテムを右クリックしてコンテキストメニューを開きます。コンテキストメニューを使用して、例えば、バインディングの追加や削除、アイテムの定義や編集を行うことができます。バインディングボックスまたはその中のアイテムが選択されたとき、その定義が詳細入力ヘルパーに表示され、編集を行うことができるということにも注目してください。

バインディングのポートはバインディングボックスのタイトルバーを右クリックして、(WSDL 1.1 ドキュメントでは)「**バインディングのポートを作成**」コマンド、または(WSDL 2.0 ドキュメントでは)「**バインディングのエンドポイントを作成**」コマンドを選択することで行うことができます。関連するポートまたはエンドポイントが(メインウィンドウのサービスセクションにある)サービスボックスの中に作成されます。バインディングとポートの関連は黒色の接続線により表されます。

バインディングのドキュメンテーションは各バインディングボックスの下部に表示されます。

サービス

サービスグループでは複数のポート(WSDL 1.1)またはインターフェース(WSDL 2.0)のリレーションがまとめられます。メインウィンドウのサービスセクションにあるサービスボックス(下のスクリーンショット)により表示されます。1つのサービスボックスには1個以上のポートまたはエンドポイント宣言が存在します。



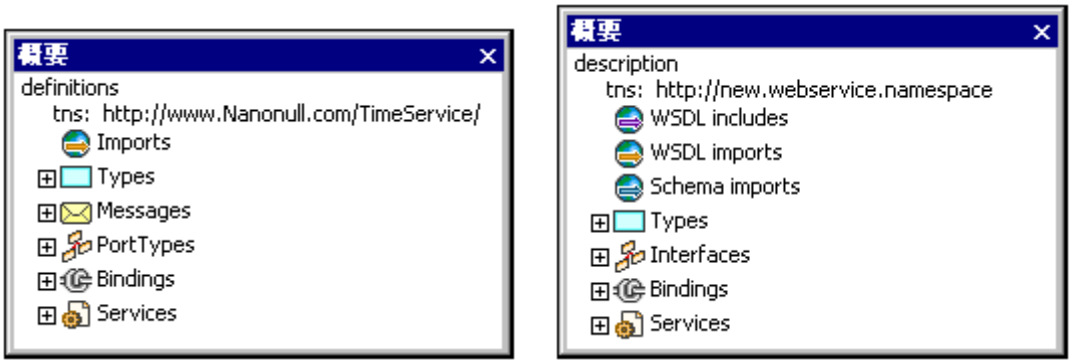
サービス名、ポートまたはエンドポイントの名前、ポートまたはエンドポイントに関連付けられたバインディング、そしてポートまたはエンドポイントのアドレス情報は詳細入力ヘルパーにて直接編集することができます。サービスボックスかその中にある特定のアイテムを右クリックして表示されるコンテキストメニューから、サービスもしくはそのアイテムに関係するコマンドを選択します。

サービスに関するドキュメンテーションは個々のサービスボックス下部に表示されます。

4.5.2 概要入力ヘルパー

概要入力ヘルパー(下のスクリーンショット)にはWSDLドキュメントの概要がコンポーネントがターゲット名前空間、インポートされたスキーマ、インクルード/インポートされたWSDLドキュメントごとにカテゴライズされて表示されます。ポートタイプ(WSDL 2.0 ではインターフェース)に加えて、メッセージ(WSDL 1.1)、バインディング、そしてサービスなどのドキュメントで定義された様々な型が表示されます。

概要入力ヘルパーでは、インポートならびにインクルードされたXMLスキーマやWSDLファイルを管理することもできます。



WSDL 1.1 (左) と WSDL 2.0 (右)内の概要入力ヘルパー

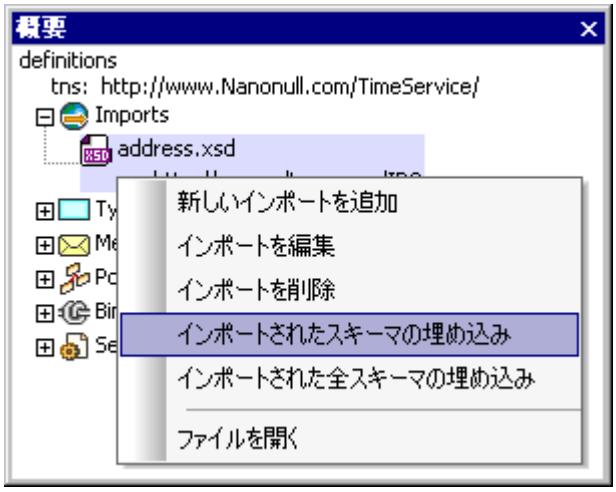
各カテゴリでは、コンポーネントがツリービューで表示されます。ツリーアイテムは展開したり折りたたんだりして、内容を表示したり隠すことができます。概要入力ヘルパーのコンポーネントを選択することで、プロパティーの編集を行うことができる詳細入力ヘルパー[242]にその情報が表示されます。ツリーに表示されているWSDLとスキーマコンポーネントの名前は直接編集することができます。外部で定義されたコンポーネント(インクルードやインポートされたWSDLドキュメントはスキーマでグレーで表示されています)を編集することはできません。概要入力ヘルパー内の個々のカテゴリについての説明は以下でなされます。

ターゲット名前空間(WSDL 1.1 ならびに2.0)

ツリーが**tns** で表示され、概要入力ヘルパーにて編集することができます。その他の名前空間は全てテキストビューで編集する必要があります。

Imports (WSDL 1.1)

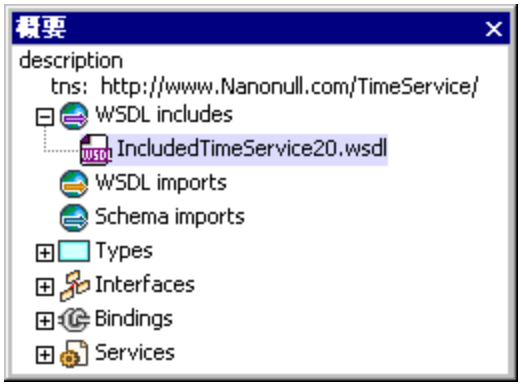
XMLスキーマ(XSD) ファイルならびにWSDLファイルをアクティブなWSDLドキュメントにインポートすることができます。XMLスキーマまたはWSDLファイルをインポートするには、Imports アイテム、またはImports リストに表示されている既にインポートされたアイテムを右クリックして、**「新しいインポートを追加」** を選択します。Imports リストにあるインポートされたファイルを右クリックすることでコンテキストメニューが表示され、新たなインポートを加え、選択されたファイルを置き換えるファイルを選択(**インポートを編集**)し、インポートされたファイルを削除することができます。ファイルをロケーションから開くこともできます。ファイルは(`.wsdl` ファイルの場合)WSDLビュー、または(`.xsd` ファイルの場合)スキーマビューで開かれ、そのビューで編集することができます。



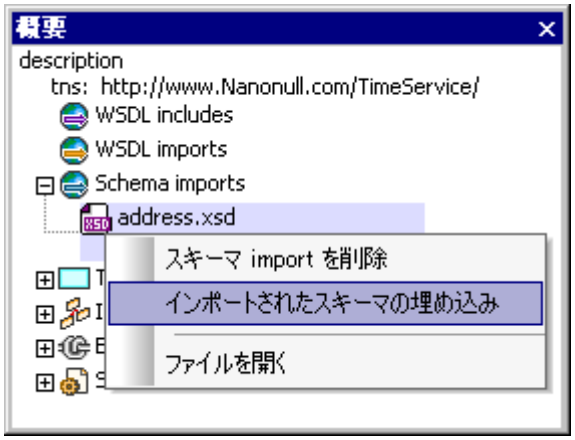
インポートされたXML スキーマはW SDL ファイルに埋め込むことができます。インポートされたスキーマを埋め込むと、そのスキーマが型 要素内のインラインスキーマとして作成され、`import` 要素が削除されます。インポートされたスキーマを埋め込むには、**Imports** リスト内にあるスキーマを右クリックして、「**インポートされたスキーマの埋め込み**」または「**インポートされた全スキーマの埋め込み**」コマンドを選択します。インポートされた全てのスキーマに対して適用される後者のコマンドは、**Imports** アイテムのコンテキストメニューからもアクセスすることができます。

WSDL includes, WSDL imports, Schema imports (WSDL 2.0)

XML スキーマ(XSD) ファイルをインポートし、W SDL ファイルをアクティブな W SDL ドキュメントにインポートすることができます。ファイルをインクルードまたはインポートするには、インクルードまたはインポートするアイテム(W SDL includes、W SDL imports、Schema imports)を右クリックして、インクルードまたはインポートを加えます。インポートされたファイルの名前空間はインポートされたファイルのターゲット名前空間から自動的に生成されます。



インクルードまたはインポートされたファイルを右クリックすることで表示されるコンテキストメニューから、ファイルを削除し XMLSpy で開いたりすることができます。ファイルはW SD Lビュー(`.wsdl` ファイルの場合)またはスキーマビュー(`.xsd` ファイルの場合)、そこで編集作業を行うことができます。その後、インポートされたXML スキーマをW SDL ファイルへ埋め込むことができます(以下のスクリーンショットを参照)。



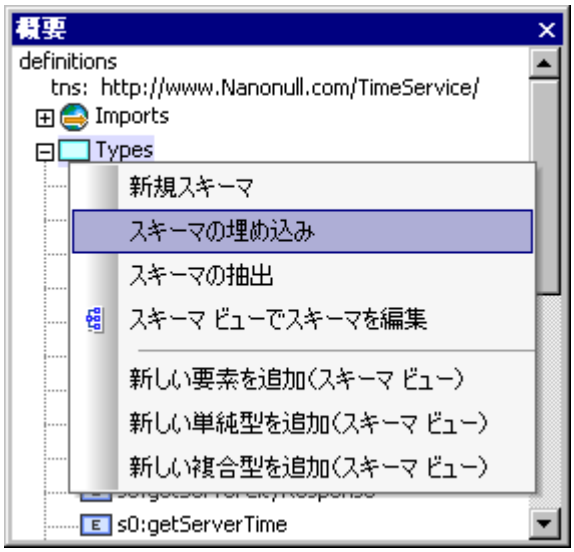
インポートされたスキーマを埋め込むと、そのスキーマが `types` 要素内のインラインスキーマとして作成され、`import` 要素が削除されます。インポートされたスキーマを埋め込むには、**Imports** リスト内にあるスキーマを右クリックして、「**インポートされたスキーマの埋め込み**」または「**インポートされた全スキーマの埋め込み**」コマンドを選択します。インポートされた全てのスキーマに対して適用される後者のコマンドは、**Imports** アイテムのコンテキストメニューからもアクセスすることができます。

Types (WSDL 1.1 および 2.0)

WSDL ドキュメントで定義された全ての型を黒色で表示します。インポートされたスキーマや WSDL ドキュメント内で定義された型はグレーで表示されます。

以下の関数を利用することができます:

- **新規スキーマの作成**: **Types** アイテムを右クリックして、「**新規スキーマ**」を選択します(以下のスクリーンショットを参照)。WSDL ファイル内に埋め込まれた空のスキーマが新たに作成され、埋め込まれたスキーマとともにスキーマビューに表示されます。スキーマドキュメントの編集を完了した後に変更をスキーマファイルへ保存することで、変更が WSDL ドキュメントに書き込まれます。WSDL ドキュメントに埋め込まれたスキーマを新たに作成する際にこの機能を利用することができます。



- **スキーマの埋め込み**: **Types** アイテムを右クリックして、「**スキーマの埋め込み**」コマンドを選択します。埋め込みを行うファイルを選択するためのダイアログが表示されます。「OK」をクリックすることで、スキーマが `types` 要素内にインラインスキーマとして作成されます。選択されたスキーマが既にインポートされている場合、既にインポートされているスキーマを埋め込むか選択するように促されます。

- **スキーマの抽出**: **Types** アイテムを右クリックして「**スキーマの抽出**」を選択します。埋め込まれた各スキーマファイルが一時ファイルとしてスキーマビューにて開かれ、各々のファイルに対して名前を付けて保存ダイアログが表示されます。スキーマファイルを保存する場合、スキーマが抽出され、指定された場所に保存された後でWSDL ファイルへインポートされます。インラインスキーマとして処理されていたスキーマファイルは削除され、インポートされた外部スキーマとして処理されます。

- **スキーマビューでスキーマを編集**: 埋め込まれたスキーマをスキーマビューにて編集することができます。Types アイテムまたは Types リスト内にあるスキーマコンポーネントの名前を右クリックして、**スキーマビューで「スキーマを編集」**または**「編集」**(**スキーマビュー**)を選択します。WSDL ドキュメントの `types` 定義から一時的な XSD ファイルが生成されます。この XSD ドキュメントはスキーマビューにて編集することができます。XSD ドキュメントの編集後にファイルを保存することで、変更点がWSDL ドキュメント内にある `types` の定義へ保存されます。変更点を保存すること無く XSD ドキュメントを閉じた場合、WSDL ドキュメントが修正されることはありません。

- **スキーマコンポーネントの追加**: XML スキーマ要素(WSDL 1.1 ならびに 2.0)、単純型(WSDL 1.1)、または複合型(WSDL 1.1)を追加することができます。**Types** アイテムまたは Types リスト内にあるスキーマコンポーネントの名前を右クリックして、**追加**コマンドを選択します。WSDL ドキュメントの `types` 定義から一時的な XSD ファイルが生成され、スキーマビューにて表示されます。このファイルには新規コンポーネントが名前の無い状態で含まれています。このコンポーネントを編集し、ファイルを保存することで、新規コンポーネントがWSDL ドキュメントの `types` 定義へ書き込まれます。

- **スキーマコンポーネントの削除**: **Types** リスト内にあるスキーマコンポーネントを右クリックして、コンテキストメニューから「**削除**」を選択することで、スキーマコンポーネントを削除することができます。

メッセージ(WSDL 1.1)

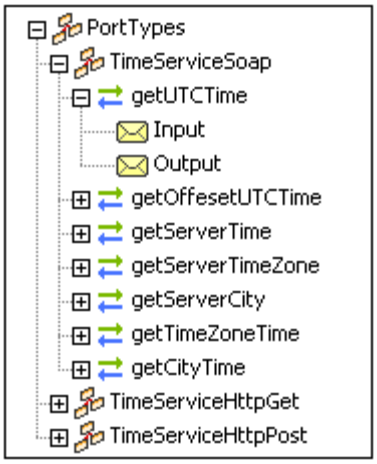
メッセージまたはそのサブコンポーネントが選択されたとき、そのプロパティーの編集を詳細入力ヘルパー[242]にて行うことができます。その他にも、コンテキストメニューを使用して以下のことを行えます:

- 概要入力ヘルパーにてメッセージが選択されている場合、メッセージパートの追加や削除、またはメッセージの追加を行うことができます。
- 概要入力ヘルパーにてメッセージパートが選択されている場合、そのメッセージパートを削除し、メッセージに新たなメッセージパートを追加しすることができます。

- 「同期」コマンドにより選択されたメッセージまたはメッセージパートに対応するポートタイプボックスをハイライトすることができます。

PortType (WSDL 1.1)

コンテキストメニューからポートタイプに関する以下の機能を使用することができます:

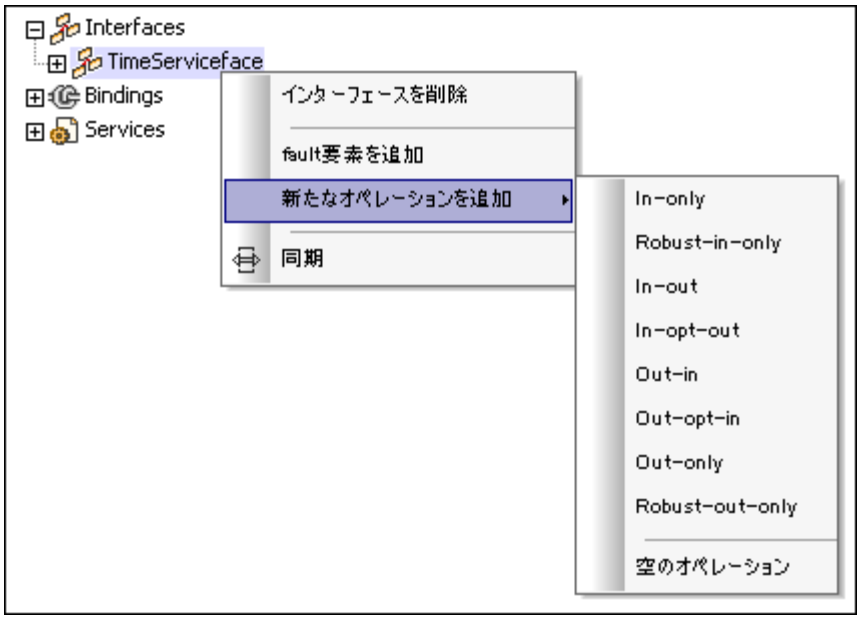


- PortTypes アイテムが選択された状態で、ポートタイプを追加することができます。
- ポートタイプが選択された状態で、ポートタイプを追加し、選択されたポートタイプを削除し、選択されたポートタイプにオペレーションを追加することができます。
- オペレーションが選択された状態で、新たなオペレーションを追加し、選択されたオペレーションを削除し、要素(入力、output、またはfault)を選択されたオペレーションに追加することができます。
- メッセージ要素(入力、output、またはfault)が選択された状態で、新たなメッセージを追加し、選択されたメッセージを削除することができます。
- 「同期」コマンドにより、選択されたポートタイプ、オペレーション、メッセージがハイライトされます。

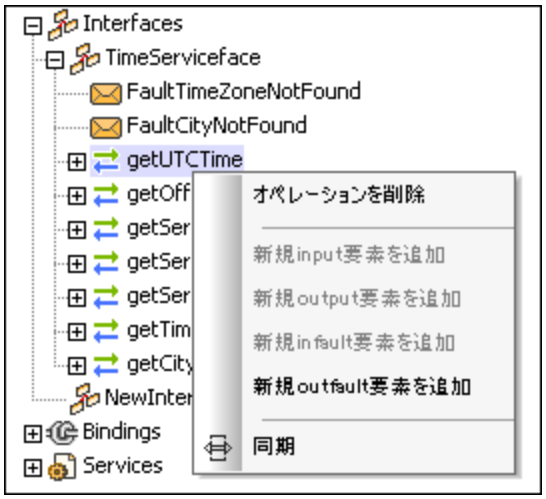
Interfaces (WSDL 2.0)

インターフェースの管理はコンテキストメニューから行うことができます。

- インターフェースを追加するには、*Interfaces* アイテムを右クリックして、メニューコマンドから「**新たなインターフェースを追加**」を選択します。
- インターフェースを右クリックすることで、(以下のスクリーンショットにあるような)ポップアップメニューが表示され、選択されたインターフェースを削除し、選択されたインターフェースにfault やオペレーションを追加することができます。追加されるオペレーションの種類は「**新たなオペレーションを追加**」コマンドのサブメニューから選択することができます。対応するバインディングオペレーションも全て、Baindings 内に追加されます。同じように、オペレーションが削除された際にも、それを参照しているバインディングオペレーションも削除されます。



- オペレーションを右クリックすることで、(以下に表示される)コマンドが表れ、選択されたオペレーションを削除し、(`infault` や `outfault` といった)要素をオペレーションに追加することができます。



- メッセージ要素を右クリックすることで、選択されたメッセージを削除するポップアップメニューが表示されます。
- 「同期」コマンドにより、デザイン内で選択されたインターフェース、オペレーション、またはメッセージがハイライトされます。

Bindings (WSDL 1.1 ならびに2.0)

バインディングが選択された状態から、新たなバインディングを既存のバインディングに追加し、選択されたバインディングを削除し、選択されたバインディングにオペレーションを追加することができます。オペレーションやメッセージが選択された状態からは、PortTypes (WSDL 1.1) やInterfaces (WSDL 2.0) カテゴリー内のオペレーションやメッセージに対して利用可能なオプションを利用することができます。「同期」コマンドをクリックすることにより、選択されたバインディング、オペレーション、そしてメッセージがハイライトされます。

Services (WSDL 1.1 ならびに2.0)

サービスが選択された状態から新たなサービスを追加し、選択されたサービスを削除し、選択されたサービスにポートを追加することができます。「同期」コマンドをクリックすることにより、選択されたサービスやポートがハイライトされます。

4.5.3 詳細入力ヘルパー

詳細入力ヘルパーではメインウィンドウまたは概要入力ヘルパーで選択されたアイテムのプロパティが表示されます(以下のスクリーンショットを参照)。これらプロパティは詳細入力ヘルパーで編集することができます。

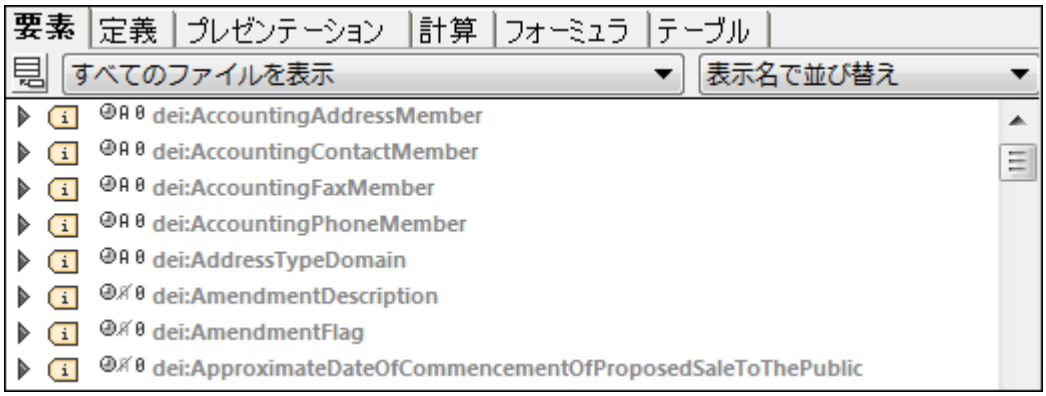
例えば上のスクリーンショットで示される様に、メインウィンドウ内で(`TimeService` サービスの) `TimeServiceSoap` ポートが選択されている場合、`TimeServiceSoap` のプロパティが詳細入力ヘルパーにて表示され、編集することができます。`name` や `description` のテキストフィールドを編集するには、フィールドをダブルクリックして、テキストの編集を行ってください。上のスクリーンショットの例にある `binding` のようなプロパティの場合、コンボボックスをクリックして、現れた選択肢の中からオプションを選ぶことができます。

4.6 XBRL ビュー

XBRL ビューはXBRL タクソノミの編集を行うことができるグラフィカルなビューです。素早い検証とエラーを発見した際にはその旨を表示し、タクソノミの開発を素早く正確に行うことができます。

XBRL ビューには以下のパーツが含まれています:

- メインウィンドウには6 つのタブが表示されます: 要素[243]、定義[243]、表示[247]、計算[247]、フォーミュラ[247]、タブ[247]（以下のスクリーンショットを参照）。これらのタブの詳細については、この項以下のサブセクションを参照ください。



- 3つのパワフルな入力ヘルパー[250]: 概要、グローバル要素、詳細。これら入力ヘルパーにより、タクソノミファイルの管理やメインウィンドウでの編集が行ないやすくなります。
- メッセージウィンドウではタクソノミの妥当性に関するメッセージが表示されます。

このセクションではXBRL ビューのメインウィンドウならびに入力ヘルパーに関する記述と、それらの使用する方法について記述します。より詳しい情報を知るには、XBRL[721] とXBRL メニュー[1391]にあるコマンドの解説を参照ください。

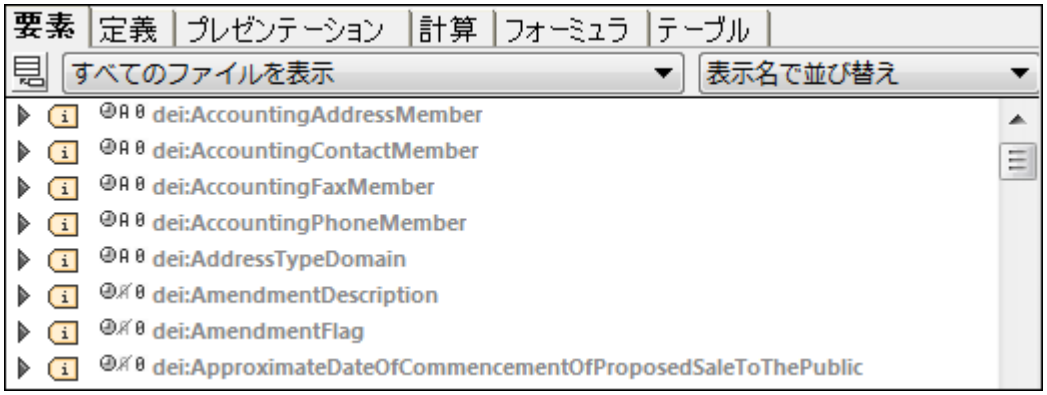
XBRL ビューにあるその他の機能

このセクションで記述される編集機能以外にも、以下の便利な機能を使用することができます:

- ドキュメントの生成[1398]によりHTML、W ord、またはRTF フォーマットの詳細なドキュメントを作成することができます。
- XMLSpy の「ファイル | 印刷」コマンドにより、現在使用しているビューの印刷[1144]を行うことができます。

4.6.1 メインウィンドウ : 要素タブ

以下のスクリーンショットに示されるメインウィンドウの「要素」タブには、タクソノミのコンセプトが（デフォルトでは）インポートされたタクソノミ内のコンセプトと共に表示されます。現在のタクソノミ内のコンセプトが黒色で表示され、インポートされたタクソノミのコンセプトは灰色で表示されます。



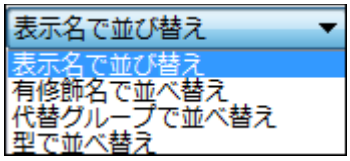
詳細に関しては、 XBRL[721] のセクションとXBRL メニュー[1391]内のコマンドの説明を参照してください。

表示するタクソノミを選択

メインウィンドウタブの下にある *ファイル* コンボボックス(以下のスクリーンショットを参照)を使えば、現在のタクソノミにあるコンセプトだけを表示するのか、それともインポートされたタクソノミ内にあるコンセプトも一緒に表示するのか選択することができます。インポートされた数多くのタクソノミを表示させないだけでも編集スピードが格段に向上します。

要素の並び替え

メインウィンドウのタブ以下の右側にある *ソート* コンボボックス(以下のスクリーンショット参照)では、メインウィンドウ内に表示されている要素の並び替えを行うことができます。



並べ替えの条件は以下から選ぶことができます:

- *表示名で並べ替え*: 要素は名前のアルファベット順にリストされます。名前にはプレフィックスが含まれます。ですから `abc:yname` は `bcd:xname` の前に発生しません。
- *修飾名で並べ替え*: 要素名は完全に解決されます(これは名前空間が展開されていることを意味します)、そして、展開された名前がアルファベット順にリストされています。ですから、`abc` 名前空間プレフィックスは `http:people.altova.com` にバインドされ、`abc:yname` は、`http:people.altova.com:yname` に対して解決されます。
- *代替グループ*: 代替グループは4つあります: 項目、タプル、*hypercubeItems*、*demensionItems*。
- *型で並べ替え* : XBRL 要素の Type 属性を参照します。

メインウィンドウ内での要素の検索

メインウィンドウ内で要素を検索するには、「Ctrl」+ F のキーコンビネーションを使用します。検索ダイアログが表示されるので、目的の文字列を入力して検索を行うことができます。タクソノミで使用されている名前空間プレフィックスも検索されます。

コンセプト（<element> 要素）について

各タクソノミコンセプトは XML スキーマの <element> 要素にて定義されています（以下を参照）。この定義はタクソノミドキュメントをテキストビュー[130]で見ることにより確認することができます。（インポートされたタクソノミドキュメントの定義を見るには、そのドキュメントを一度 XMLSpy で開かなければなりません）

```
<xs:element id="icui_UnrealizedHoldingLoss"
      name="UnrealizedHoldingLoss"
      substitutionGroup="xbrli:item"
      type="xbrli:monetaryItemType"
      xbrli:balance="credit"
      xbrli:periodType="instant"
      abstract="false"
      nillable="true"/>
```

要素タブに表示される各要素（またはコンセプト）は、その代替グループを意味するアイコン（item、tuple、hypercube、またはディメンション）と共に表示されます。更に、コンセプトの `balance`、`periodType`、`abstract`、そして `nillable` 属性の値を意味するアイコンがコンセプト名の左に表示されます（以下のスクリーンショットを参照）。

コンセプトの名前を変更するには、コンセプト名をダブルクリックして名前の編集を行ないます。

代替グループ

コンセプトの代替グループ値はコンセプトのアイコンで表示されます：

`xbrli:item`

`xbrli:tuple`

`xbrldt:hypercubeItem`

`xbrldt:dimensionItem`

以下のスクリーンショットは `substitutionGroup` 値が `xbrli:item` の要素を表しています。

balance、periodType、abstract、nillable 属性

要素名の左にあるアイコン（上のスクリーンショットを参照）にはコンセプトのメイン属性が左から右の順序で表示されます：

- `xbrli:balance`：＋のアイコンが資産（`credit`）、－のアイコンが負債（`debit`）を表します
- `xbrli:periodType`：グレーで表示された時計のアイコンは期間（`duration`）を表し、白の時計アイコンはインスタント（`instant`）を表します

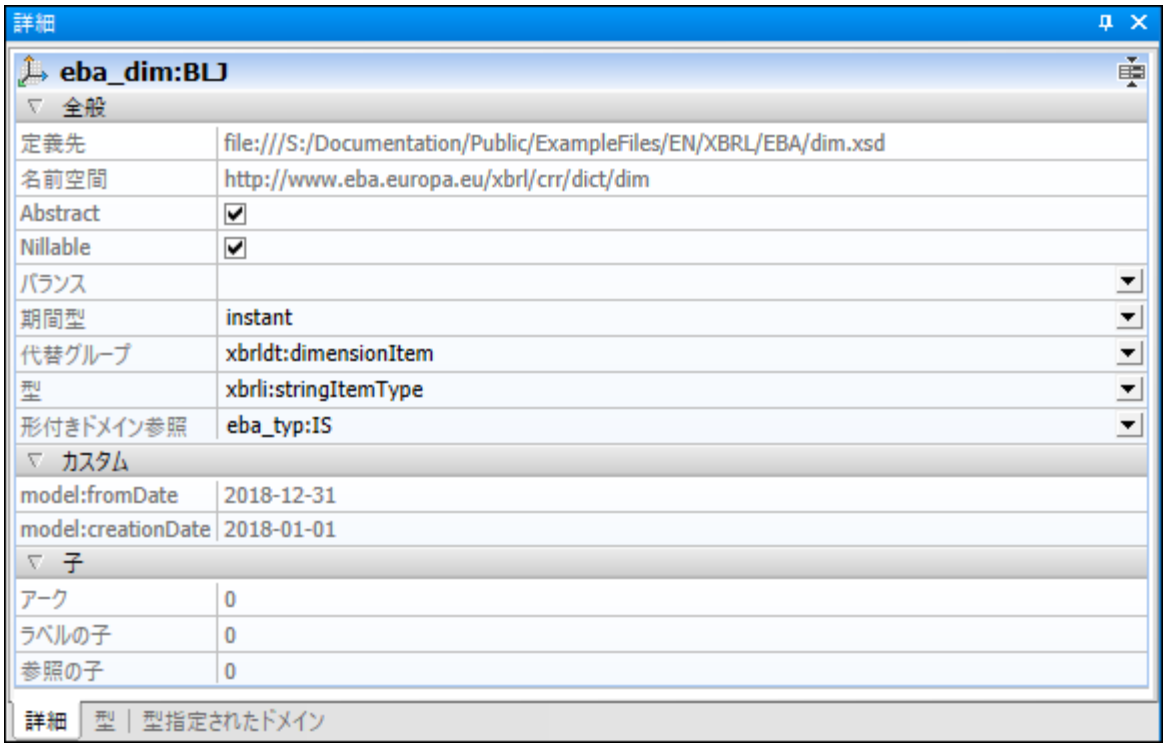
- xs:abstract：黒色のA アイコンはtrue で、グレーはfalse を表します
- xs:nillable：黒色の0 アイコンはtrue で、グレーはfalse を表します

上に記されたxbrli: 属性はXBRL スキーマからのもので、XML スキーマのxs: 属性とは異なるものということご注意してください。

上のスクリーンショットでは、＋ アイコンによりxbrli:balance 属性の値がcredit（貸方）であることが示されます。スクリーンショットに示されるその他の値（xbrli:periodType、xs:abstract、xs:nillable）は、それぞれduration、false（abstract ではない）、true（nillable である）を表しています。

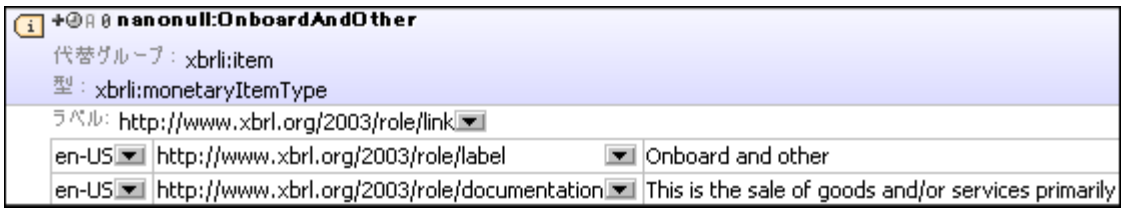
これら4つのアイコンのどれかにマウスカーソルを合わせると、4つ全ての属性値が表示されます。アイコンをクリックすることで、その属性に許されている値と共にリストボックスが表示され、マウスクリックにより選択を行うことで、コンセプト属性の値を素早く変更することができます。

これらの属性値は詳細入力ヘルパーでも編集することができます。



属性 substitutionGroup とtype

要素名の左にある矢印シンボルをクリックすることで、要素の表示が展開され、xs:substitutionGroup とxs:type の2つの要素が画面に表示されます（以下のスクリーンショットを参照）。



上のスクリーンショットでは、xs:substitutionGroup と xs:type 属性の値はそれぞれ xbrli:item と xbrli:monetaryItemType となっています。これらの値は対応するコンボボックスのドロップダウンリストから別の値を選択することで編集可能です。詳細入力ヘルパー内の対応する値をダブルクリックし、キーボードから値を入力することで、両方の属性の値を変更することができます。

ラベルリンク

ラベルの子を加えるには、要素を右クリックし、コンテキストメニューから「**ラベルリンクロールの追加**」を選択します。これでラベル行が追加(以下のスクリーンショットを参照)され、ラベルプロパティ(言語、ラベルの種類、そして値)を追加、指定することができます。ラベルの展開や縮退は、ボックスの右端にある+または−ボタンをクリックすることで行えます。

各ラベルの言語とラベルの種類プロパティはコンボボックスから値を選択することで編集することができます(上のスクリーンショットを参照)。

参照リンク

参照の子を加えるには、要素を右クリックして、コンテキストメニューから「**参照リンクロールの追加**」を選択します。これで参照ボックスが作成され、参照フィールドをクリックすることで参照プロパティの値を編集することができます。参照の展開や縮退は、ボックスの右端にある+または−ボタンをクリックすることで行えます。

4.6.2 メインウィンドウ: 定義、プレゼンテーション、計算、フォーミュラ、テーブルタブ

メインウィンドウには コンセプト間のリレーションシップを表す3つのタブが含まれています:

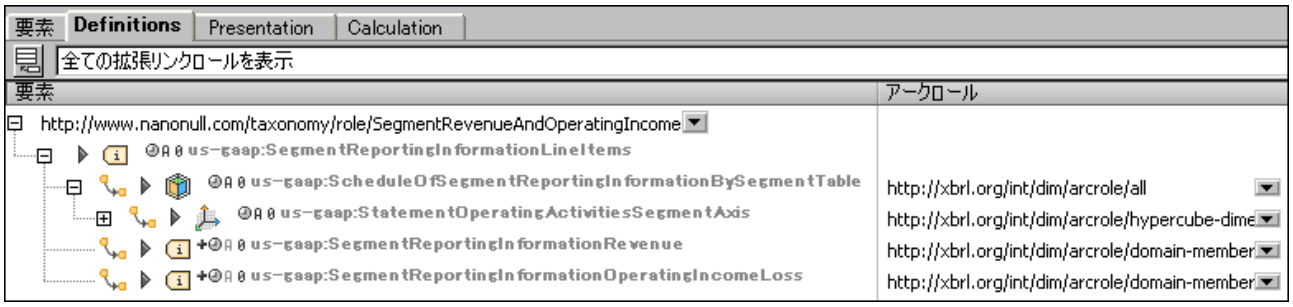
- 定義 タブにて表示される 定義 リレーションシップ
- プレゼンテーション タブにて表示される プレゼンテーション リレーションシップ
- 計算 タブにて表示される 計算 リレーションシップ
- フォーミュラタブにて表示されるフォーミュラの定義とリレーションシップ
- テーブルタブにて表示されるテーブルの定義

各タブ(定義、プレゼンテーション、計算、フォーミュラ、テーブル)では、対応するタクソノミリレーションシップが表示され、リレーションシップをグラフィカルに修正することができます。2つのコンセプト間のリレーションシップは、あるコンセプトから別のコンセプトへアークを構築することで作成されます。この *from-to* アークは曲がった矢印として表示されます。2つのコンセプト間のリレーションシップは(from から to への方向と共に)アークロールとして指定されます。リレーションシップの to-end にて要素が選択されているときにアークのアークロールが詳細入力ヘルパーに表示され、定義 リレーションシップの場合、定義 タブのアークロール列に表示されます。

リレーションシップアークは、全てのリレーションシップ(定義、プレゼンテーション、計算)において同一に表示されます。このセクションでは 定義 タブにおいてリレーションシップがどのように表示されるかの簡単な説明と共に、各タブが個別に扱われます。プレゼンテーション ならびに 計算 アークに関するより詳細な情報は、各セクションを参照ください。リレーションシップに関するより詳細な情報は、リレーションシップの作成: パート1 757 ならびに リレーションシップの作成: パート2 760 を参照ください。

定義 タブ

定義 タブではタクソノミの定義全てが表示されます。これらの定義は 定義 リレーションシップ（.xml）ファイル内に収められている 定義 アークにより指定されます。XBRL ビューの 定義 タブには 定義 アークのセットから得られた構造が展開/縮退可能なツリー形式で表示されます（以下のスクリーンショットを参照）。



このグラフィカルな表示では、2 つのエンドポイント（*from* エンドポイントと *to* エンドポイント）を持つ曲がった矢印として各 定義 アークが表示されます。2 つの要素のリレーションシップの種類は要素の*アークロール*列に表示されます。例えば、`hypercube-dimension` リレーションシップでは、リレーションシップ（またはアークロール）は対となる要素の ディメンション パートと一緒に表示されます。アークロールURI は詳細入力ヘルパーからも入力できます。

定義 リレーションシップに関する詳細は、リレーションシップの作成: パート 1 757 を参照ください。

プレゼンテーション タブ

プレゼンテーション アークには 定義 アークと同一の属性があり、同様に機能します（上の 定義 タブの項を参照ください）。アークロールURI は詳細入力ヘルパーにて入力することができます。重要な プレゼンテーション 属性の1つに親要素から派生した子供の要素が表示される順序を決定する `order` 属性があります。プレゼンテーション タブではこれら子供要素は正しい順序で表示されます。子要素をリスト内の他の場所にドラッグすることで `order` 属性の値を素早く変更することができます。詳細入力ヘルパーを使っても `order` 属性の値を変更することができます（以下のスクリーンショットを参照）。

プレゼンテーション リレーションシップに関する詳細は、リレーションシップの作成: パート 2 760 を参照ください。

計算 タブ

計算アークも 定義 アークと同様の属性を持ち、同じように機能します（上の 定義 タブの項を参照ください）。アークロールURI は詳細入力ヘルパーを使って入力することができます。アークロールには以下の2種類があります:

- 要素の値を合計して他の要素に受け渡すアークロール
- 合計リレーションシップではなく、等価リレーションシップとして機能するアークロール

前者の場合、`weight` 属性により、どれだけ要素の値が集計要素に合計されるかが決定されます。`1.0` という属性値により、要素の値がそのまま合計されます。負の数値により、値が合計から差し引かれます。`weight` 属性の値は 計算タブまたは詳細入力ヘルパーから修正することができます。

フォーミュラ タブ

XBRL フォーミュラは フォーミュラ タブで定義および管理することができます。フォーミュラ タブは概要入力ヘルパーおよび詳細入力ヘルパーと共にフォーミュラを作成および編集することができます。フォーミュラ間の定義とリレーションシップはダイアグラムで実行することができます。詳細はXBRL フォーミュラ エディター 767 のセクションを参照してください。

テーブルタブ

テーブルはXBRL タクソノミ内で定義されている概念の表示の代替法を提供します。テーブルタブ内で作成し編集することのできるこれらの定義はテーブルリンクベースに含まれています。テーブルリンクベースの編集の詳細に関しては、XBRL テーブル定義エディター[791]を参照してください。

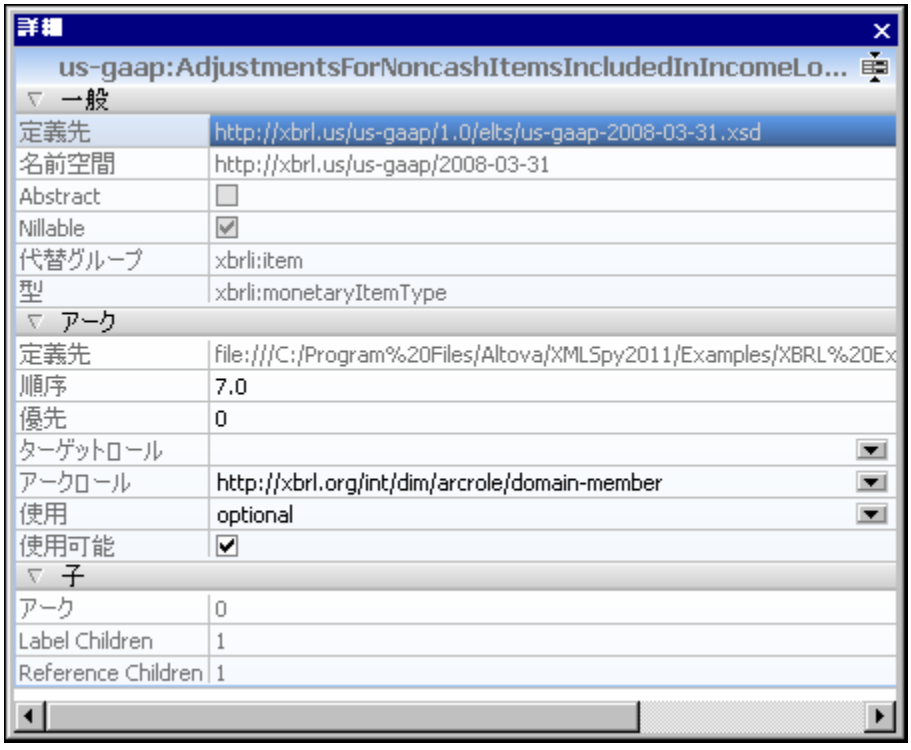
グラフィカルなビューを使った編集

以下の様な編集機能が利用可能です:

- グローバル要素入力ヘルパーから要素をドラッグして、ツリーにある要素にドラッグしてリレーションシップを作成することができます。
- アークロールの作成や編集は、(アークロール列にある) アークロールを選択することで行えます。アークロールにて定義されたリレーションシップは `from-to` リレーションシップで、選択された要素がリレーションシップの `to-endpoint` となります。
- 要素をドラッグすることでツリー内の位置を変更することができます。これにより `order` 属性の値が変更されます。
- 要素のプロパティの編集を行うには、そのプロパティシンボルをクリックした後行うか、プロパティボックスを展開した後で、ボックス内部のプロパティを編集することができます。

詳細入力ヘルパーでの編集作業

メインウィンドウにて要素が選択されていれば、定義 アークのプロパティが詳細入力ヘルパーのアークセクションに表示されます(以下のスクリーンショットを参照)。値フィールドをダブルクリックして目的の値を入力するか、(それができるのであれば) コンボボックスにて表示されるドロップダウンリストの値を選択することでこれらプロパティの値の編集を行うことができます。



その他にも、選択されている要素のアークのプロパティが詳細入力ヘルパーのアークセクションに表示されます。現在アクティブになっているタブに従って、表示されるアークは 定義 アーク、プレゼンテーション アーク、または 計算アークとなります。アークロールはアークロールフィールドにて入力することができます。

ラベルならびに参照リファレンスは子セクション以下に表示されます。これらのリレーションシップは要素タブにおいて編集できます。

追加情報に関しては XBRL[721] セクションを参照してください。

4.6.3 XBRL ビューの入力ヘルパー

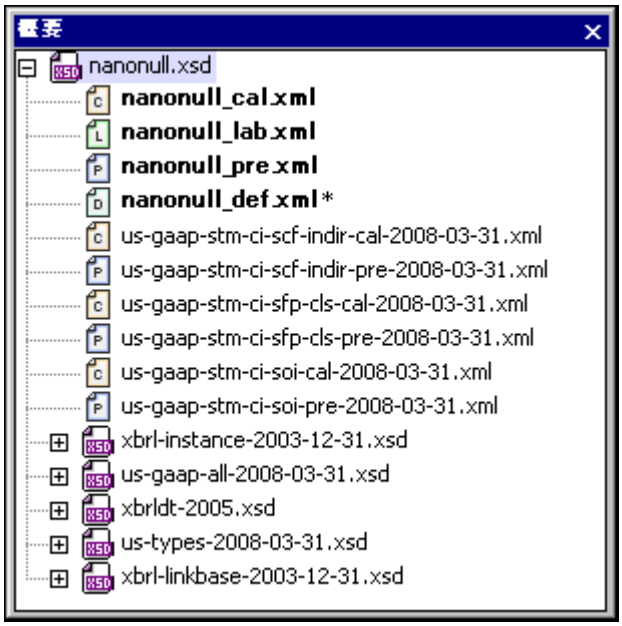
XBRL ビューには以下の入力ヘルパーが含まれています:

- 概要[250] 入力ヘルパー
- グローバル要素[251] 入力ヘルパー
- 詳細[253] 入力ヘルパー

検証情報はメッセージウィンドウ[112] に表示されます。詳細に関しては XBRL[721] セクションを参照してください。

概要入力ヘルパー

概要入力ヘルパーにはタクソノミファイル[739] がツリー構造で表示されます(以下のスクリーンショットを参照)。



ツリーは以下のように管理されます:

- メインコンセプトファイル(XML スキーマドキュメント)はツリーのルートに表示される現在アクティブなファイルです。
- リレーションシップリンクベースファイル(XML ファイル)はリレーションシップの種類を表す以下のような文字と色により表示されるファイルです。

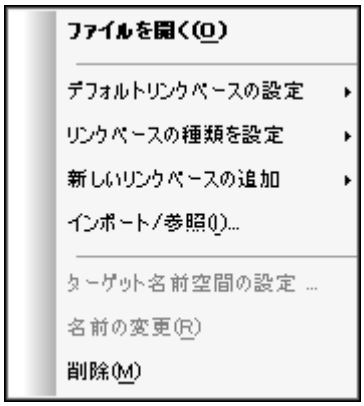
  は定義リンクベースを表します。
  はラベルリンクベースを表します。
  は計算リンクベースを表します。
  は表示リンクベースを表します。
  は参照リンクベースを表します。

はフォーミュラリンクベースを表します。

- インポートされたスキーマは階層の下部に表示されます。XML スキーマ(.xsd)は全てXSD アイコンで表示されます。

概要入力ヘルパーに表示される上の情報はメインコンセプトファイルの /schema/annotation/appinfo 要素から取得されます。より詳細な情報については[タクソノミファイル](739)を参照ください。

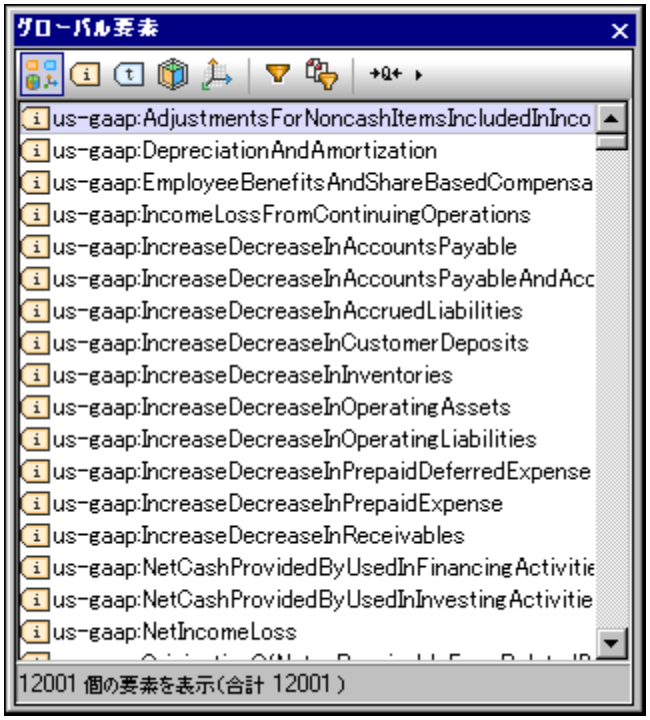
概要入力ヘルパーにあるファイルを右クリックすると(以下のスクリーンショットに表示される)コンテキストメニューが表示され、以下のようなコマンドが使用できます:



- *ファイルを開く* : 選択されたXML スキーマまたはXML ファイルをXMLSpy で開きます。
- *デフォルトリンクベースの設定* : 単一のリレーションシップ種類(例えばプレゼンテーションリレーションシップ)に対して複数のファイルが存在するとき、これらのうち1つをデフォルトリンクベースとしてセットすることができます。新たに作成されたリレーションシップは、そのリレーションシップのデフォルトリンクベース内に保存されます。リレーションシップのデフォルトリンクベースは太字で表示されます。
- *リンクベースの種類を設定* : 選択されたリンクベースのリレーションシップの種類を指定します。目的のリレーションシップの種類をサブメニューから選択してください。すべてにより、選択されたリンクベースは全てのリレーションシップの種類において使用されるようになります。
- *新しいリンクベースの追加* : 5つあるリレーションシップ(定義、presentation、calculation、ラベル、resource)のうちの1つを含むリンクベースファイルを新たに作成します。追加されたファイル名は右クリックして表示されるコンテキストメニューの「名前の変更」コマンドにより変更することができます。新たに追加されたリンクベースを参照するlinkbaseRef 要素がコンセプトファイルに新たに追加されます。
- *インポート/参照* : 標準タクソノミをインポートするか、既存のXML スキーマまたはリンクベースに対する参照を作成します。標準タクソノミオプションを選択した場合、標準タクソノミのリストがダイアログに表示されます。インポートしたいタクソノミを選択してください(詳細は[タクソノミのインポート](748)を参照ください)。XML スキーマまたはリンクベースへの参照を作成した場合、コンセプトファイルにlinkbaseRef 要素が新たに作成されます。XML スキーマまたはリンクベースオプションをクリックすると、目的のファイルを参照するためのダイアログが表示されます。
- *ターゲット名前空間の設定* : ターゲット名前空間をセットし、xs:schema スキーマ要素(つまりタクソノミ全体)の中にその名前空間を加えます。「[XBRL | ターゲット名前空間の設定](1395)」コマンドの詳細を参照ください。
- *名前の変更* : 選択されたファイルの名前を変更します。
- *削除* : 概要にて選択されたファイルを削除し、コンセプトファイルからlinkbaseRef 要素を削除します。

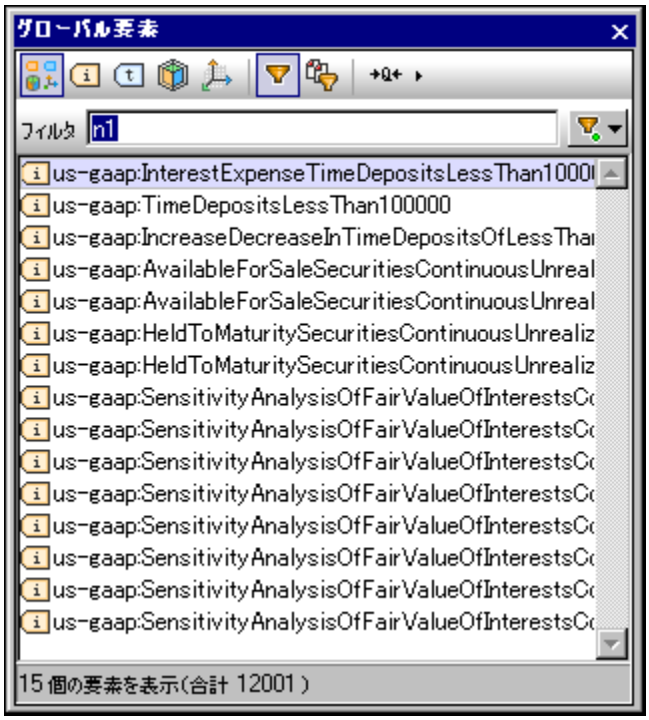
グローバル要素入力ヘルパー

(以下のスクリーンショットで示される)グローバル要素入力ヘルパーにはタクソノミ内にある全てのアイテム、Tuple、ハイパーキューブ、そして次元が表示されます。グローバル要素入力ヘルパーからメインウインドウへ要素をドラッグすることもできます。



グローバル要素入力ヘルパーでは以下の機能を利用することができます:

- *種類によるフィルタリング*: ツールバーにある各アイコンをクリックすることで、グローバル要素入力ヘルパーにて表示される種類(アイテム、タプル、ハイパーキューブ、次元)を切り替えることができます。*全ての要素を表示*アイコン により、全ての要素を表示します。選択されているフィルターがハイライトされます。
- *テキストによるフィルタリング*: テキストフィルターアイコン により、テキストフィルターが表示されます(以下のスクリーンショットを参照)。

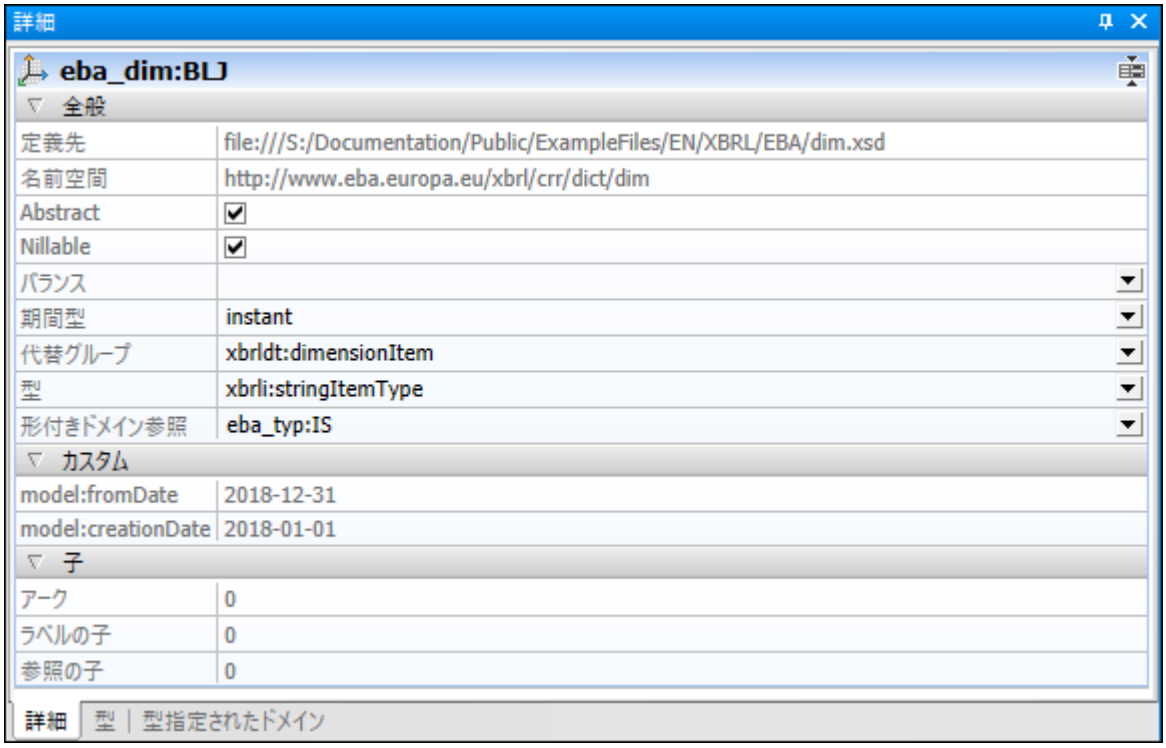


テキストフィルターが表示された後、テキストフィルターバーの右側にあるコンボボックスを使ってフィルタリングを行う対象を選択することができます。選択することのできるオプションには *含む*、*含まない*、*開始文字* があります。上のスクリーンショットの例では、*含む* オプションが選択されています。次に、フィルタリングを行う文字列をテキストフィルターボックスに入力します。入力された値に従って、フィルタリングされた結果が表示されます。上の例では、`n1` を含む要素のリストが表示されています。
フィルタリングではエントリ名をテキストとして扱うのと、エントリ名はフォーマットアイコン(下記参照)で変更することができ、フィルタリングの結果もそれによって変化することにご注意してください。

- *ソースによるフィルタリング*: 目的の*ソース*をポップアップから選択し、選択された*ソース*に存在する要素だけを表示することができます。
- *フォーマット* : 要素名が表示される形式には3つあります: *短い有修飾名*、*展開有修飾名*、*ラベル*。短い有修飾名では、対応する名前空間に割り当てられた*プレフィックス*が使用されます。展開有修飾名(アイコン:)では名前空間全体が使用されます。ラベルでは、要素に関連付けられたラベルが使用されます。フィルタリングは、現在表示されている要素のリストから、フィルタリングテキストにマッチしたものを表示するものなので、フォーマットの選択によりテキストによるフィルタリングオプションの結果が変わることにご注意してください。
- *ドラッグアンドドロップ機能* : グローバル要素入力ヘルパーにある要素を(定義、Presentation、Calculation といった*リレーションシップビュー*の種類に関わらず)*メインウィンドウのリレーションシップ*にドラッグすることができます。

詳細入力ヘルパー

メインウィンドウにて要素が選択されている場合、(以下のスクリーンショットにある)詳細入力ヘルパーには、そのプロパティが表示されます。要素にカスタム属性が存在する場合、*カスタム* という名前のセクションが入力ヘルパー内に表示され、属性を含んでいます。



要素のプロパティによっては詳細入力ヘルパーにて編集を行うことができます：*Abstract*、*Nillable*、*バランス*、*期間型*、*代替グループ*、そして*型*。

4.6.4 XBRL ビューの設定

XBRL ビューのために構成することのできる設定には2つの種類があります:

- XBRL ビュー内のコンポーネントのフォントと色。これらの設定はオプションダイアログ（「ツール | オプション」[1478]）のフォントと色のセクションによりアクセスすることができます。
- ビューのレイアウトとフォーマットおよびラベルのデフォルト。これらの設定は「XBRL | XBRL ビュー設定」コマンドによりアクセスすることのできる XBRL ビュー設定ダイアログ[1401] 内で使用することができます。

4.7 Authentic ビュー

Authentic View には、ウィンドウの上部にメニューバーとツールバーがあり、残りのインターフェイスは以下の3つの部分から構成されています: プロジェクトウィンドウ、メインウィンドウ、入力ヘルパー。これらの部分は以下に表示されています。

Authentic ビューインターフェイス[540] の詳細は Authentic[526] | Authentic ビューインターフェイス[540] のセクションで説明されています。

4.8 ブラウザービュー

ブラウザービューは、通常以下を閲覧するために使用されます:

- 関連したXSLT ファイルをもつXML ファイル。ブラウザービューに切り替えると、XML ファイルは、関連したXSLT スタイルシートを使用して素早く変換され、結果は直接ブラウザービューに表示されます。
- HTML ファイルは、HTML として直接作成されるか、またはXML ファイルのXSLT 変換 を介して作成されます。

ブラウザービュー内のXML およびHTML ファイルを閲覧するには、**ブラウザー** タブをクリックします。

Microsoft Internet Explorer およびXSLT に関する注意点

ブラウザービューは、またMicrosoft Internet Explorer 5.0 または以降を必要とします。XSLT スタイルシートにより変換されたXML ファイルを閲覧するためにブラウザービューを使用する場合は、XSLT 1.0 標準を完全にサポートするXML パーサであるMSXML 3.0 をサポートするInternet Explorer 6.0 またはは以降を使用することが奨励されます。MSXML 4.0.のインストールを行う場合は、(IE 5 内のXSLT へのサポートは、公式のXSLT 勧告と100% 互換性があるわけではありません。)IE 5 を使用して、ブラウザービュー内で問題が発生した場合、IE 6 またはは以降にアップグレードしてください。使用中のInternet Explorer のバージョンのXSLT に対するサポートを確認してください。

ブラウザービュー機能

ブラウザービューで以下の機能を使用することができます。[ブラウザー](#) 1365 メニュー、[ファイル](#) 1128 メニュー、および[編集](#) 1148 を使用してアクセスすることができます。

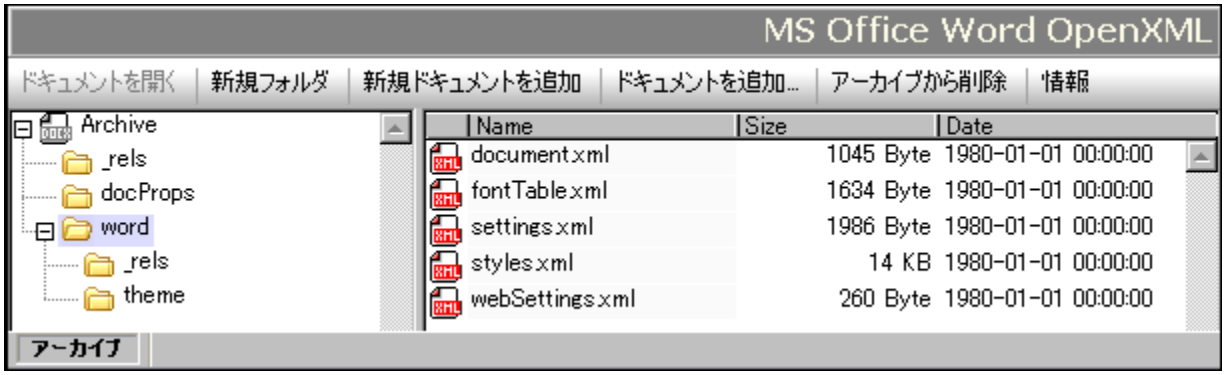
- *個別のウィンドウで開く:* ブラウザービューが個別のウィンドウである場合、同じドキュメントの編集ビューで横に並べて表示することができます。これを行うにはメニューコマンド [**ブラウザー | 個別のウィンドウ**] をクリックします。これは2 つのウィンドウ間のブラウザービューを切り替えるトグルです: (i) 個別のウィンドウと(ii) メインウィンドウ内のタブ付けされたビュー。これらのコマンドは(メインウィンドウの下の) **ブラウザービュー** ボタンのドロップダウンメニューで使用することができます。オプションダイアログの[ビュータブ](#) 1466 によりブラウザービューがデフォルトで個別のウィンドウで表示されるかを設定することができます。
- *「前へ」と「戻る」:* コマンド ブラウザーコマンド は、ブラウザービュー内でロードされたページをナビゲートするために使用されます。これらのコマンドは**ブラウザー** メニュー内にあります。
- *フォントのサイズ:* **ブラウザー** メニューを介して、調節することができます。
- *停止、更新、印刷:* 標準 ブラウザーコマンドは、**ブラウザー** と**ファイル** メニュー内にあります。
- *検索:* テキスト文字列のために検索を有効化します。これらのコマンドは**編集** メニュー内にあります。

4.9 アーカイブビュー

Archive View ではOffice Open XML（OOXML）ファイルやZIP ファイル、またはEPUB ファイルを開くことができます。OOXML やZIP ファイル、EPUB ファイルの構造を変更するだけでなく、Archive View ではアーカイブ内にある個々のファイルを開き、XMLSpy の編集ビューにより編集を行ない、アーカイブ内に直接保存しなおすことができます。

アーカイブファイルとアーカイブビュー

（OOXML やZIP、EPUB ファイルといった）アーカイブファイルがXMLSpy で作成または開かれたとき[836]、ファイルは（以下のスクリーンショットにある）Archive View にて開かれます。アーカイブファイルは同時に複数個開くことができ、各アーカイブファイルが個別のArchive View ウィンドウにて表示されます。アーカイブファイルの種類はArchive View の右上に表示されます。以下のスクリーンショットに表示されているアーカイブファイルの種類はMicrosoft Office W ord Open XML です。



フォルダービュー

フォルダービューはArchive View ウィンドウの左側に配置され、圧縮されたアーカイブのフォルダー構造を表示します。各階層においてフォルダーはアルファベット順で表示されます。サブフォルダーを参照するにはフォルダーの左にある+シンボルをクリックしてください。フォルダーの横に+シンボルが無い場合、サブフォルダーは存在しません。フォルダー内にあるドキュメントファイルを閲覧するには、フォルダーを選択することで、メインウィンドウにファイルが表示されます。上のスクリーンショットでは、メインウィンドウにて表示されているドキュメントは`word` フォルダー内部のもので、`_rels` と`theme` の2つのサブフォルダーを含んでいます。

メインウィンドウ

メインウィンドウにはフォルダービューで選択されているフォルダー内に含まれるドキュメントが表示されます。ドキュメントはアルファベット順で表示され、圧縮していない状態でのファイルサイズと更新日時が表示されます。Archive View で表示されているドキュメントをダブルクリックすることで開くことができます。ドキュメントは別のXMLSpy ウィンドウにて開かれます。

コマンドボタン

コマンドボタンはArchive View ウィンドウの上部に位置しています。

- *ドキュメントを開く*：メインウィンドウ内でドキュメントが選択されているときに有効になります。クリックすることで選択されたドキュメントが開かれます。メインウィンドウに表示されているドキュメントをダブルクリックすることでもドキュメントを開くことができます。
- *新規フォルダー*：フォルダービューにて現在選択されているフォルダーに新たなフォルダーを追加します。フォルダービューにてフォルダーが作成されたら、すぐに名前を付けなければなりません。新しいフォルダーは、アーカイブファイルが保存されたときにアーカイブに保存されます。
- *新規ドキュメントを追加*：フォルダービューにて現在選択されているフォルダーに新たなドキュメントを追加します。このボタンをクリックすることでXMLSpy の新規ドキュメントを作成ダイアログが表示されます。新しく作成されたドキュメントは別のXMLSpy ウィ

ンドウにて表示されます。フォルダービューにてファイルが選択されたら、すぐにそのファイルの名前を付けなければなりません。アーカイブファイルが保存されたとき、または編集ウィンドウにてドキュメントが保存された時に作成されたドキュメントは保存されます。

- *ドキュメントを追加...* : 参照ダイアログを起動し、追加するドキュメントを選択することができます。メインウィンドウで現在選択されているフォルダーにドキュメントが追加され、別のXMLSpy ウィンドウにてドキュメントが開かれます。アーカイブ内のドキュメントを保存するには、ドキュメントの編集ウィンドウから保存を行うか、アーカイブファイルを保存する必要があります。
- *アーカイブから削除* : メインウィンドウ、またはフォルダービューにて選択されたドキュメントやフォルダーをアーカイブから削除します。アーカイブファイルが保存されるまで、変更は有効になりません。
- *情報* : 情報ウィンドウの表示を切り替えます。以下を参照ください。

情報ウィンドウ

情報ボタンをクリックすることで、情報ウィンドウの表示を切り替えることができます。情報ウィンドウでは、アーカイブに含まれるファイルの数や、圧縮時と非圧縮時のファイルサイズ、また圧縮率といったアーカイブファイルに関する一般的な情報が表示されます。

4.10　共通のショートカット

一般的に使用される編集コマンドのデフォルトのショートカットは以下にリストされています。カスタマイズダイアログのキーボードタブ[1443]内でデフォルトのショートカットを変更することができます。

関数キーのショートカット（検証と変換のためのショートカットも含む）機能キーのショートカット

F1	ヘルプメニュー
F1 + Alt	最後に開かれたファイルを開く
F3	次を検索
F4 + CTRL	アクティブなウィンドウを閉じる
F4 + Alt	XMLSpyを閉じる
F5	更新
F6 + CTRL	開かれているウィンドウを循環する
F7	整形式チェック
F8	検証
F10	XSL 変換
F10 + CTRL	XSL:FO 変換

ファイルとアプリケーションコマンド

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

その他のキー

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

編集コマンド

CTRL + A	すべて選択
CTRL + F	検索

CTRL + G	ライン/文字に移動する
CTRL + H	置き換える
CTRL + V	貼り付け
CTRL + X	切り取り
CTRL + Y	やり直し
CTRL + Z	元に戻す

(C) 2015-2021 Altova GmbH

5 XML

このセクションではXMLSpy にてXMLドキュメントを操作する方法が記述されています。XMLに関する以下のような側面がカバーされます:

- XMLドキュメントを作成、開く、保存する方法[262]。このセクションではファイルの作成に関するXMLSpy の重要な設定についても説明されます。
- XMLドキュメントはテキストビュー[266]、グリッドビュー[268]、またはAuthentic ビュー[299]にて編集することができます。使いやすいビューを選択して、編集時にビューの切り替えを行うことができます。各ビューではそれぞれのビューが持つ利点を活かすことができます。
- XML フラグメント[304] をXMLドキュメントに外部フォルダーから簡単に素早く追加することができます。
- XMLドキュメントの入力ヘルパー[301]が持つ独自の機能に関する解説。
- XSLT やXQuery を含むXMLドキュメントの操作方法[306]。これらの処理を行うのに関連したXMLSpy が持つ様々な機能が説明されます。PDF フォント[308]に関するセクションでは、PDF 出力を生成する際のフォントの処理方法について説明されます。
- XMLドキュメントで作業をする際に使用できるその他の機能[381]を紹介します。

Altova W eb サイト: XML エディター

5.1 XMLドキュメントを作成、開く、保存する

XMLドキュメントを作成、開く、そして保存するときには、以下のことがらが関連します:

- テキストビュー、グリッドビュー、Authenticビューのうち、どのビューでXMLドキュメントは開かれているのか
- 新規XMLドキュメントが作成されたとき、スキーマ(XMLスキーマまたはDTD)が自動的に、もしくは手動で割り当てられるか、または割り当てられないのか
- XMLドキュメントにスキーマが割り当てられている場合、ドキュメントを開き保存するときに検証は自動的に行われるのか

デフォルトビュー

アプリケーション全体で有効になる設定により、(新規または既存に関わらず)XMLファイルが開かれるビューを指定することができます。これらの設定はオプションダイアログ「**ツール | オプション**」に収められています。

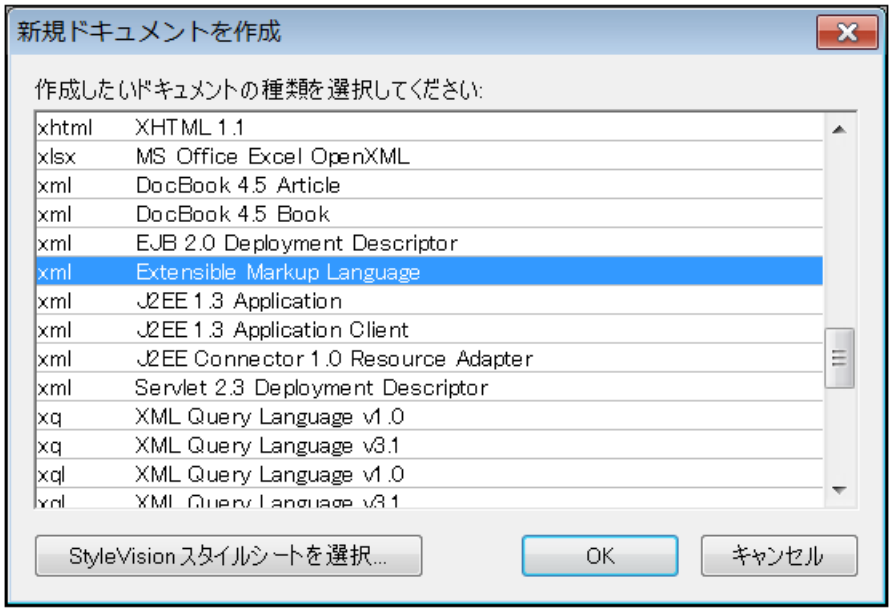
オプションダイアログの*ファイルの型*セクションにて `.xml` を選択し、デフォルトビューから目的の編集ビュー(テキストビューまたはグリッドビュー)を選択します。スキーマビューならびにWSDLビューはXMLスキーマとWSDLドキュメントでしか選択できないことと、ブラウザービューは表示のためのビューであり、編集ビューでは無いことご注意してください。

*ファイルの型*タブでは、選択されたファイルの種類に対してXMLSpyをデフォルトのエディターに設定することもできます。

StyleVision Power Stylesheet(SPS)ファイルが割り当てられている場合、Authenticビューでも XMLドキュメントを編集することができます。割り当てられたSPSファイルと一緒にXMLファイルが開かれた場合、Authenticビューで直接開くように設定することができます。オプションダイアログの*ビュー*セクションにある、*常にファイルをAuthenticビューで開く*にチェックを入れてください。このオプションにチェックが入っていない場合、*ファイルの型*タブ `.xml` ファイルで指定されたデフォルトビューにてファイルが開かれます。

スキーマの割り当て

新規XMLファイルを作成するには、「**ファイル | 新規作成**」メニューコマンドを選択してください。新規ドキュメントを作成ダイアログ(以下のスクリーンショット)が表示されます。



XMLドキュメントとして選択できるアイテムが複数あることにご注意してください。*Extensible Markup 言語* として記述されているオプションにより一般的なXMLドキュメントが作成されます。その他のオプションでは、例えばDocBook といったスキーマと、XMLドキュメントが既に関連付けられていたりします。関連付けされているこれらのオプションを選択すると、(i) 対応するスキーマが自動的に割り当てられ、(ii) 割り当てられたスキーマにより妥当なドキュメントのスケルトンがデフォルトでXMLドキュメントに作成されます。XMLドキュメントのスケルトンは独自に作成することができます。アプリケーションフォルダーの `Template` フォルダーに保存すれば、スケルトンドキュメントが新規ドキュメントを作成ダイアログにて使用できるようになります。

一般的なExtensive Markup 言語 をドキュメントの種類として選択した場合、ドキュメントにスキーマ(DTD またはXML スキーマ)を割り当てるかどうか質問されます。ここでは、スキーマを参照するか、スキーマを割り当てること無くXMLドキュメントの作成にとりかかることができます。

勿論、「DTD / スキーマ」メニューを使えば、ドキュメントの編集中いつでもスキーマを割り当てることができます。

自動検証

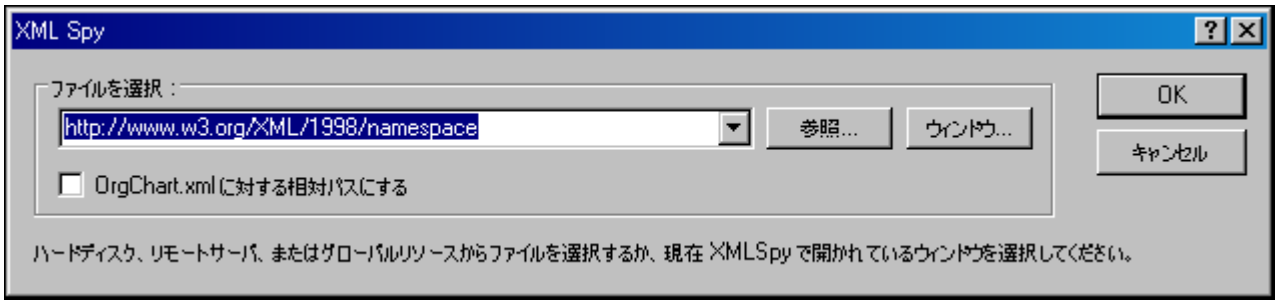
既存のXMLドキュメントにスキーマが割り当てられている場合、ドキュメントを開いたり保存する際に自動的に検証を行うことができます。オプションダイアログ(「ツール | オプション」)の*ファイル*セクションから設定を行うことができます。

*ファイル*タブ内の自動検証設定をファイルの型のセクションの設定と組み合わせることで、特定のファイルの型を自動的に検証することができます。これら2つのタブにある設定を行うことで、

5.2 スキーマの割り当てと検証

Altova ウェブサイト : XML バリデーター、XML 検証

XML ドキュメントの作成時[262]にスキーマ(DTD または XML スキーマ)の割り当てを行うことができます。スキーマの割り当てや変更は、「DTD / スキーマ」メニュー以下にある「DTD の割り当て」または「スキーマの割り当て」コマンドにより、いつでも行うことができます。



ダイアログ内の相対パスにチェックを入れることで、XML ドキュメントへ挿入されるスキーマファイルへのパスを XML ファイルから見た相対パスにすることができます。

スキーマのグローバルリソース

グローバルリソースとはファイルやフォルダーへのエイリアスです。ターゲットファイルまたはフォルダーは、(「ツール | アクティブな構成」により)グローバルリソースのアクティブ構成を変更することでグラフィカルに変更することができます。グローバルリソースを使えば割り当てられた複数のスキーマの切り替えを瞬時に行うことができ、試験を行うときなどに力を発揮します。グローバルリソースの使用方法については、Altova グローバルリソース[934]を参照ください。

XML スキーマと DTD

XML スキーマには無い DTD の便利な機能の1つにエンティティがあります。もし検証されたスキーマにてエンティティを使用したい場合は、DOCTYPE 宣言を XML ドキュメント内で行ない、その中でエンティティの宣言を行う必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrgChart [
  <!ENTITY name-int "value">
  <!ENTITY name-ext SYSTEM "extfile.xml">
]>
<OrgChart xmlns="http://www.xs.com/org"
          xsi:schemaLocation="http://www.xs.com/org OrgChart.xsd">
  ...
</OrgChart>
```

DTD 内部でエンティティの宣言を行った後に、それらを XML ドキュメント内で使用することができます。ドキュメントは整形式ならびに妥当なものとなります。しかし外部で解析されたエンティティは Authentic ビューでサポートされません。

スキーマの定義へ移動

XML ドキュメントが開かれていれば、「DTD/スキーマ」メニュー以下の DTD へ移動または スキーマへ移動コマンドをクリックすることで、ベースとなっている DTD または XML スキーマを直接開くことができます。更に、XML ドキュメント内のノードにカーソルを移動させ、「DTD/スキーマ | 定義へ移動」コマンドを使うことで、ノードのスキーマ定義に直接移動することができます。

検証と整形式のチェック

ドキュメントの検証や整形式のチェックを行うには、**検証 (F8)** や**整形式のチェック** (F7) コマンドを XML メニューまたはツールバーから使用します。エラーや警告がある場合は、メッセージウィンドウに表示されます。XML ドキュメントが不正の場合、スキーマ内にある情報をベースに、スマートフィックスによるエラーの修正[1227]を行うことができます。

RaptorXML Server[957] を使用して XML ドキュメントを検証することができます。

5.3 テキストビュー内の XML

XMLSpy は編集ビューセクション内のテキストビュー[130] 内で説明されているテキストビュー内の一般的に使用可能な編集機能に加え、特殊な XML テキスト編集機能を提供します。

- テキストのコメントインとアウト[266]
- XML 文字のエスケープとエスケープの解除[266]
- ファイルパスの挿入[267]
- XInclude を介したXML フラグメントの挿入[267]
- XPath とXpointer 式をクリップボードにコピーする[267]

テキストのコメントイン/アウト

XML コメントの開始とコメントの終了区切り文字 <!-- と--> をそれぞれ使用してXML ドキュメント内のテキストをコメントアウトすることができます。XMLSpy 内でこれらの区切り文字は **編集 | コメントイン/アウト** メニューコマンドを使用して簡単に挿入することができます。

テキストのブロックをコメントアウトするには、コメントアウトするテキストを選択し **コメントイン/アウト** を **編集** メニューまたは選択されたテキストを右クリックすることにより表示されるコンテキストメニューから選択します。コメントアウトされたテキストはグレーで表示されます (*以下のスクリーンショットを参照*)。

```
<Department>
    <Name>Administration</Name>
    <Person>
    <Person>
    <Person>
    <!--<Person>
        <First
        <Last></Last>
        <PhoneExt></PhoneExt>
        <EMail></EMail>
        <LeaveTotal></LeaveTotal>
        <LeaveUsed></LeaveUsed>
        <LeaveLeft></LeaveLeft>
    </Person>-->
</Department>
```

テキストのコメントされたブロックをアンコメントするには、コメント区切り文字を **除外** してコメントされたブロックを選択し、コマンド **コメントイン/アウト** を **編集** メニューまたは選択されたテキストを右クリックすると表示されるコンテキストメニューから選択します。コメント区切り文字はテキストがグレーアウトされなくなると削除されます。

空のラインに関するメモ

XML ドキュメント内では、ビューを変更またはドキュメントを保存する際に空のラインは破棄されます。空のラインを保持するには、これをコメントの区切り文字で囲みます。

XML 文字のエスケープとエスケープの解除

(*下にリストされる*)5個のXML 特殊文字 対応する(*下にリストされる*) エンティティレファレンスに従いテキストのブロックをハイライトし、コンテキストメニューコマンド **XML 文字をエスケープ** または **XML 文字をアンエスケープ**を選択してエスケープおよびアンエスケープすることができます。テキストのそのブロック内のXML 特殊文字は選択されたコマンドに従いエスケープまたはアンエスケープされます。

<	<
>	>
&	&

' '

" "

例:

<a> という文字列に対して「XML 文字のエスケープ」コマンドを適用すると <a> という文字列への変換が行われ、
<a> という文字列に対して「XML 文字のエスケープの解除」コマンドを適用することで、文字列が <a> に変換されます。

ファイル パスの挿入

編集 | ファイル パスを挿入[1154] コマンドにより、目的のファイルを参照し、そのファイル パスを XML ドキュメント内で選択されている場所に挿入することができます。このコマンドによりファイル パスを素早く正確に入力することができます。詳細については コマンドの記述[1154] を参照ください。

XInclude を介したXML フラグメントの挿入

編集 | XInclude を挿入[1155] により、XInclude により XML ドキュメント全体のコンテンツまたはフラグメントを編集中の XML ドキュメントに挿入することができます。このコマンドにより、XML ドキュメント全体、または XML ドキュメントのフラグメントを(XInclude メカニズムの XPointer 拡張を介して)素早く正確に入力することができます。詳細については、コマンドの記述を参照ください。詳細については コマンドの記述[1155] を参照ください。

XPath ならびに XPointer 表現をクリップボードにコピー

編集 | XPath をコピー[1154] および 編集 | XPointer をコピー[1154] を使うことで、選択されているノードの XPath と XPointer 表現をそれぞれクリップボードにコピーすることができます(ノードの識別は XML ドキュメント内の位置により行われます)。これにより目的とするノードの正確な XPath ならびに XPointer 表現を取得することができます。

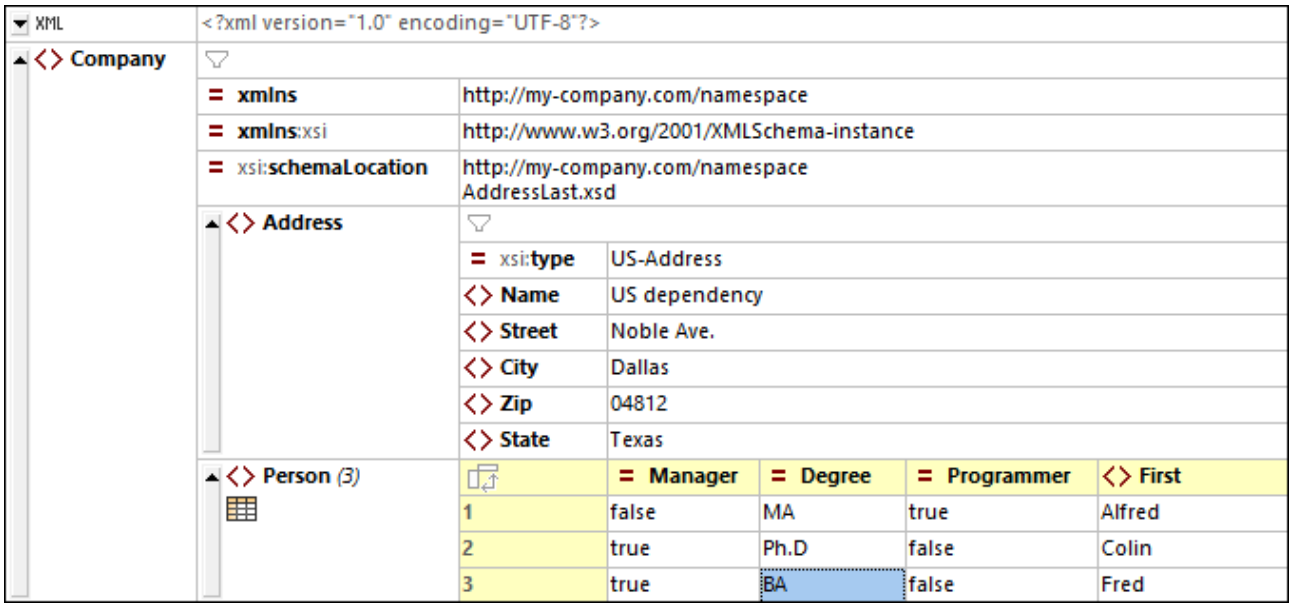
例えば、Offices というドキュメント要素以下にある3番目の Office 要素がテキストビューまたはグリッドビューにて選択されている場合を考えてみましょう。この場合 /Offices/Ofice[3] という XPath 条件式がコピーされます。コピーされた XPointer 式は、3番目の Office 要素の前に Office 要素が他の名前を持つ兄弟を持たない場合 element(/1/3) になります。

コピーされた条件式をは任意の場所に挿入することができます。例えば、XPath 条件式を XSLT スタイルシートへ挿入し、XPointer 条件式を xinclude 要素の href 属性内に挿入することができます。

コマンドのより詳細な情報については、ユーザーリファレンス内の対応するセクションを参照ください。

5.4 グリッドビュー内の XML

グリッドビュー[149] はXML ドキュメント の階層構造を展開または折りたたみ可能なネストされたコンテナーのセットを介して表示します。これはドキュメントの構造の明確なピクチャを提供します。グリッドビューでは、ドキュメント構造を簡単に変更でき、コンテンツを簡単に編集することができます。



上のスクリーンショット内で、ドキュメントがグリッドフォームで表示されていることに注意してください。ノードがコンテンツを含むことが可能な場合、ノードは(*テキスト* ノードの場合と同様)直接コンテンツを含む、または(*要素* ノードの場合と同様)以下の2つのフィールドに分割されます: ノード名とノードコンテンツ。ノード名は太字で表示され、ノードコンテンツは通常の書体で表示されます。

グリッドビューツールバー

アプリケーションウィンドウの上のツールバーエリアにあるグリッドビューツールバーを使用してグリッドビュー内の編集に関連するコマンドにアクセスすることができます(*下のスクリーンショット*)。名前とショートカットを確認するにはツールバーアイコンにマウスをポイントしてください。

メモ これらのコマンドはXML メニュー[1198] 内で使用可能です。

グリッドビュー機能の概要:

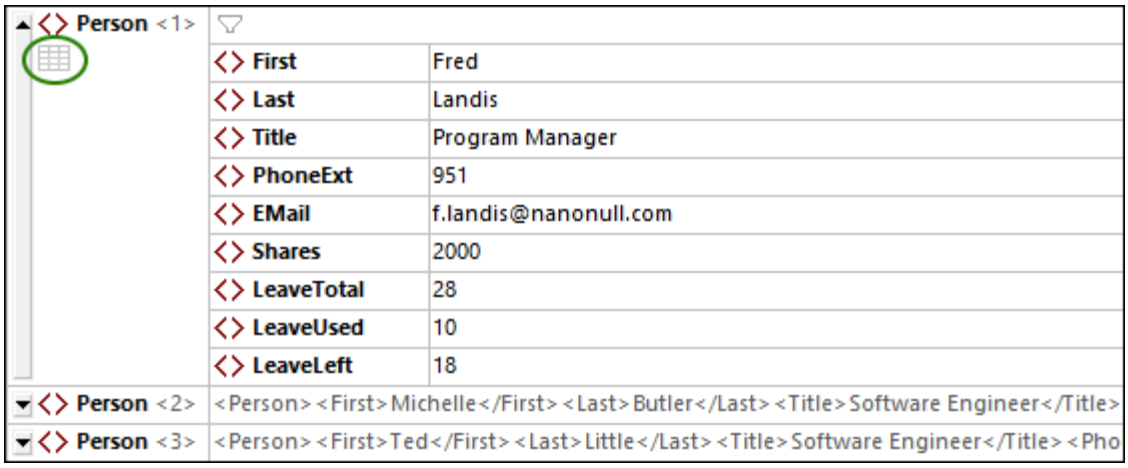
グリッドビュー機能の概要に関しては以下を参照してください:

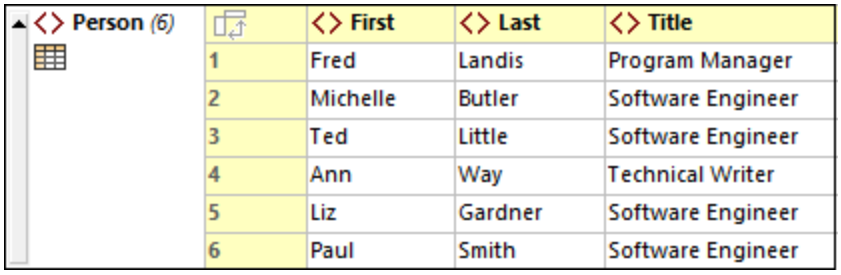
- グリッドビューの表示機能を表示するドキュメントの表示[270]。
- ドキュメントの構造がグリッドビューでどのように表示されるかを記述し、ドキュメント構造を変更するためにグリッドビュー機能を使用する方法を説明するドキュメントの構造[273]。
- グリッドビュー内でコンテンツの編集と検証を表示するドキュメントコンテンツ[275]
- グリッドビューに適用する設定をリストし説明するグリッドビュー設定[296]。

テーブルとして表示する

繰り返し要素は*標準のグリッドビュー*でドキュメント順に垂直下法で順番に表示されています(*左下のスクリーンショット参照*)。しかしながら、繰り返し要素を *テーブルの行* として表示すると追加編集機能が与えられます。下の*スクリーンショット*では `Person` 要素は繰り返し要素です。左の*スクリーンショット*は標準の*グリッドビュー*を表示しています。最初の `Person` 要素が展開されて表示され、続く*インスタンス*は折りたたまれて表示されます。右の*スクリーンショット*は繰り返し`Person` 要素をテーブルの行として表示しています。

テーブルビューに切り替えるには(左下の*スクリーンショット*内で緑色の丸で囲まれている) **テーブルモード** *アイコン をクリックします*。テーブルディスプレイに切り替えると、アイコンは色で表示されます *(右下のスクリーンショット)*。





テーブルビューはその行と列全体がテーブル内の他の列と行に相対的に操作可能になるユニークな編集の利点を提供します。これにより列の値上のテーブル行の並べ替えなどの操作が可能になります。例えば、右上の*スクリーンショット*では、6個の `Person` 要素が[グリッドビューツールバー](#)[268]内の単一のコマンドにより `Last` 子要素をベースに並べ替えられることができます。これはXML ノードセットを並べ替えるための通常の方法であるXSLT 変換を実行するよりもシンプルです。

詳細に関しては、[テーブル表示](#)[281]のトピックを参照してください。

フィルターとフォーミュラ

グリッドビューは2つのユニークで役に立つ機能を提供します。

- グリッドビューで表示されているデータのフィルターを可能にする[フィルター](#)[285]。ドキュメント内で1つまたは複数の要素にフィルターを設定することができます。
- [フォーミュラ](#)[286] により結果を計算、またはドキュメント内に保管可能なノードセットを生成することができます。

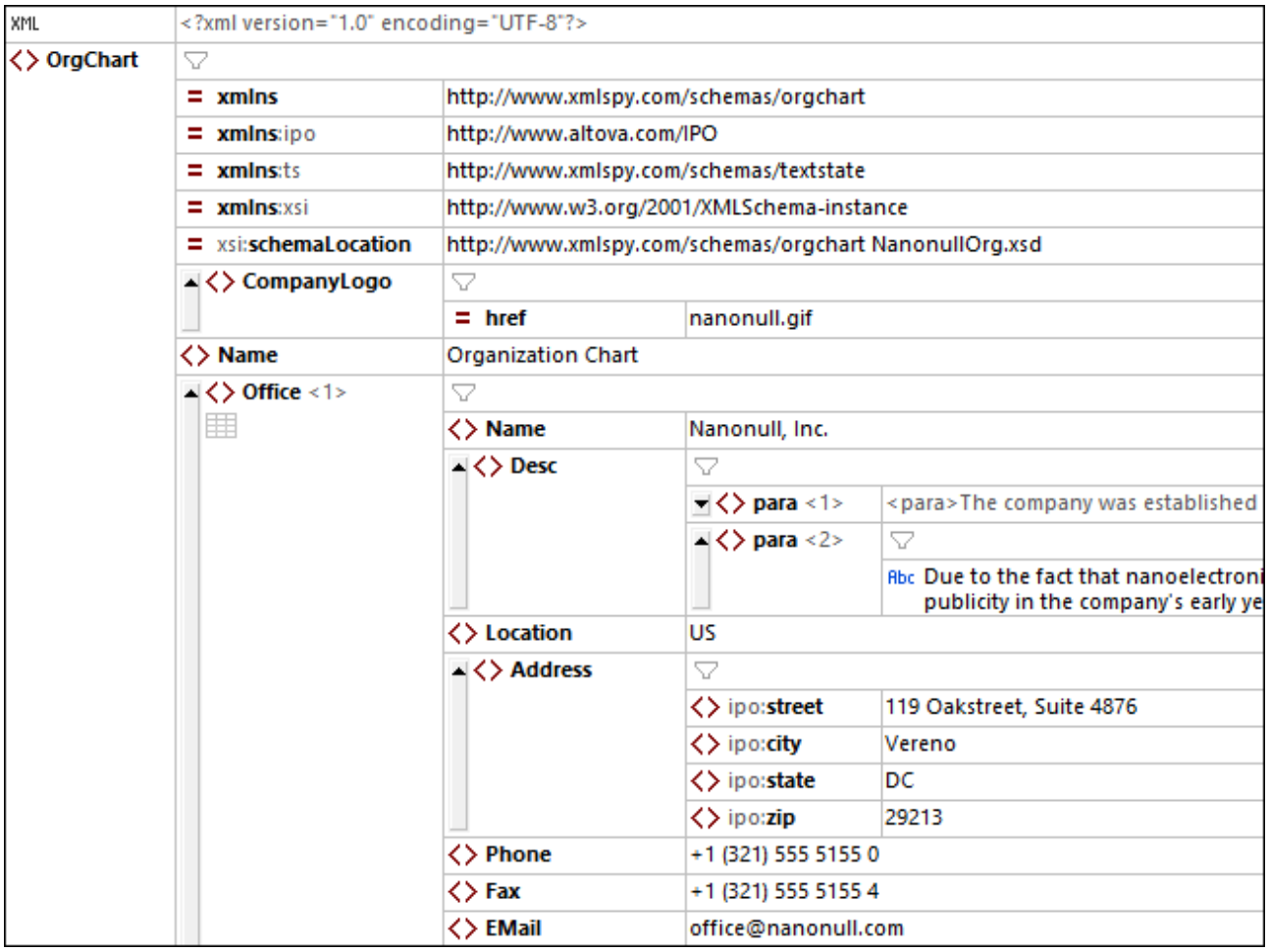
詳細に関しては、関連するトピックを参照してください。

グリッドビュー内でのイメージとチャート

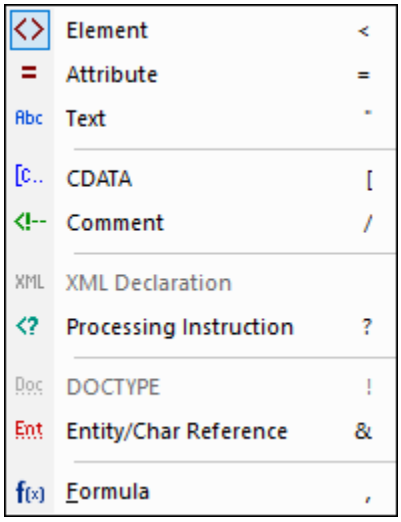
イメージとチャート はグリッドビュー内でグラフィックとして表示することができます。この方法についての詳細はグリッドビュー内のイメージの表示 289 とグリッドビュー内のチャート 290 を参照してください。

5.4.1 ドキュメントの表示

グリッドビュー内ではXMLドキュメントは階層的に表示されます（*下のスクリーンショットを参照*）。



各グリッドラインは要素、属性、コメントまたはテキストなどの1つの（ノードとして既知の）XML構造アイテムを表示します。XMLドキュメント内で使用可能な異なるノード型はアイコンと共に下のスクリーンショットで表示されています。グリッドビューでは、ノードの型はグリッドセル内の左側にあるアイコンで示されています。型 *要素*、*属性* と *処理命令* のノードはキーと値を持ちますが、*テキスト*、*CDATA* および *コメント* は値のみを持っています。例えば、要素ノードは名前（自身のキー）とコンテンツ（自身の値），を持ちますが、テキストノードは値のみを持ちます。



グリッドビュー表示の機能

以下の機能に注意してください:

- Ctrl + **マウスのホイール** または Ctrl+[プラス] と Ctrl+[マイナス] を使用してグリッドをズームすることができます。
- 同じ名前を持つ兄弟要素には 1 から始まる番号が与えられます。上のスクリーンショットでは、例えば、1 の番号を持つ `Office` 要素が `Office` 要素のシーケンスの最初になります。同じ名前の兄弟要素を持たないためスクリーンショットの他の要素は番号を付けられていません。
- ノードシンボルの左側の矢印アイコンを押してノードを展開または折りたたむことができます *(上のスクリーンショットを参照)*。ノードが折りたたまれると、コンテンツは単一のライン内のテキストで表示されます。
- 複数のコンポーネントを同じレベルで選択すると、これらを **Shift** を押し、選択されたコンポーネントの矢印をクリックして展開/折りたたむことができます。
- ワードラップが[グリッドビュールールバー](#)[268]を介してオンに切り替えられている場合、セルの幅より長いテキストを含むすべてのセルはラップされます。ワードラップを[グリッドビューツールバー内の](#)[268]アイコンをクリックしてオフに切り替えます。
- [フォーミュラ](#)[617] はXMLSpy に固有です。

ツールバー/セル内のコマンドボタン

XML グリッドビュー内では、グリッドセル内または[グリッドビューツールバー](#)[268] 内のボタンは現在の選択範囲 (セルまたは列) のコンテキストで使用可能な機能を提供します。これらのコマンドの使用は一般的に使用される操作を更に素早く簡単に実行する方法です。使用可能なコマンドは下のテーブルにリストされています。表示に関連したコマンドは青い背景と共に表示されています。ドキュメント (構造とコンテンツ) の編集のためのコマンドはベージュの背景で表示されています。特定のセルのコンテキスト内の追加コマンドのためにセルの[コンテキストメニュー](#)[625]を使用してください。メモ セルボタンまたはツールバーアイコンをポイントするとコマンドの名前がツールヒントとして表示されます。

	セルコンテンツに従いグリッド列の幅を最適化
/	色で表示されている場合 [テーブルの表示](#)[613] がオンに切り替えられています。それ以外の場合はオフです。表示するためにクリックします。
/	テーブルの左側のセル内のコマンドをトグルします。行を列に、または列を行に切り替えます。
/	色で表示されている場合、テーブル、オブジェクト、または配列のための[フィルター](#)[617] が有効化されています。それ以外の場合フィルターは無効化されています。無効化/有効化するためにクリックします。式を編集するには、ダブルクリックします。

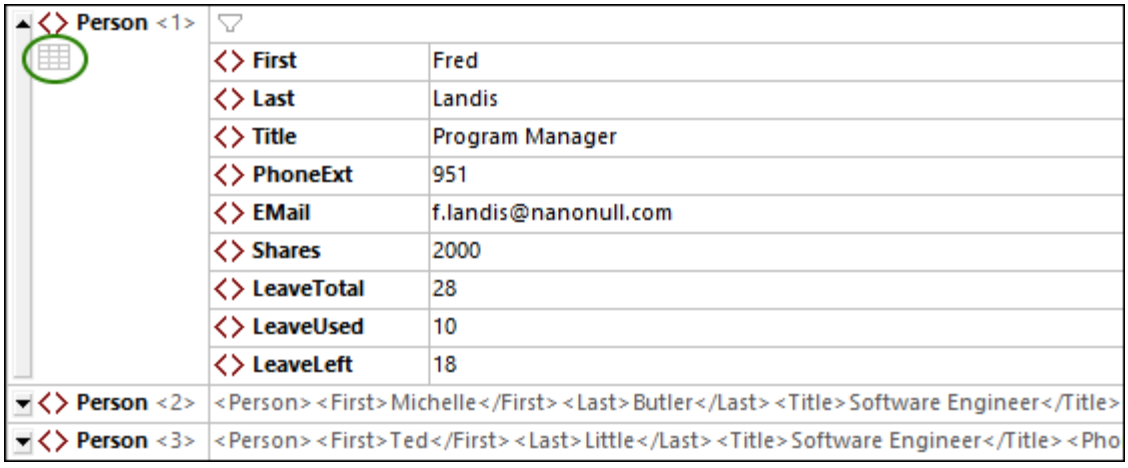
	セルをワードラップするためのトグルコマンド。選択されると、ワードラップがオンになります。デフォルトでは、アイテムのコンテンツのみがラップされます。アイテムの名前を追加でラップする場合、このオプションをアイコンのドロップダウンリストから選択します。
100%	グリッドビューのズームレベル
	テーブルビュー内で列ヘッダーが選択されると有効化されます。ボタンはテーブルの行を列のコンテンツの降順または昇順の順序にそれぞれ並べ替えます。

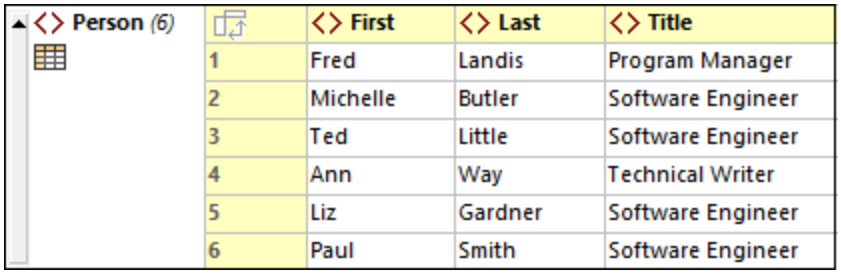
メモ 他のツールバーコマンドの詳細に関しては、セクション[グリッドビュー内のコンテキストメニュー](293)を参照してください。

テーブルとして表示する

繰り返し要素は*標準*のグリッドビューでドキュメント順に垂直下法で順番に表示されています(*左下のスクリーンショット参照*)。しかしながら、繰り返し要素を *テーブルの行* として表示すると追加編集機能が与えられます。下の*スクリーンショット*では`Person` 要素は繰り返し要素です。左の*スクリーンショット*は標準のグリッドビューを表示しています。最初の`Person` 要素が展開されて表示され、続く*インスタンス*は折りたたまれて表示されます。右の*スクリーンショット*は繰り返し`Person` 要素をテーブルの行として表示しています。

テーブルビューに切り替えるには(左下の*スクリーンショット*内で緑色の丸で囲まれている) **テーブルモード** *アイコン をクリックします*。テーブルディスプレイに切り替えると、アイコンは色で表示されます *(右下のスクリーンショット)*。





テーブルビューはその行と列全体がテーブル内の他の列と行に相対的に操作可能になるユニークな編集の利点を提供します。これにより列の値上のテーブル行の並べ替えなどの操作が可能になります。例えば、右上の*スクリーンショット*では、6個の`Person` 要素が[グリッドビューツールバー](268)内の単一のコマンドにより`Last` 子要素をベースに並べ替えられることができます。これはXML ノードセットを並べ替えるための通常の方法であるXSLT 変換を実行するよりも*シンプル*です。

詳細に関しては、[テーブル表示](281)のトピックを参照してください。

5.4.2 ドキュメント構造

グリッドビューではXMLドキュメント構造を視覚的に編集することができます。例えば、ドキュメント構造の個別のセグメントを折りたたむまたは展開する、ノードを挿入、追加および削除する、異なる個所にノードをドラッグアンドドロップする、および1つの型を他の型に変換することができます。

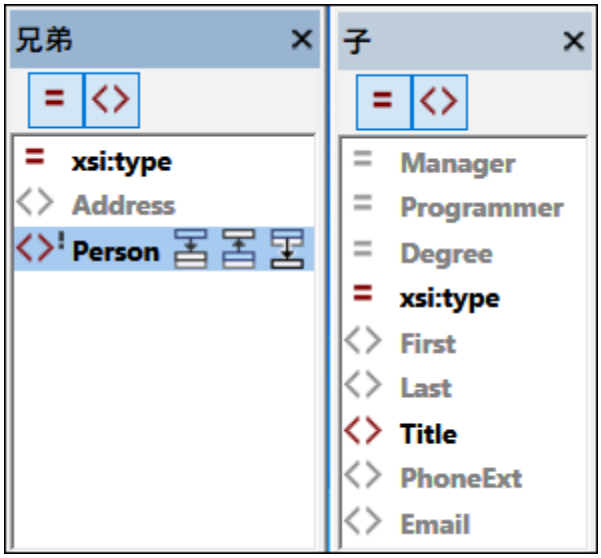
新規のノードの追加

ドキュメントに新規のノードを追加する方法が2通りあります:

- 兄弟と子入力ヘルパーは選択された箇所に特定のノードを入力します。これが作動するにはXMLドキュメントがスキーマ(DTDまたはXML スキーマ)に関連付けられている必要があります。例えば、メインウィンドウ内でノードを選択する場合、(スキーマに従い) その箇所で許可されている兄弟と子がそれぞれ兄弟と子入力ヘルパーに表示されます。入力ヘルパーを使用して追加するノードを選択することができます。
- 兄弟、子、または親として定義されていない新規のノードを追加することができます。グリッドビューセルのコンテキストメニュー内でこれを行うコマンドを使用することができます。また XML メニュー[1198] とグリッドビューツールバー[268] でも使用することができます。

兄弟と子入力ヘルパー

XMLドキュメントがスキーマ(DTD またはXML スキーマ)に関連づけられている場合、使用可能な選択されたノードの兄弟と子が兄弟と子入力ヘルパーに表示されます。これらの入力ヘルパー内で要素と属性のトグルの可視性を対応するツールバーアイコンをクリックすることによりオンとオフに切り替えることができます。必須のノードは感嘆符と共に表示されています。灰色表示されているノードは追加することができません。追加できない理由は以下の理由の一つです: (i) ノードが既に追加されており、下のスクリーンショット内の`Address` 兄弟の場合で示されている通りスキーマにより更なるノードが許可されている場合。(ii)下のスクリーンショット内の`First` 子が追加された後に追加可能になる`Last` 子の場合で示されている通り灰色表示されているノードの前に他のノードが追加される必要がある場合。



ノードを以下のように追加します:

- *兄弟:* 入力ヘルパー内で、兄弟として追加するノードを選択します。グリッド内で選択されているノードの後または前に兄弟を追加するか、または選択されたノードの兄弟の最後として付加するかにより適切なアイコンします *(上の左側のスクリーンショットを参照)*。
- *子:* 子として追加するノードをダブルクリックします。

新規のノードの追加

ドキュメント（すなわちグリッドセル）内のノードが選択されるとからのノードを兄弟、子、親として追加することができます。これらの操作のためのコマンドは下にリストされており、以下で使用することができます: (I) セルのコンテキストメニュー(ii) XML メニュー[1198]。および(iii) グリッドビューツールバー[268]。

コマンド	ショートカット
後に(兄弟) を挿入	Ctr+Enter
前に(兄弟) を挿入	Ctr+Shift+Enter
付加(兄弟)	Ctr+Alt+A
子の追加	Ctr+Alt+Enter
要素内のラップ	Ctr+Alt+W

以下の点に注意してください:

- 新規のノードはデフォルトで空の要素として作成されます。ノード型を後に変更[275] することができます。
- **要素内のラップ** コマンドは現在のノードの周りに要素ノードを作成します。この要素は現在のノードの新規の親となります。

標準の Windows メカニズムを使用して構造を変更します。

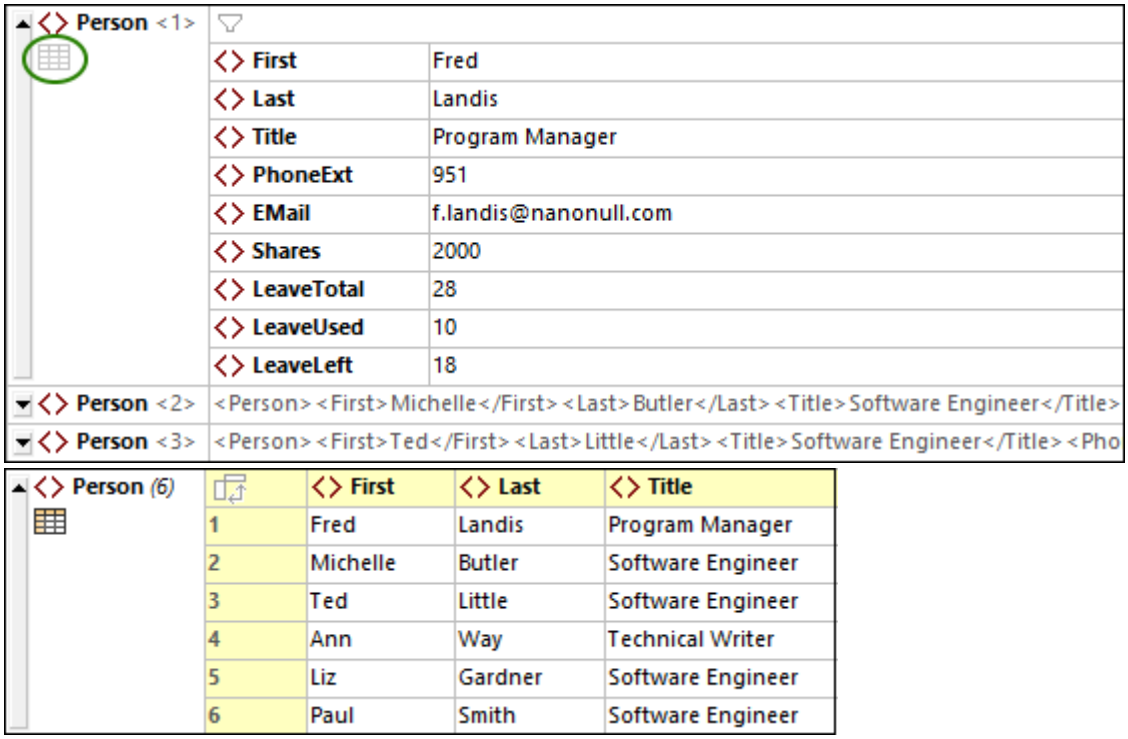
グリッドビューでは Windows メカニズムを使用してドキュメント構造を変更することもできます:

- *削除:* コンポーネントを選択して **Delete** キーを使用して削除します。
- *移動:* コンポーネントを選択してドラッグアンドドロップを使用して新規のロケーションにドロップします。

テーブルとして表示する

繰り返し要素は*標準のグリッドビュー*でドキュメント順に垂直下法で順番に表示されています(*左下のスクリーンショット参照*)。しかしながら、繰り返し要素を *テーブルの行* として表示すると追加編集機能が与えられます。下の*スクリーンショット*では `Person` 要素は繰り返し要素です。左の*スクリーンショット*は標準のグリッドビューを表示しています。最初の `Person` 要素が展開されて表示され、続く*インスタンス*は折りたたまれて表示されます。右の*スクリーンショット*は繰り返し `Person` 要素をテーブルの行として表示しています。

テーブルビューに切り替えるには(左下のスクリーンショット内で緑色の丸で囲まれている) **テーブルモード** *アイコンをクリックします*。テーブルディスプレイに切り替えると、アイコンは色で表示されます *(右下のスクリーンショット)*。



Person <1>	
First	Fred
Last	Landis
Title	Program Manager
PhoneExt	951
EMail	f.landis@nanonull.com
Shares	2000
LeaveTotal	28
LeaveUsed	10
LeaveLeft	18
Person <2>	<Person><First>Michelle</First><Last>Butler</Last><Title>Software Engineer</Title>
Person <3>	<Person><First>Ted</First><Last>Little</Last><Title>Software Engineer</Title><Pho

Person (6)	First	Last	Title
1	Fred	Landis	Program Manager
2	Michelle	Butler	Software Engineer
3	Ted	Little	Software Engineer
4	Ann	Way	Technical Writer
5	Liz	Gardner	Software Engineer
6	Paul	Smith	Software Engineer

テーブルビューはその行と列全体がテーブル内の他の列と行に相対的に操作可能になるユニークな編集の利点を提供します。これにより列の値上のテーブル行の並べ替えなどの操作が可能になります。例えば、右上のスクリーンショットでは、6個の `Person` 要素がグリッドビューツールバー[268]内の単一のコマンドにより `Last` 子要素をベースに並べ替えられることができます。これはXML ノードセットを並べ替えるための通常の方法であるXSLT 変換を実行するよりもシンプルです。

詳細に関しては、テーブル表示[281]のトピックを参照してください。

5.4.3 ドキュメントコンテキスト

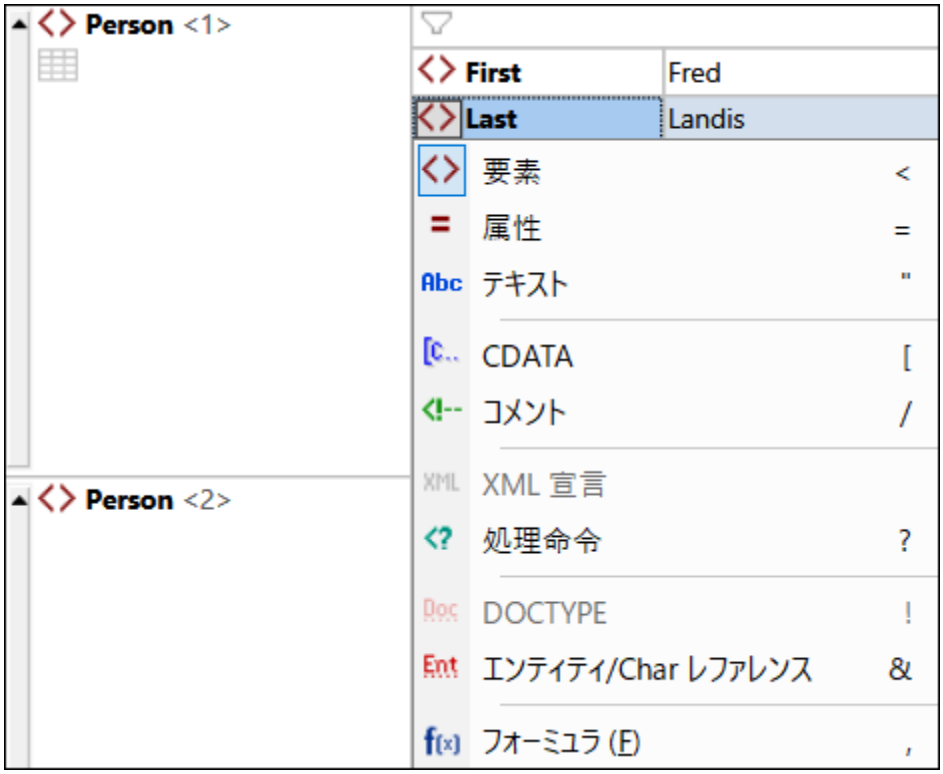
グリッドビュー内のXML コンテンツの編集はとても理解しやすいです: コンテンツフィールドの内部をダブルクリックし、必要に応じてコンテンツを編集します。ノードの型アイコンをクリックし表示されるメニューから他の型を選択することによりノード型は素早く変更されます。エンティティ入力ヘルパーを介してエンティティを挿入することができます。グリッドビューは更に検証と検索と置き換え機能もオファーしています。

ノードの型の選択と変換

グリッドビュー内のセルはXML ドキュメントのノードを含んでいます。よりよく理解するために型は以下のように分類されています:

- *名前-値型:* 要素、属性、処理命令 (PI)。この型のノードは名前と値を有しています。
- *値型:* テキスト、CDATA、コメント、エンティティ/文字レファレンス。この型のノードは値のみを有します。
- *定義型:* XML 宣言、DOCTYPE (内部または外部 DTD)。これらのノードはXML ドキュメントのプロパティを定義します。
- *XMLSpy に固有の特別なコンテナー型:* フォーミュラ。

値の前のシンボルにより示される値の型 *(下のスクリーンショットを参照)*。型を変更するには、シンボルをクリックして、表示されるメニューから型を選択します。または、セルを右クリックして、表示されるコンテキストメニューの 型 サブメニューから型を選択します。型のシンボルとショートカットは下のスクリーンショットに表示されています:



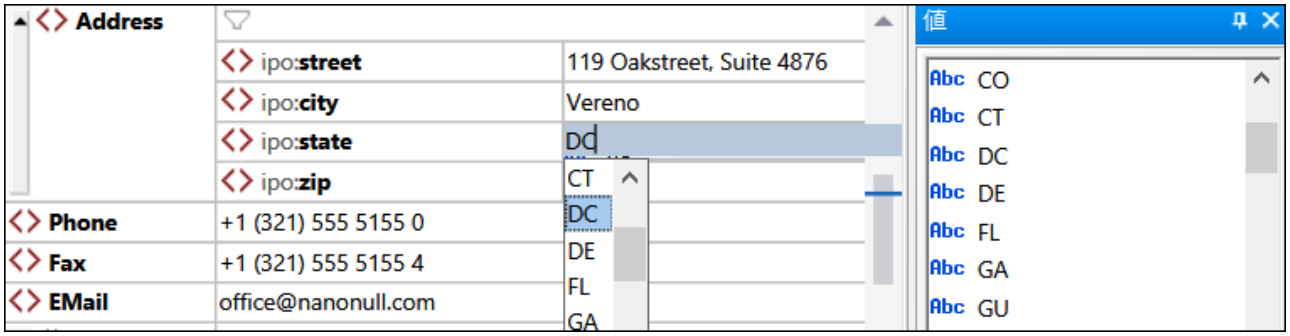
以下の点に注意してください：

- 型 *要素*、*属性* および *PI* は名前と値を持ちますが型 *テキスト*、*CDATA* および *コメント* は値だけを持ちます。例えば、要素ノードは名前とコンテンツを持ちますが、テキストノードは値のみを持ちます。
- 型の変換は元のキーと値を保持しようと試みます。例えば、要素を属性に変換すると属性は (i) 要素と同じ属性 (ii) 要素のコンテンツと可能な限り同じの値を持ちます。
- 要素の子ノードのシーケンス内では属性は常に最初にリストされます。この結果、型変換はノードの並べ替えを引き起こします。
- (テキストと太字および斜体要素を含む) パラグラフ要素など要素の子が混在する文字データ) 複合型コンテンツの場合 連続したテキストノードは許可されておらず自動的に連結される可能性があります。
- 複数のノードが単一型に変換されるために選択される場合、この変更が先に行われるか否か、または、警告を与えるかを決定するオプションを設定することができます。

自動補完

編集される XML ドキュメントがスキーマに関連している場合自動補完が有効化されます。

自動補完はカーソルの位置で入力オプションを提供します。これは以下を使用して行うことができます (*下のスクリーンショット参照*)。これらのオプションはスキーマ内の定義をベースにしており (i) メインウィンドウ内のポップアップ (ii) 入力ヘルパーを介して提供されます。ポップアップと入力ヘルパーはそれぞれカーソルのロケーションで有効なエントリのリストを表示します。ポップアップリスト内のエントリを移動するには矢印キーを使用してください。エントリをポップアップウィンドウから選択、または入力ヘルパー内のエントリをダブルクリックして挿入します。



ツールバー/セル内のコマンドボタン

XML グリッドビュー内では、グリッドセル内または[グリッドビューツールバー](#)[268] 内のボタンは現在の選択範囲（セルまたは列）のコンテキストで使用可能な機能を提供します。これらのコマンドの使用は一般的に使用される操作を更に素早く簡単に実行する方法です。使用可能なコマンドは下のテーブルにリストされています。表示に関連したコマンドは青い背景と共に表示されています。ドキュメント（構造とコンテンツ）の編集のためのコマンドはベージュの背景で表示されています。特定のセルのコンテキスト内の追加コマンドのためにセルの[コンテキストメニュー](#)[625]を使用してください。メモ セルボタンまたはツールバーアイコンをポイントするとコマンドの名前がツールヒントとして表示されます。

	セルコンテンツに従いグリッド列の幅を最適化
/	色で表示されている場合 [テーブルの表示](#)[613] がオンに切り替えられています。それ以外の場合はオフです。表示するためにクリックします。
/	テーブルの左側のセル内のコマンドをトグルします。行を列に、または列を行に切り替えます。
/	色で表示されている場合、テーブル、オブジェクト、または配列のための[フィルター](#)[617] が有効化されています。それ以外の場合フィルターは無効化されています。無効化/有効化するためにクリックします。式を編集するには、ダブルクリックします。
WRAP	セルをワードラップするためのトグルコマンド。選択されると、ワードラップがオンになります。デフォルトでは、アイテムのコンテンツのみがラップされます。アイテムの名前を追加でラップする場合、このオプションをアイコンのドロップダウンリストから選択します。
100%	グリッドビューのズームレベル
	テーブルビュー内で列ヘッダーが選択されると有効化されます。ボタンはテーブルの行を列のコンテンツの降順または昇順の順序にそれぞれ並べ替えます。

メモ 他のツールバーコマンドの詳細に関しては、セクション[グリッドビュー内のコンテキストメニュー](#)[293]を参照してください。

変更中に検証

編集中に検証 モードはデフォルトでオフに切り替えられています。オンに切り替えられていると、グリッドビュー内でのドキュメントの編集中に整形式チェックと検証チェックが実行されます。（整形式のチェックに加え）検証を実行するにはDTD またはXML スキーマがドキュメントに割り当てられている必要があります。エラーはエラーの存在するテキストを赤色で表示し、エラーの箇所を赤い感嘆符で表示することで表示されています。エラーのためにスマートフィックスを使用することができる場合、電球アイコンがエラーを生成するライン内に表示されます。アイコンにマウスをポイントすると、使用することのできるスマートフィックスのリストのポップアップが表示されます。適用するフィックスを選択します。詳細に関しては[ドキュメントの検証](#)[302] を参照してください。

編集中に検証 モードは(i) [「XML | 編集中に検証」](#)[1233] メニューコマンド (ii) **編集中に検証** ツールバーボタン または(iii) [オプションダイアログの検証設定](#)[1455] の *編集時* オプションによりオンとオフに切り替えることができます。

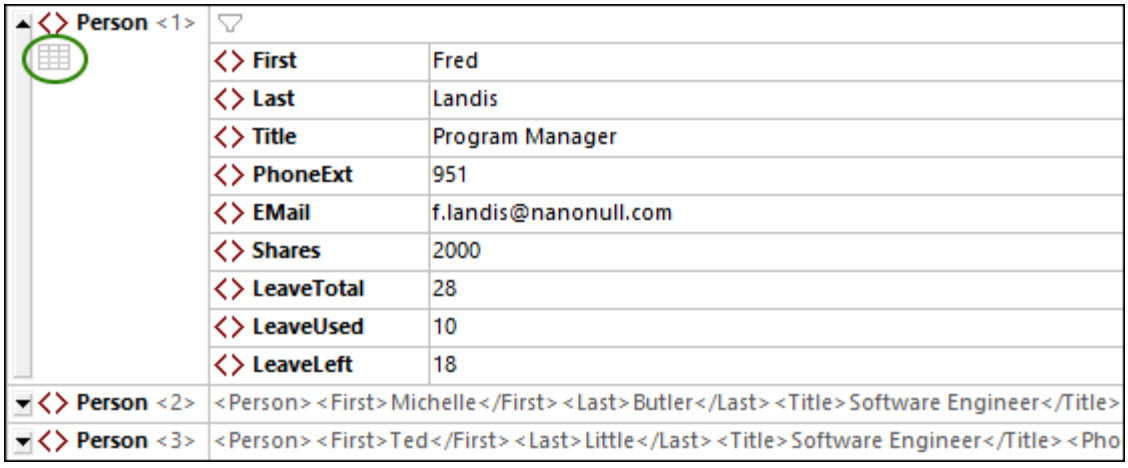
検索と置換

グリッドビュー内でテキストを検索し置き換えるために **編集** メニューの **検索** (Ctrl+F) と **置換** (Ctrl+H) コマンドを使用することができます。結果はオレンジ色でハイライトされ、含まれているセルもオレンジ色でハイライトされています。

テーブルとして表示する

繰り返し要素は*標準のグリッドビュー*でドキュメント順に垂直下法で順番に表示されています(*左下のスクリーンショット参照*)。しかしながら、繰り返し要素を *テーブルの行* として表示すると追加編集機能が与えられます。下の*スクリーンショット*では `Person` 要素は繰り返し要素です。左の*スクリーンショット*は標準のグリッドビューを表示しています。最初の `Person` 要素が展開されて表示され、続く*インスタンス*は折りたたまれて表示されます。右の*スクリーンショット*は繰り返し `Person` 要素をテーブルの行として表示しています。

テーブルビューに切り替えるには(左下の*スクリーンショット*内で緑色の丸で囲まれている) **テーブルモード** *アイコンをクリックします*。テーブルディスプレイに切り替えると、アイコンは色で表示されます *(右下のスクリーンショット)*。



テーブルビューはその行と列全体がテーブル内の他の列と行に相対的に操作可能になるユニークな編集の利点を提供します。これにより列の値上のテーブル行の並べ替えなどの操作が可能になります。例えば、右上の*スクリーンショット*では、6個の `Person` 要素が[グリッドビューツールバー](#)[268]内の単一のコマンドにより `Last` 子要素をベースに並べ替えられることができます。これはXML ノードセットを並べ替えるための通常の方法であるXSLT 変換を実行するよりもシンプルです。

詳細に関しては、[テーブル表示](#)[281]のトピックを参照してください。

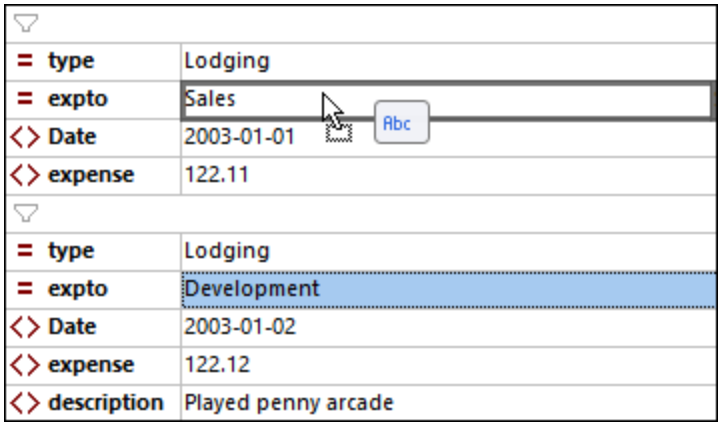
5.4.4 グリッドビュー内でのドラッグとドロップ

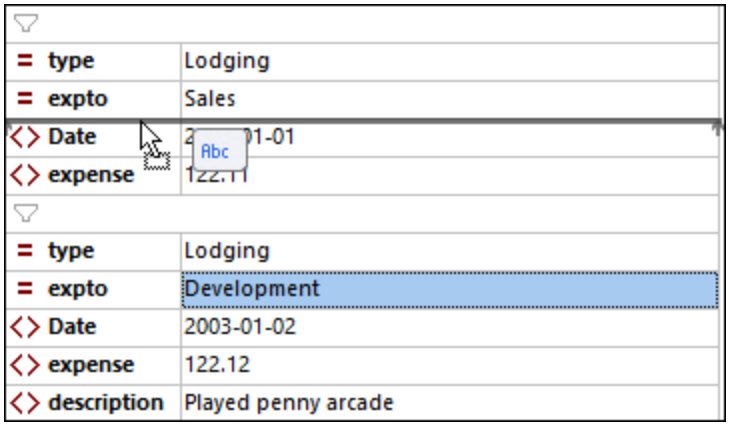
グリッドビューはJSON ドキュメントフラグメントをグリッドビューにXMLSpy 内、外部アプリケーション、またはW eb サイトで開かれているXML ドキュメントからドラッグすることのできる役に立つドラッグオーバーレイを提供します。

ターゲットノードにカーソルをポイントすると、ドラッグされている（アイテムの型と数量）だけでなくドロップされた際にどのようにアイテムが作成されるかについての情報を提供するドラッグオーバーレイが表示されます。例えば、下のスクリーンショットでは`Development` がドラッグ（移動）されています。アイテムが文字列であることを`Abc` ポップアップにより示されています。

- （左側のスクリーンショットで示されている通り）値フィールドにポイントすると、フィールド全体がマークされ、マークされたフィールドの値がドラッグされたフィールドの値と置き換えられます。
- （右側のスクリーンショット内の`Date` などの）ノード名にポイントすると、値文字列をドロップするとツリー内にテキストノードを作成することを示す挿入ラインが表示されます。

メモ　　**Ctrl+Z** またはメニューコマンド **Edit | Undo**を使用して移動を元に戻すことができます。



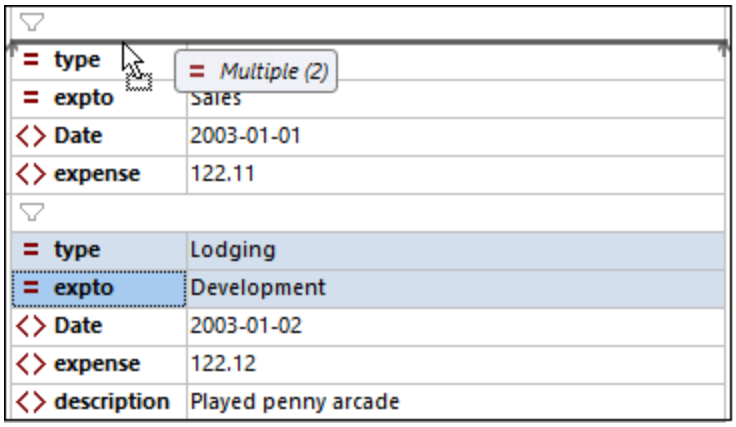


ドラッグオーバーレイ内に含まれている情報

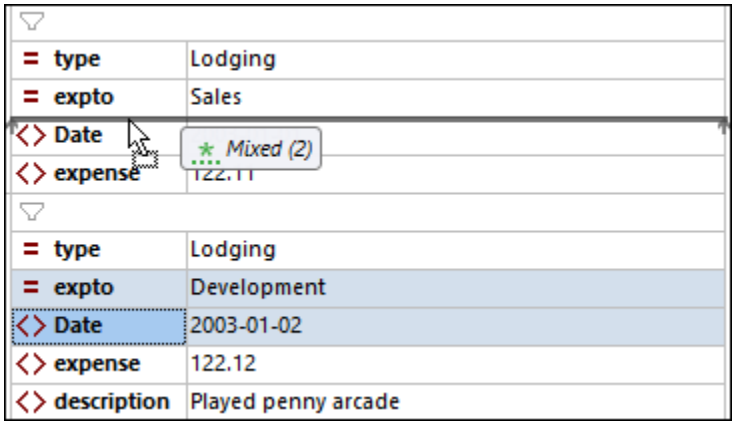
以下の種類のドラッグオーバーレイ情報が表示されます。

通常のグリッドビュー

- 値フィールドにドロップされた値はターゲットフィールド内の値を上書きするためにソース値を移動します（*左上のスクリーンショット参照*）。
- ツリーにノードとしてドロップされる値はテキストノードを作成します（*右上のスクリーンショット参照*）。
- 単純型の複数のノードのオーバーレイはドロップされたノード型とこれらのノードの数値を表示します。（値ではなく）ノードを選択するために、ノードの名前をクリックする必要があります。下のスクリーンショットでは、ドラッグオーバーレイ内の情報は2個の属性がドロップされることを示しています。

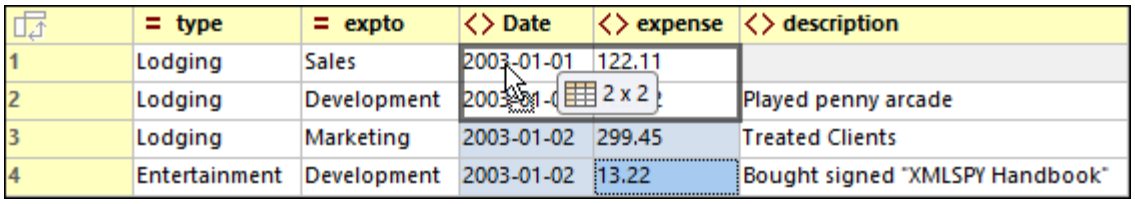


- 異なる型の複数のノードのためのオーバーレイはテキスト *混合* とクリップボード上のノードの数量を表示しています（*下のスクリーンショット参照*）。（値ではなく）ノードを選択するために、ノードの名前をクリックする必要があります。

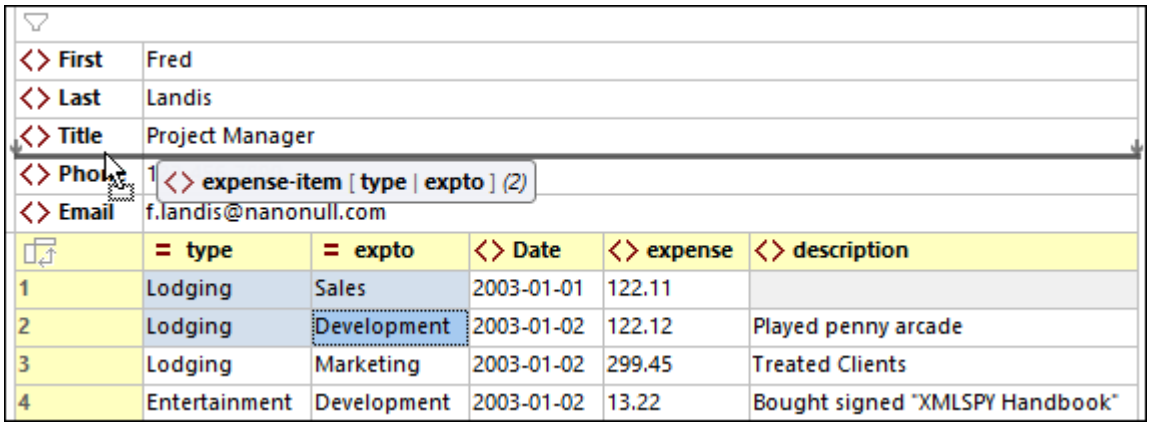


テーブルの表示

- 値がドラッグされると、選択されたセルの数値は `列 x 行` により示されています（*下のスクリーンショット参照*）。ドラッグされたセルの間トリックは対応するマトリックスを置き換えます。ターゲットセル（マトリックスをドロップするセル）はマトリックスの左のセルの値を受け取ります。他のドラッグされたセルがターゲットセルから右方向および下方向にフィルします。例えば、下のスクリーンショットでは青色のセルがドラッグされ最初の `Date` セルにドラッグされています。これによりマークされたセルが置き換えられます。ドラッグされたセルのマトリックスがテーブルの枠を超えると、列および/または行の適切な数量がすべてのドラッグされたセルに対応するように追加されます。この場合、拡張される枠は点線で表示されます。



- ターゲットがセルではなくノードツリーの場合、追加されるノードは(i)（角かっこに含まれる）追加されるノード、と(ii) 新規の要素のために追加される行の数量 と共に表示されます（*下のスクリーンショットを参照*）。

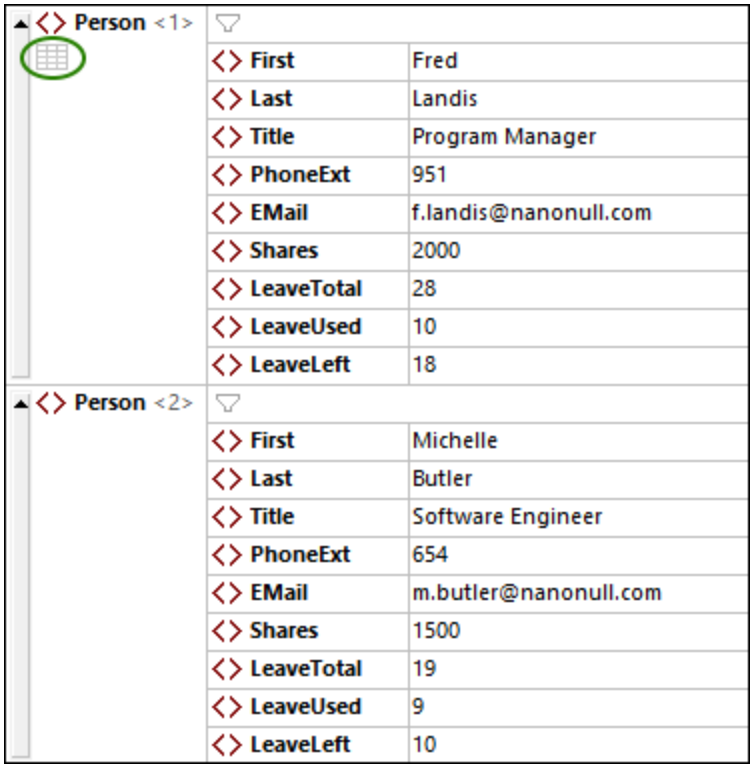


- （ヘッダーをクリックして）選択し、隣接する列ヘッダーにドロップして列を移動することができます。挿入ラインは移動された列が配置されるターゲット列のサイドを示しています。

5.4.5 テーブルの表示

テーブルの表示について

標準のグリッドビューでは、要素が繰り返されると、各要素インスタンスが表示されます。下のスクリーンショットでは `Person` 要素は繰り返し要素です。



上に表示される Person 要素などの繰り返し要素をテーブルとして表示することができます（*下のスクリーンショット*）。テーブル表示では、繰り返し要素の子ノードはテーブルの列を形成し、繰り返し要素のインスタンスはテーブルの行を形成します。テーブルビューに切り替えるには繰り返し要素の最初のインスタンス上に表示されている（*上のスクリーンショット内で緑色で囲まれている*）**テーブルモード** アイコンをクリックします。テーブルディスプレイに切り替えると**テーブルモード** アイコンは色で表示されます（*右のスクリーンショット*）。

Person (6)		First	Last	Title
	1	Fred	Landis	Program Manager
	2	Michelle	Butler	Software Engineer
	3	Ted	Little	Software Engineer
	4	Ann	Way	Technical Writer
	5	Liz	Gardner	Software Engineer
	6	Paul	Smith	Software Engineer

テーブルビューはその行と列全体がテーブル内の他の列と行に相対的に操作可能になるユニークな編集の利点を提供します。これにより列の値上の行の並べ替えなどの操作が可能になります。例えば、上のスクリーンショットでは、6個の Person 要素がGUI 操作により Last 子要素をベースに並べ替えられることができます。（*詳細に関しては下を参照してください*）このような操作はXML ノードセットを並べ替えるための通常の方法であるXSLT 変換を実行するよりもシンプルです。

メモ

- テーブル表示は同じ名前の要素のシーケンスに対してのみ適用することが可能です。
- テーブルの表示は単一の要素のために使用することができます。要素の**テーブルモード** アイコンは要素をクリックすると表示されます。
- オプションダイアログ内でテーブル表示の色を設定することができます: [フォントと色 | グリッドビュー | セルの色]1472。

テーブル表示内のビューと編集のためのアイコン

下に表示されるアイコンはテーブル表示内で使用可能なアイコンで、ビューと編集機能を提供します。これらはテーブルセルおよび/または グリッドビューツールバーで使用することができます。

	セルコンテンツに従いグリッド列の幅を最適化
/	色で表示されている場合 テーブルの表示613 がオンに切り替えられています。それ以外の場合はオフです。表示するためにクリックします。
/	テーブルの左側のセル内のコマンドをトグルします。行を列に、または列を行に切り替えます。
/	色で表示されている場合、テーブル、オブジェクト、または配列のためのフィルター617 が有効化されています。それ以外の場合フィルターは無効化されています。無効化/有効化するためにクリックします。式を編集するには、ダブルクリックします。
WRAP	セルをワードラップするためのトグルコマンド。選択されると、ワードラップがオンになります。デフォルトでは、アイテムのコンテンツのみがラップされます。アイテムの名前を追加でラップする場合、このオプションをアイコンのドロップダウンリストから選択します。
100%	グリッドビューのズームレベル
	テーブルビュー内で列ヘッダーが選択されると有効化されます。ボタンはテーブルの行を列のコンテンツの降順または昇順の順序にそれぞれ並べ替えます。

テーブル表示内の編集

テーブルの表示では以下で説明されている編集アクションを実行することができます。

テーブル行(テーブルの繰り返し要素の新規インスタンス)を追加します。
テーブルの繰り返し要素の他のインスタンスである新規の行を以下のように追加することができます:

1. 新規の行を追加する行の上または下の *行番号を含むセル*を右クリックします。
2. コマンド **後に挿入** (Ctrl+Enter) を選択して選択された行の下に行を追加、または **前に挿入** (Ctrl+Shift+Enter) を選択して選択された行の前に行を追加します。これらのコマンドは XML メニュー[1198] とグリッドビューツールバー[268] で使用することができます。

新規の行は要素ノードとして作成されます。希望する場合ノードの型を変更することができます *(詳細に関しては下を参照してください)*。

兄弟または子要素をテーブルセルに追加する
テーブルセルがテーブル行の子要素を表示している場合、この子要素に以下の兄弟ノードまたは子ノードを与えることができます。テーブルセルを右クリックしてコマンド **付加** (Ctrl+Alt+A) または **この追加** (Ctrl+Alt+Enter) をそれぞれ選択します。これらのコマンドは XML メニュー[1198] とグリッドビューツールバー[268] で使用することができます。新規の行は要素ノードとして作成されます。希望する場合ノードの型を変更することができます *(詳細に関しては下を参照してください)*。

要素内でセルをラップする
テーブルセルの周りに要素を作成することができます。セルとセルの親の間のレベルに新規の要素が作成されます。これを行うにはテーブルセルを右クリックしてコマンド **要素内でラップ** (Ctrl+Alt+W) を選択します。このコマンドは XML メニュー[1198] とグリッドビューツールバー[268] で使用することができます。

テーブル列を追加します(テーブルの繰り返し要素のすべてのインスタンスの新規の子ノード)
テーブルの繰り返し要素のすべてのインスタンスの新規の子ノードである新規の列を以下のように追加することができます:

1. 列ヘッダーまたは列の空ではないセルを右クリックします。
2. コマンド **後に挿入** (Ctrl+Enter) を選択して選択された行の右側に列を挿入、または **前に挿入** (Ctrl+Shift+Enter) を選択して選択された行の左側に列を追加します。これらのコマンドは XML メニュー[1198] とグリッドビューツールバー[268] で使用することができます。

新規の列は要素ノードとして作成されます。希望する場合ノードの型を変更することができます *(詳細に関しては下を参照してください)*。

列のノード型と名前を変更する
列のノード型を変更するには、列のノード型アイコンをクリックし、表示されるメニューから新規のノード型を選択します。このノードの型は繰り返し要素の全てのインスタンスのために変更されます。

列ヘッダーの名前を変更するには、名前をダブルクリックして編集します。この結果、このノードの名前は繰り返し要素の全てのインスタンス内で変更されます。

選択された列の値のテーブル行を並べ替える
自身の子ノード(列)の一つの相対的な値の上のテーブルの行を並べ替えることができます。例えば、アルファベット順に並べ替えられたテーブルの繰り返し要素を与えるために *LastName* 列をベースに行を並べ替えることができます。列を並べ替えるには、列ヘッダーを選択し、グリッドビューツールバー[268] コマンド **昇順で並べ替え** または **降順で並べ替え**をクリックします。これらのコマンドは XML メニュー[1198] 内でも使用可能です。

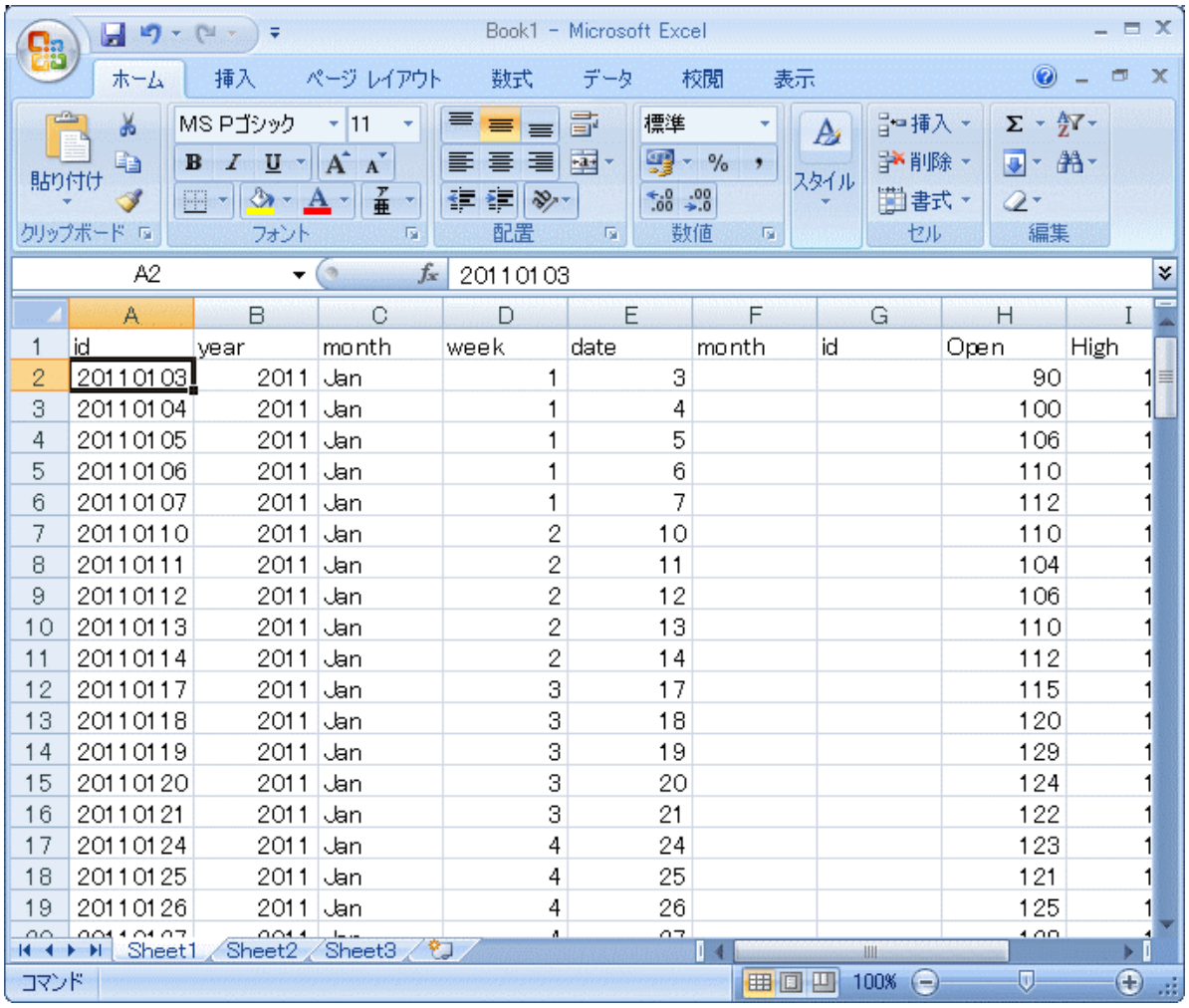
テーブル表示と外部アプリケーション

テーブル表示と(MS Excel などの)スプレッドシートアプリケーション間でデータを転送するためにテーブル構造を活用することができます。データの表示からデータを移動するには、以下を行います:

1. テーブルの表示内で、コピーするノードを選択します。セル自身、列ヘッダー、行ヘッダーまたはテーブル全体をクリックしてこれを行います。テーブル全体または列ヘッダーをクリックすると、列ヘッダーもコピーされます。それ以外の場合はコピーされません。(下のスクリーンショット内では、行 2 と行 3 が選択されています。)

expense-item (3)	type	expto	Date	expense	description
1	Lodging	Sales	2003-01-01	122.11	
2	Lodging	Development	2003-01-02	122.12	Played penny arcade
3	Lodging	Marketing	2003-01-02	299.45	Treated Clients

2. コンテキスト [コピー | タブで区切られたテキストとしてコピーする](1150) を選択します。
3. データを直接スプレッドシートプログラムに張り付けます。



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

データの転送は両方向に行うことができます。スプレッドシートからテーブルの表示にデータをコピーすることができます。以下の手順を行ってください:

1. 外部アプリケーション内で範囲を選択して(**Ctrl+C** を使用して)クリップボードにコピーします
2. テーブルの表示内で単一のセルを選択します。
3. コピーされたデータを **Ctrl+V** を使用して貼り付けます。

データは元の構造に対応し、テーブルビュー内で選択されたセルから開始する構造を持つテーブルに張り付けられます。以下の点に注意してください。

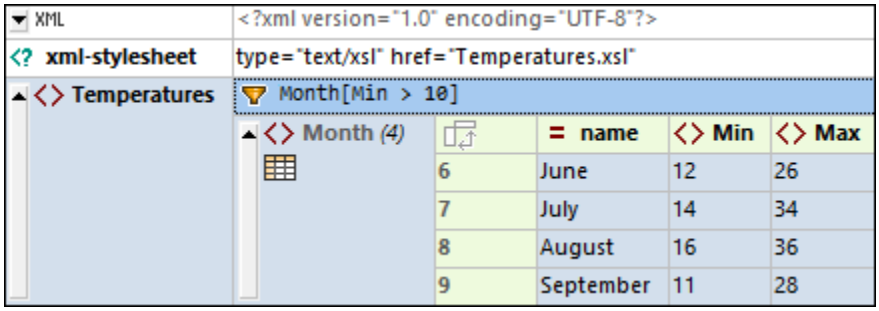
(C) 2015-2021 Altova GmbH

- データがテーブルビュー内のこれらのセルに既存の場合、新規のデータは元のデータを上書きします。
- 新規のデータに適用するために更なる行/または列が必要な場合、これらが作成されます。
- データは対応するセルにより表示される要素のコンテンツになります。

更に複雑なデータ転送タスクのために、XMLSpy のユニークな インポート/エクスポート 1331 関数を使用してください。

5.4.6 フィルター

XML グリッドビュー内のフィルターを要素ノードに適用することができ、これによりそのノードの子孫をフィルターできるようになります。フィルターが適用可能な都度、灰色表示されたフィルターアイコンが表示されます。フィルターが一度定義されると、このアイコンは色が付けられて表示されます(*下のスクリーンショット参照*)。フィルターは XQuery 3.1 式を使用して定義されます。例えば、下のスクリーンショットではフィルターは `Temperatures` ノードは `10` より大きい値を持つ `Min` 子要素を持つ `Month` 要素のみを表示するように設定されています。(スクリーンショット内では対応する要素インスタンスのインデックス番号がリストされています。)



フィルターをセットアップするには、フィルターする要素を右クリックして **フィルター** を表示されるコンテキストメニューから選択し Xquery 式を入力して **入力**をクリックします。*フィルター*セルはアイコン [▼] により示されています(*上のスクリーンショットを参照*)。*フィルター*アイコンをクリックしてフィルターを有効化/無効化することができます。

上のスクリーンショットで使用されている XML ドキュメントのリスティング

```
<?xml version="1.0" encoding="UTF-8"?>
<Temperatures>
   <Month name="January">
      <Min>-5</Min>
      <Max>3</Max>
   </Month>
   <Month name="February">
      <Min>-16</Min>
      <Max>1</Max>
   </Month>
   <Month name="March">
      <Min>-9</Min>
      <Max>7</Max>
   </Month>
   <Month name="April">
      <Min>2</Min>
      <Max>16</Max>
   </Month>
   <Month name="May">
      <Min>8</Min>
      <Max>21</Max>
```

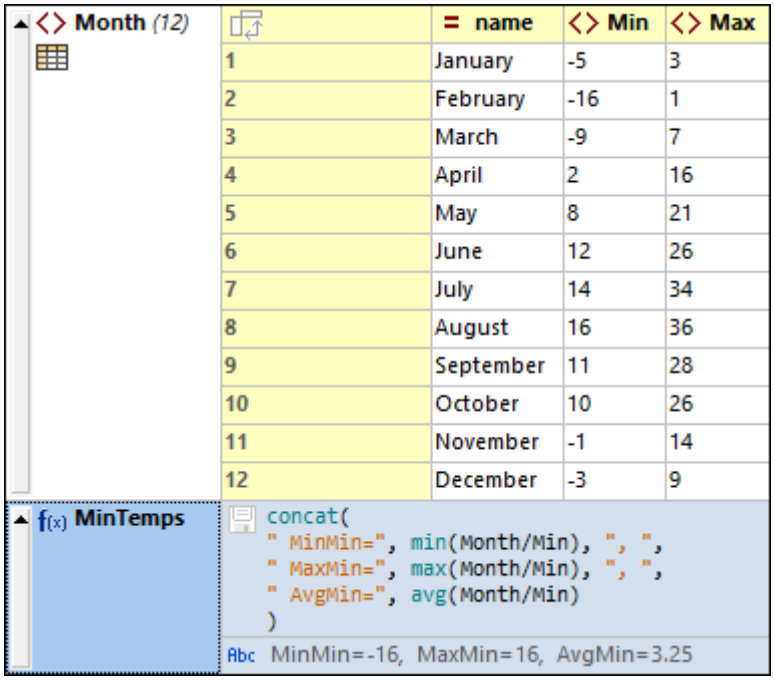
```
    </Month>
    <Month name="June">
       <Min>12</Min>
       <Max>26</Max>
    </Month>
    <Month name="July">
       <Min>14</Min>
       <Max>34</Max>
    </Month>
    <Month name="August">
       <Min>16</Min>
       <Max>36</Max>
    </Month>
    <Month name="September">
       <Min>11</Min>
       <Max>28</Max>
    </Month>
    <Month name="October">
       <Min>10</Min>
       <Max>26</Max>
    </Month>
    <Month name="November">
       <Min>-1</Min>
       <Max>14</Max>
    </Month>
    <Month name="December">
       <Min>-3</Min>
       <Max>9</Max>
    </Month>
</Temperatures>
```

フィルターに関する以下の点に注意してください：

- フィルターは要素ノードのみに適用することが可能です。
- フィルターのXQuery 式のコンテキストノードは現在のノードです。上のスクリーンショットでは、例えば、コンテキストノードは **Temperatures** ノードです。
- フィルターをネストすることが可能です。ネストされたフィルターは親フィルターのフィルターされたコンテンツに適用することができます。
- 式内に新規のラインを追加するには、「Ctrl+Enter」を押します。更に読みやすいようにするために式を複数行に表示するためにこの機能は役に立ちます。
- フィルターされたコンテンツは視覚的な表示のためだけのものです。実際のコンテンツは変更されずそのままです。
- 各フィルターは個別に実行され他のフィルターまたはフィルター[286]により影響されません。
- フィルターはXMLドキュメント内で保管されませんが、次回の使用のために特別なメタデータとして保管することができます。フィルターが保管されると、グリッドビュー内でドキュメントが再度開かれると自動的に適用されます。ツール | オプション | ビュー | グリッドビュー設定[1466]に移動して、フィルターを常に自動的に保管、フィルターをリクエストにより保管、またはフィルターを保管しないことを指定します。

5.4.7 フォーミュラ

結果を計算またはドキュメント内に保管されるノードセットを生成するためにXML グリッドビュー内のフォーミュラはXQuery 3.1 式を使用します。フォーミュラはXQuery 3.1 式を使用して定義されます。例えば、下のスクリーンショットでは最低、最高、および全ての最低気温のセットの平均気温を計算する **MinTemps** という名前のフォーミュラが作成されています。



Month (12)

	name	Min	Max
1	January	-5	3
2	February	-16	1
3	March	-9	7
4	April	2	16
5	May	8	21
6	June	12	26
7	July	14	34
8	August	16	36
9	September	11	28
10	October	10	26
11	November	-1	14
12	December	-3	9

f(x) MinTemps

```
concat(
  " MinMin=", min(Month/Min), ", ",
  " MaxMin=", max(Month/Min), ", ",
  " AvgMin=", avg(Month/Min)
  )
```

Abc MinMin=-16, MaxMin=16, AvgMin=3.25

フォーミュラを作成するには、以下を行います:

1. フォーミュラを表示する箇所に新規のノードを追加します。
2. ノードはデフォルトで要素として作成されます。ノードの型を[275]フォーミュラに変更します。
3. `f(x)` アイコンを含むセルをダブルクリックし、フォーミュラの名前を入力します *(上のスクリーンショットを参照)*。
4. このアイコンをクリックして、フォーミュラの出力をドキュメントに保存することができます。
5. 式のセルをダブルクリックして式内に入力し「Enter」をクリックします。

上のスクリーンショットで使用されている XML ドキュメントのリスティング

```
<?xml version="1.0" encoding="UTF-8"?>
<Temperatures>
   <Month name="January">
      <Min>-5</Min>
      <Max>3</Max>
   </Month>
   <Month name="February">
      <Min>-16</Min>
      <Max>1</Max>
   </Month>
   <Month name="March">
      <Min>-9</Min>
      <Max>7</Max>
   </Month>
   <Month name="April">
      <Min>2</Min>
      <Max>16</Max>
   </Month>
   <Month name="May">
      <Min>8</Min>
      <Max>21</Max>
```

```
    </Month>
    <Month name="June">
       <Min>12</Min>
       <Max>26</Max>
    </Month>
    <Month name="July">
       <Min>14</Min>
       <Max>34</Max>
    </Month>
    <Month name="August">
       <Min>16</Min>
       <Max>36</Max>
    </Month>
    <Month name="September">
       <Min>11</Min>
       <Max>28</Max>
    </Month>
    <Month name="October">
       <Min>10</Min>
       <Max>26</Max>
    </Month>
    <Month name="November">
       <Min>-1</Min>
       <Max>14</Max>
    </Month>
    <Month name="December">
       <Min>-3</Min>
       <Max>9</Max>
    </Month>
</Temperatures>
```

上のスクリーンショットで使用されているフォーミュラ式のリスティング

```
concat(
" MinMin=", min(//Month/Min), ", ",
" MaxMin=", max(//Month/Min), ", ",
" AvgMin=", avg(//Month/Min)
)
```

以下の点に注意してください：

- フォーミュラのXQuery 式のコンテキストノードはフォーミュラノードの親ノードです。
- 式内に新規のラインを追加するには、「Ctrl+Enter」を押します。更に読みやすいようにするために式を複数行に表示するためにこの機能は役に立ちます。
- （フォーミュラを定義する式である）フォーミュラはアプリケーション全体メタデータファイル内に存在する）ドキュメントのメタデータ内に保管されています。フォーミュラはXML ファイル内に処理命令としても保存されます。
- フォーミュラにより生成される結果はフォーミュラのXQuery 式の下のセル内に表示され、ドキュメントのメタデータ内に保管されます。結果はドキュメント内でも保管することができます。XQuery 式の横のディスクアイコンをクリックしてこれを行います。アイコンがクリックされているかに関わらず、フォーミュラの結果は計算されドキュメントのメタデータに保管されます。
- フォーミュラの結果が計算の場合、フォーミュラに割り当てられた名前を持つ要素のコンテンツとして保管されます。例えば、上のスクリーンショットに表示されている`MinTemps` フォーミュラでは、結果は`MinTemps` という名前が付けられている要素内に保管されます。
- ツール | オプション | ビュー | グリッドビュー設定 1466 に移動して以下を行います：(i) メタデータ内にフォーミュラを常に、リクエストに応じて保管するか、または決して保管しないかを指定する。(ii) ドキュメント内でフォーミュラを保管するか指定する。

テーブル内のフォーミュラ

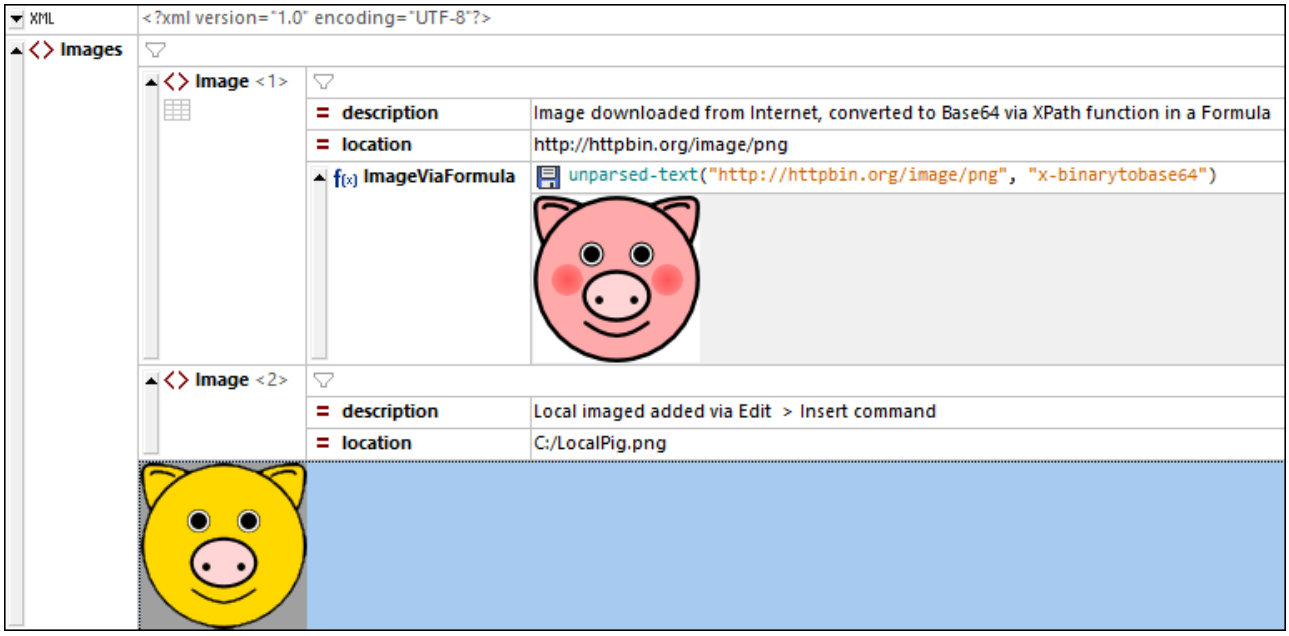
(テーブルディスプレイ[613]) 内のテーブル列のすべてのセルに同じフォーミュラが含まれている場合、フォーミュラは列のヘッダー内で一度のみ表示されます (*下のスクリーンショットを参照*)。フォーミュラ計算の*結果*はしかしながら対応するセル内に表示されます。列ヘッダー内のフォーミュラはグリッドビュー表示です。(テキストビュー参照) XML ドキュメント内で、フォーミュラは各テーブル行アイテムのために繰り返されています。

Month (12)		name	Min	Max	MidRange (Min + Max) div 2
	1	January	-5	3	# -1
	2	February	-16	1	# -7.5
	3	March	-9	7	# -1
	4	April	2	16	# 9
	5	May	8	21	# 14.5
	6	June	12	26	# 19
	7	July	14	34	# 24
	8	August	16	36	# 26
	9	September	11	28	# 19.5
	10	October	10	26	# 18
	11	November	-1	14	# 6.5
	12	December	-3	9	# 3

セルの単一フォーミュラが異なる場合でも、各フォーミュラは対応するセル内で表示されます。フォーミュラがヘッダーに表示されるようにテーブル列のすべてのフォーミュラが同じの場合、および個別のセルのために異なるフォーミュラを作成する場合、テーブルの表示をオフに切り替えて、そのセルのフォーミュラを編集します。テーブルディスプレイに切り替えると、フォーミュラは全てのセルのために個別のセルのために表示されます。

5.4.8 イメージ

イメージを直接グリッドビューで表示することができます (*下のスクリーンショット参照*)。これを行うには、イメージはBase64-エンコードで挿入される必要があります。



グリッドビューセルにBase64-エンコードイメージを挿入する方法が2通りあります:

- 型 *Formula* であるノードを作成します。フォーミュラ式セル内で *(上のスクリーンショット参照)* イメージを Base64 エンコードに変換する以下のXPath 式を入力します: `unparsed-text("<Image-URL>", "x-binarytobase64")`。XPath 関数 `unparsed-text` はイメージを Base64 エンコードに変換します。イメージはフォーミュラ式を含むセルの下に表示されます。
- イメージを追加するセル内にカーソルをポイントします。例えば、上のスクリーンショットでは、`Image` 要素を選択することによりイメージは2番目の`Image` 要素のエンテンツとして追加されます。[**編集 | 挿入 | エンコードされた外部ファイル**]を選択します。表示されるダイアログ内: (i) イメージへのパス(ローカルまたはインターネット)を入力します(ii) *Base 64* を選択します。(iii) *テキストの作成*を選択します。イメージはBase64 エンコードに変換され、エンコードされたテキストはセル内に挿入されます。しかしながら、エンコードではなくセル内のイメージが表示されます *(上のスクリーンショット参照)*。

URL を相対的パスを含む以下の書式で提供することができます:

- `http://httpbin.org/image/png`
- `file:///c:/LocalPig.png`
- `C:/LocalPig.png`
- `LocalPig.png`

一般的に使用されるイメージフォーマットはサポートされています。これらはPNG、JPEG、BMP、およびアニメーション GIF を含みます。SVG はXMLドキュメントとして読み取られます。グリッドビューではイメージは`<svg>` 要素の最後の子として表示されます。

5.4.9 チャート

`altovaext:chart` という名前のAltova XPath/XQuery 拡張子を使用してXML グリッドビュー内でチャートを作成することができます(*下のスクリーンショットを参照*)。拡張子は下で説明されています。これに関してはチャート関数 2053のセクション内で他のチャート共に説明されています。下のスクリーンショットで表示されている`altovaext:chart` 拡張子は グリッドビューフォーミュラ 286内で定義されているXQuery `Let` 式内で使用されています。チャートはフォーミュラの下でイメージとして表示されています。(下で与えられているXMLドキュメントリスティングとXquery 式を使用しチャート関数を試すことができます。)

XML <?xml version="1.0" encoding="UTF-8"?>

xml-stylesheet type="text/xsl" href="Temperatures.xsl"

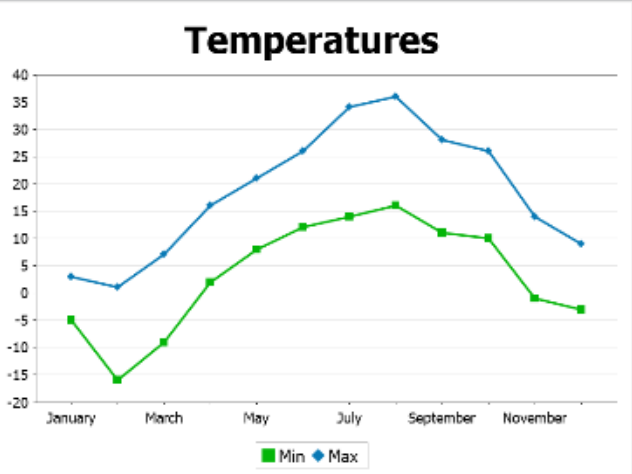
Temperatures

Month (12)

	name	Min	Max
1	January	-5	3
2	February	-16	1
3	March	-9	7
4	April	2	16
5	May	8	21
6	June	12	26
7	July	14	34
8	August	16	36
9	September	11	28
10	October	10	26
11	November	-1	14
12	December	-3	9

new_elem

```
let $months := //Month return
altovaext:chart(map{ "title": "Temperatures", "kind": "LineChart" },
(
    (:name, X-axis,        Y-axis :)
    ['Min', $months/@name, $months/Min],
    ['Max', $months/@name, $months/Max]
))
```



上のスクリーンショットで使用されている XML ドキュメントのリスティング

```
<?xml version="1.0" encoding="UTF-8"?>
<Temperatures>
   <Month name="January">
      <Min>-5</Min>
      <Max>3</Max>
   </Month>
   <Month name="February">
      <Min>-16</Min>
      <Max>1</Max>
```

```
    </Month>
    <Month name="March">
       <Min>-9</Min>
       <Max>7</Max>
    </Month>
    <Month name="April">
       <Min>2</Min>
       <Max>16</Max>
    </Month>
    <Month name="May">
       <Min>8</Min>
       <Max>21</Max>
    </Month>
    <Month name="June">
       <Min>12</Min>
       <Max>26</Max>
    </Month>
    <Month name="July">
       <Min>14</Min>
       <Max>34</Max>
    </Month>
    <Month name="August">
       <Min>16</Min>
       <Max>36</Max>
    </Month>
    <Month name="September">
       <Min>11</Min>
       <Max>28</Max>
    </Month>
    <Month name="October">
       <Min>10</Min>
       <Max>26</Max>
    </Month>
    <Month name="November">
       <Min>-1</Min>
       <Max>14</Max>
    </Month>
    <Month name="December">
       <Min>-3</Min>
       <Max>9</Max>
    </Month>
</Temperatures>
```

上のスクリーンショット内で表示されているチャートを生成するための XQuery 式のリスティング

```
let $months := //Month return
altovaext:chart(map{ "title":"Temperatures", "kind":"LineChart" },
(
    (:name, X-axis,       Y-axis   :)
    ['Min', $months/@name, $months/Min],
    ['Max', $months/@name, $months/Max]
))
```

Altova チャート拡張子の使用

- チャート拡張関数 `altovaext:chart` は名前空間プレフィックス `altovaext:` を使用する必要があります。
- `altovaext:chart` は2個の引数を取ります: (I) チャート構成情報と(ii) チャートデータ系列情報。
- チャート構成情報は `altovaext:chart` の最初の引数です。4個のキー値ペアの順序指定されていない系列です。これらのペアは以下のために存在します: (i) チャートのタイトル(キーは `title`)、(ii) 円グラフ、折れ線グラフなどのチャートの種類(キーは `kind` 使用可能な種類に関してはチャート関数[2053] を参照してください。)、(iii) ピクセルで表示されたチャートの幅(整数のみ。キーは `width` です)。および(iv) ピクセルで表示されたチャートの高さ(整数のみ。キーは `height` です)。幅または高さの値のどちらかが、または両方の値が与えられていない場合、不足する値または値はデータを基にして自動的に計算されます。
- チャートデータ系列は `altovaext:chart` の2番目の引数です。各データ系列はサイズ3の配列です: (I) 系列の名前 (ii) x-軸の値 (ii) y-軸の値(例えば、上のサンプルのように各系列がラインを表示する場合など) 複数の系列を作成する場合は複数の配列のシーケンスを作成します。上のサンプルには最低、最高、および平均機能のための3個のデータ系列が存在します。XとY 軸のためのデータはすべての `Month` 要素のシーケンスから参照されます。

詳細に関しては、チャート関数[2053] のセクションを参照してください。

5.4.10 グリッドビュー内のコンテキストメニュー

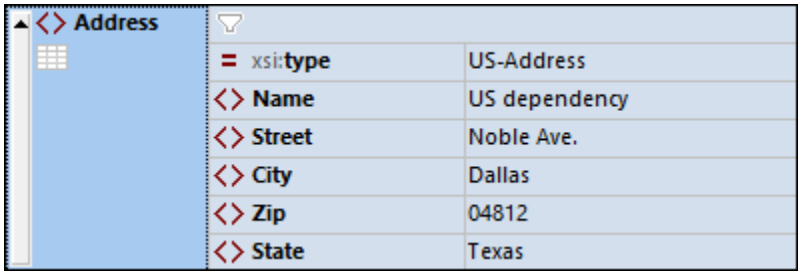
グリッドビュー内のセルを右クリックするとセルに関連したコンテンツを編集し、表示を変更するためのコマンドを提供するコンテキストメニュー(*下のスクリーンショット*) が表示されます。コンテキストメニューにキーボードの メニュー キーを押すことによりアクセスすることができます。コンテキストメニューのコマンドは下で説明されています。

型
型 コマンドにマウスをポイントすると[コンポーネントの型](#) 275 を選択することのできる表示されるサブメニューを表示します。キーボードショートカット **Ctrl+Menu** を使用して **Type** サブメニューにアクセスすることが可能です。

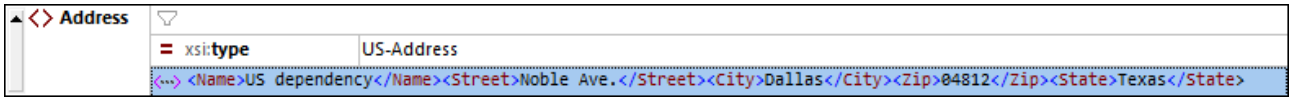
子を前/後に挿入、子の付加、子の追加
「**挿入**」と「**追加**」コマンドにより同じレベルでアイテム（デフォルトで要素）を追加することができます。**後に挿入** と **前に挿入**の場合新規のアイテムがそれぞれ選択されたアイテムの前後に追加されます。**付加** の場合、選択されたアイテムの最後の兄弟として追加されます。「**子の追加**」コマンドは新規のアイテム（デフォルトで要素）を子として追加します。名前セルをダブルクリックして編集し新規に追加された要素の名前を変更します。要素の（名前の左側の）アイコンをクリックして、必要とするノード型を選択しノード型を変更します。

要素内のラップ
（名前をダブルクリックして編集して）選択されたアイテムにデフォルトの要素を持つ親要素が与えられます。

生テキストとして編集
選択されたアイテムのテキストコンテンツを生テキストとして編集することが可能になります。HTML コードのような複雑なコンテンツを編集する場合などにこの機能は役に立ちます。例えば、下のスクリーンショットではグリッドビュー内の `Address` 要素を表示しています。



Address 要素を生テキストを表示する場合（*下のスクリーンショット*），グリッド構造はテキストビュー内同様に編集可能な生テキストを含む単一アイテムに変換されます。



要素空白文字
要素ノードで使用が可能です。**xml:space** 属性を要素のマークアップに追加し、コマンドのサブメニューから選択する値をこの属性に与えます。

- しかしながら *省略* 値を選択すると空白を正規化する効果がある **xml:space** 属性は追加されません。
- *保管* 値はすべての空白をそのままに保管し、その要素のために整形出力[131] をオフに切り替えます。保管された空白はグリッドビューでは省略記号のアイコンで表示されています。
- *デフォルト* 値はスキーマ内のその要素のために指定された値を取り、XML ドキュメント内の祖先要素から継承されている **xml:space** 値をオーバーライドするために役に立つ場合があります。

上/下/左/右への移動
コンポーネントを現在の位置からグリッド内で上下左右に移動しすることが可能になり、対応するコマンドが有効化されます。移動を実行するために対応するコマンド選択してください。

コピー
これらのコマンドは現在の選択項目をコピーするために使用されます。そのコンポーネントに許可されているオプションのみを使用することができます。

コマンド	説明
XML テキストとしてコピーする	現在の選択範囲はXML マークアップとしてシリアル化されています。
タブで区切られたテキストとしてコピー	現在のテーブル選択項目がTSV（タブ区切り値）としてシリアル化されます。
イメージとしてコピーする	現在のイメージセルがイメージとしてコピーされます

以下の点に注意してください：

- *イメージとしてコピー*は選択されたイメージのBase64 エンコード文字列をコピーします。（例えば、テーブルの表示内の他のセルなど）Base64-エンコード済み文字列がイメージとして表示されるドキュメントに文字列が張り付けられると文字列は標示されます。それ以外の場合は文字列として解析されます。
- テキストをセルに挿入するには、テキストをコピーしてセルに張り付けます。
- 例えばイメージなどをファイルからインポートするには、[編集 | 挿入 | エンコードされた外部ファイル][1154] コマンドを使用します。Base64 エンコード文字列としてイメージファイルを挿入し、この文字列をJSON グリッドビューで表示します。インポートのためにサポートされているイメージファイルフォーマットは以下の通りです: PNG、JPEG、BMP、GIF、TIFF。

XPath のコピー
選択されたノードをロケートするドキュメントルートから開始するPath 3.1 ロケーター式をクリップボードにコピーします。

XPointer のコピー
このコマンドは選択されたノードをロケートするXPointer 式をクリップボードにコピーします。詳細に関しては**XPointer/JSON-Pointer のコピー**[1154] を参照してください。

フィルター、フォーカスするフィルター
フィルター コマンドは切り替えコマンドです。選択された要素にフィルターを追加またはフィルターを無効化にします。このコマンドを使用してフィルターを追加後、フィルター式を入力する必要があることに注意してください。それ以外の場合フィルターは自動的に削除されます。フィルターが追加された後、コマンドをもう一度選択またはフィルターをクリックして無効化することができます。フィルターを削除するにはツールバー内で *全てのフィルターを削除する* をクリックします。

フォーカスするフィルター コマンドは(i) 選択されたノードをロケートするXPath 式を含むルート要素上にフィルターを作成します。そして(ii) 選択されたノード（とその子孫）を直接ルート要素の下で表示するためにグリッドビューをフィルターします。選択されたノードの表示のみに集中する場合この機能は役にたちます。フィルターを削除するには、ルート要素のコンテンツセル内のフィルターアイコンをクリックします。

フルに展開する
このコマンドは選択されているコンポーネントまたはその子孫が折りたたまれている場合に有効化されます。コンポーネントと子孫コンポーネントが展開されます。

選択されていない範囲を折りたたむ
このコマンドは選択されているコンポーネントとその祖先コンポーネント以外のすべてのコンポーネントを折りたたみます。

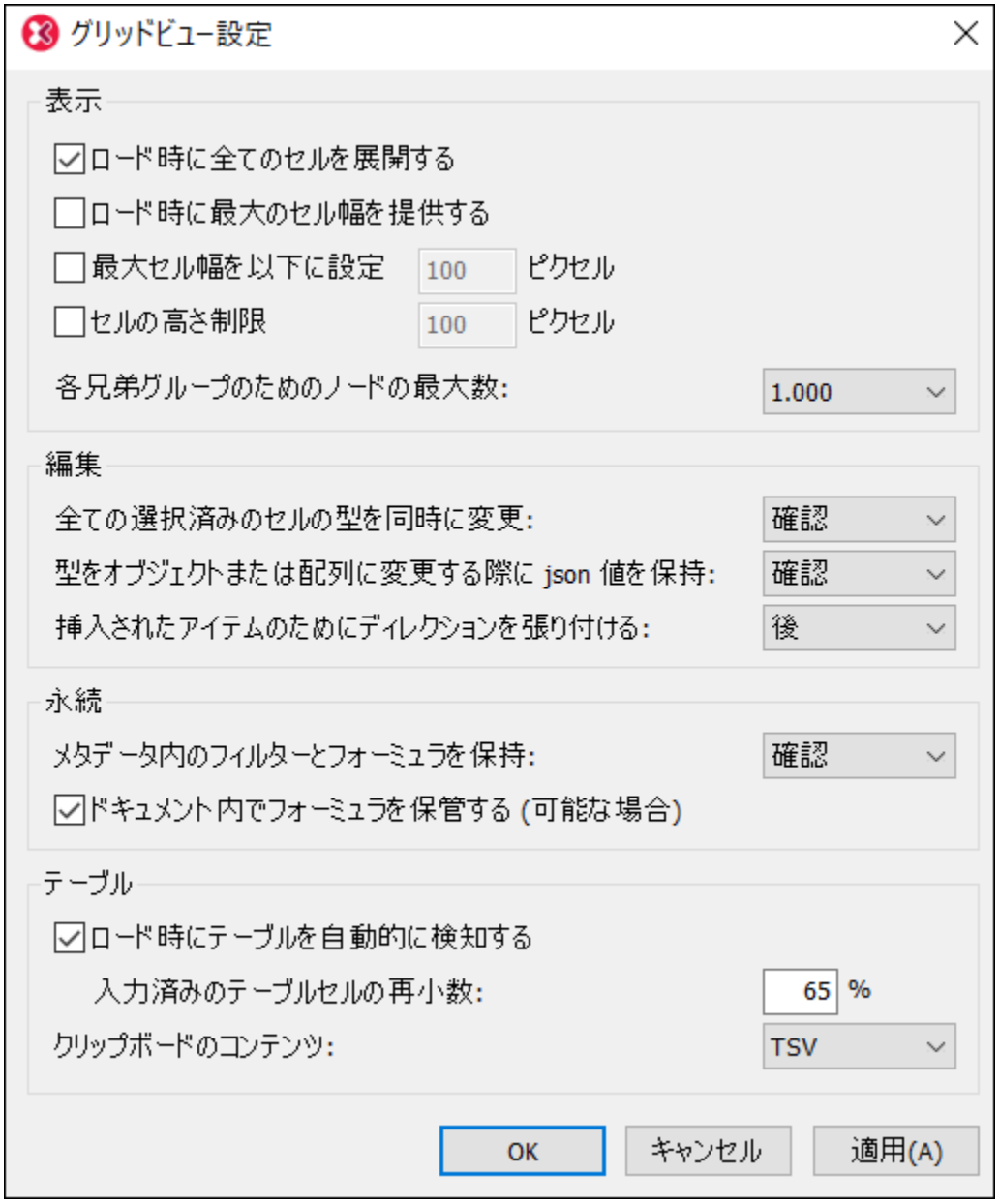
テキストの向き
このコマンドはテキストを含むノードのために使用することができ、読み取りの順序をセルの左側方向または右側方向に切り替えます。この機能はアラブ語とヘブライ語などの言語を使用する際に役に立ちます。

5.4.11 グリッドビュー設定

グリッドビューの設定はグリッドビュー設定ダイアログ（下のスクリーンショット）は[グリッドビューツールバー](268)内の設定アイコンを使用してアクセスすることができます。

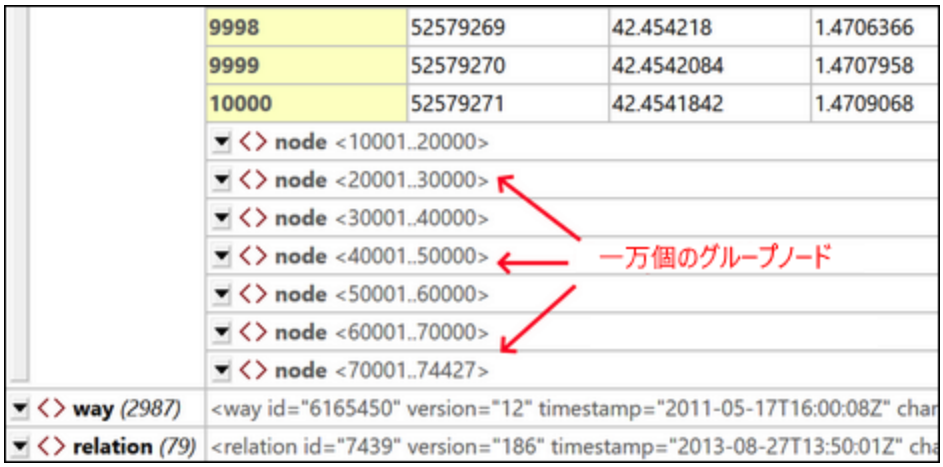
	[グリッドビューツールバー](268)内のグリッドビュー設定アイコン

グリッドビューの設定は以下で説明されています。全てのドキュメント（XML、JSON、DTD）のグリッドビューに適用することができることに注意してください。



表示
表示 セクション内のチェックボックスは自明です。以下は更に説明が必要な場合のメモです。

- ロードする際に全てのセルが展開されていない場合ルートノードとそのすべての子孫は折りたたまれています。ドキュメントを更に深く移動するためには各ノードを展開する必要があります。
- 最適な幅がオンに切り替えられている場合グリッドの幅全体が表示されます。これを達成するには、セルの一部のテキストがラップされます。
- 兄弟ノードは100、1000 または10000 ノードの兄弟グループで表示することができます（*下のスクリーンショットを参照してください*）。多数の兄弟ノードを持つファイル内でスペースの表示を省略する場合



編集

表示 セクション内のチェックボックスは自明です。以下は更に説明が必要な場合のメモです。

- 複数の選択されたセルの型を変更する場合、設定のアクションを継続するかを決定する以下のオプションが与えられます: (ユーザーにより決定される) *常に*、*なし* または *確認* 。
- JSON 型を動的な型からオブジェクトまたは列挙に変更する場合、設定のアクションを継続するかを決定するかを決定する以下のオプションが与えられます: (i) *確認* (動的な型の値が名前を持たない子 `key:value` ペアの値として保持される、または破棄されるかを確認) (ii) *常に* (名前を持たない子 `key:value` ペア内の値を保持する) (iii) *なし* (値を保持する。すなわち値を破棄する)。
- *ディレクションの貼り付け* オプションはクリップボード内の選択範囲が選択されたセルの上または下に張り付けられるかを決定します。

永続

フィルターとフォーミュラを XML と JSON ドキュメントのメタデータ内に保管することができます。フォーミュラは XML ドキュメント内の処理命令として、そして JSON ドキュメント内のコメントとして保管されます。これらの設定はフィルターとフォーミュラの保管の方法を指定します。フォーミュラの場合 *可能な場合* という用語はフォーミュラが実際にすべての JSON ドキュメント (JSON5 と JSONC ドキュメント内でのみ) のコメントとして保存できないことを意味します 。

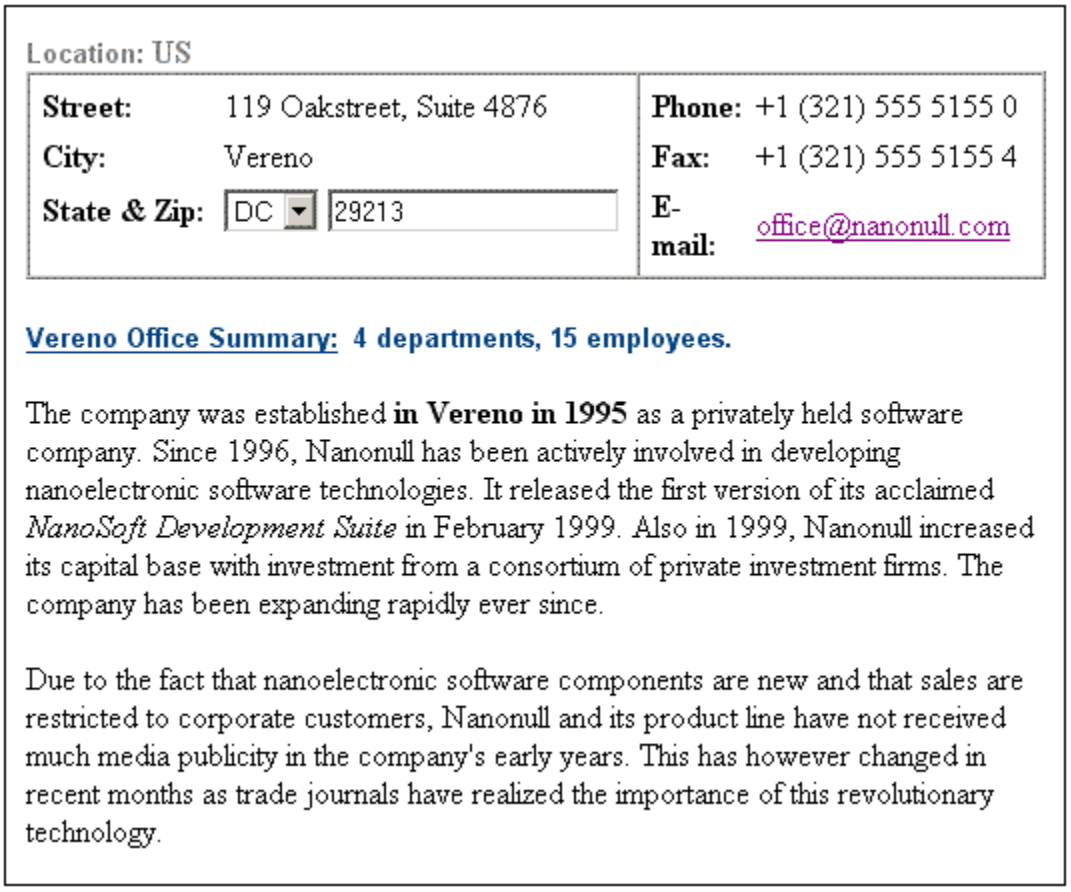
テーブル

以下の設定を使用することができます:

- 設定がロード時に JSON グリッドビューテーブルを自動的に検知する場合、テーブルをテーブルとして検知するために評価する記入されたテーブルセルの最小のパーセントを選択することができます。記入されたテーブルセルの数値がこのレベルを越さない場合、構造はそれぞれが下に表示されている繰り返し要素を持つ通常のグリッドとして表示されます。JSON グリッドビューのみにこの設定は適用されることに注意してください。XML グリッドビュー内では、テーブルセルの記入されている率に関わらずテーブルは自動的に検知されます。
- クリップボードコンテンツがタブにより区切られた値 (TSV) または (ドキュメントの型に従い) XML/JSON として保管するかを選択することが可能です。これはとても役に立つ機能です: クリップボードからテーブルを他のドキュメントに張り付ける場合、この設定により TSV としてまたはマークアップとしてテーブルがコピーされるかを選択できるようになります。(違いを確認するには、テーブルをそれぞれのフォーマットでクリップボードにコピーしテキストエディターにテーブルを張り付けてください。)

5.5 Authentic ビュー内の XML

Authentic ビューによりユーザーはテキストドキュメントのようにXMLドキュメントを編集することができます（*下のスクリーンショット*）。XML マークアップとすべての他の非コンテンツテキストはドキュメントの編集者に対して非表示になります。 cこの機能はドキュメントのコンテンツに集中しつつ有効なドキュメントの作成を可能にすることによりXML に慣れていないユーザーにとり役に立ちます。



StyleVision Power Stylesheet（SPS）がXMLドキュメントに割り当てられている場合ドキュメントのAuthentic ビューは有効化されます。XMLドキュメントがベースにされているスキーマソースにSPS はベースされておりXMLドキュメントの構造を定義します。SPS は Authentic ビューのドキュメントのレイアウトと書式設定を定義します。例えば、上のスクリーンショットに表示されるドキュメント内では、以下のAuthentic 書式設定と編集機能が使用されています:

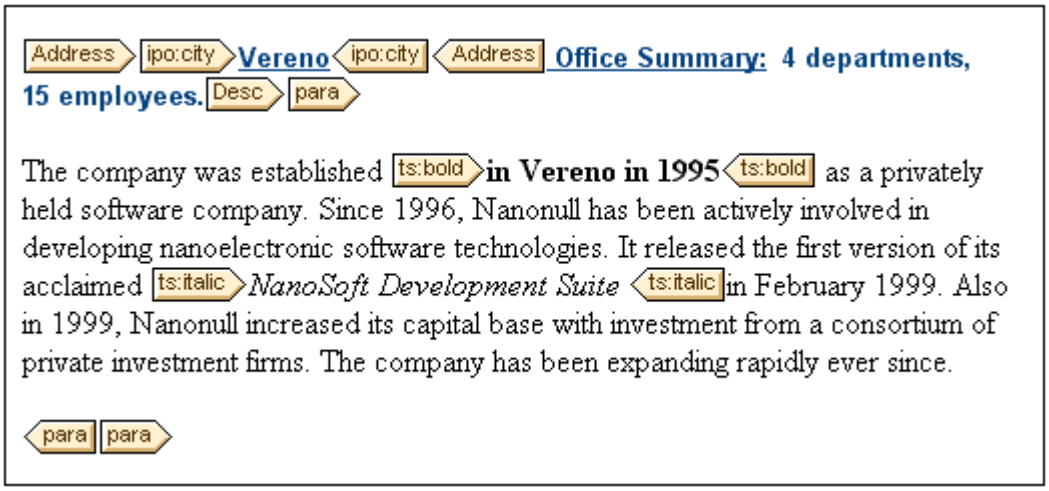
- パラグラフと他のブロック書式設定
- テーブル構造
- 色とフォントフェースなどのテキストの書式設定
- （State とZip フィールドにある）コンボボックスにより、妥当な選択肢の中から値を選択することができます。選択肢は上で記述されているように、スキーマ内にある 列挙 から取得されます。
- ドキュメント内のデータから計算を行ない表示することができます。上の例では、Office Summary の情報はユーザーによって入力されたものではなく、ドキュメント内のデータから計算して得られたものです。

SPS はXMLドキュメントをAuthentic ビューにて閲覧、そしてXML から（HTML、PDF、RTF、そしてW ord 2007 といった）標準出力を生成するためのファイルです。SPS ファイルはAltova StyleVision により作成されます。

ドキュメント構造の編集

位置を選択し、(要素や属性)入力ヘルパー、またはコンテキストメニューから目的のノードを選択することで、妥当なノードをいつでもドキュメントに挿入することができます。カーソルの位置を変更することで、選択できるノードがその場所において兄弟または子要素として妥当性を失わずに挿入することができるものに制限されます。例えば、カーソルがパラグラフ内に位置している場合、スキーまでそれが許されていれば、新たなパラグラフを追加することができます。

XML ドキュメントの構造を Authentic ビューで編集する場合、ドキュメントのマークアップを参照できれば便利です。メニューオプションの **Authentic | 大きなマークアップを表示** (または対応するツールバーアイコン)を使用することで(*以下のスクリーンショットにあるように*)マークアップをタグとして表示することができます。



コンテンツの編集

ドキュメントのノード内に入力することでコンテンツの作成や編集を行うことができます。エンティティや CDATA セクションはコンテキストメニュー(そしてエンティティはエンティティ入力ヘルパーからも)から追加することができます。

Authentic ビューにおける編集の詳細

Authentic ビューにおける編集に関する詳細については、Authentic ビューセクションを参照ください。

(C) 2015-2021 Altova GmbH

5.6 XML ドキュメントの入力ヘルパー

テキストビューと Authentic ビュー内の XML ドキュメントでは、3つの入力ヘルパーを使用することができます: 要素入力ヘルパー、属性入力ヘルパー、そしてエンティティ入力ヘルパーです。要素入力ヘルパーにより要素が追加された場合、オプションダイアログの編集セクション[1461]で指定された設定に従い、必須の子要素や必須の属性、または全ての子要素を同時に自動的に追加する(または全く追加しない)ことができます。空の属性が追加された場合、引用句に追加されます。

他のビューでは入力ヘルパーの表示も、そのビューが持つ機能に合わせて変化することにご注意してください。

要素入力ヘルパー

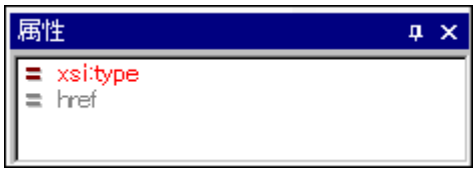
以下の点にご注意してください:

- *テキストビュー*: カーソルの挿入ポイントに要素は挿入されます。使用されていない要素は赤色で、使用されているものは灰色で表示されます。必須の要素には要素名の隣に感嘆符(!)が表示されます。
- *Authentic ビュー*: 選択された要素の後、前、そして中に要素を挿入することができます。更に、ドキュメント内で現在選択されている要素がドキュメントのツリー構造にて表示されます。Authentic ビューにおける編集に関する詳細については、Authentic ビューセクションを参照ください。

属性入力ヘルパー

以下の点にご注意してください:

- *テキストビュー*: カーソルが要素の開始タグ内にあるスペースの後にある場合、その要素に対して定義された属性が表示されます。使用されていない属性は赤色で、使用された属性はグレーで表示されます。必須の属性には属性名の隣に感嘆符(!)が表示されます。

  属性を挿入するには、必要とされる属性をダブルクリックしてください。カーソルがポイントされている個所で等号サインと属性の値を制限する引用符と共に属性が挿入されます。属性の値を直接入力できるようにカーソルを引用符の間をポイントしてください。
- *Authentic ビュー*: 要素が選択されているとき、その要素に対して宣言されている属性が表示されます。入力ヘルパーにて属性の値を入力してください。

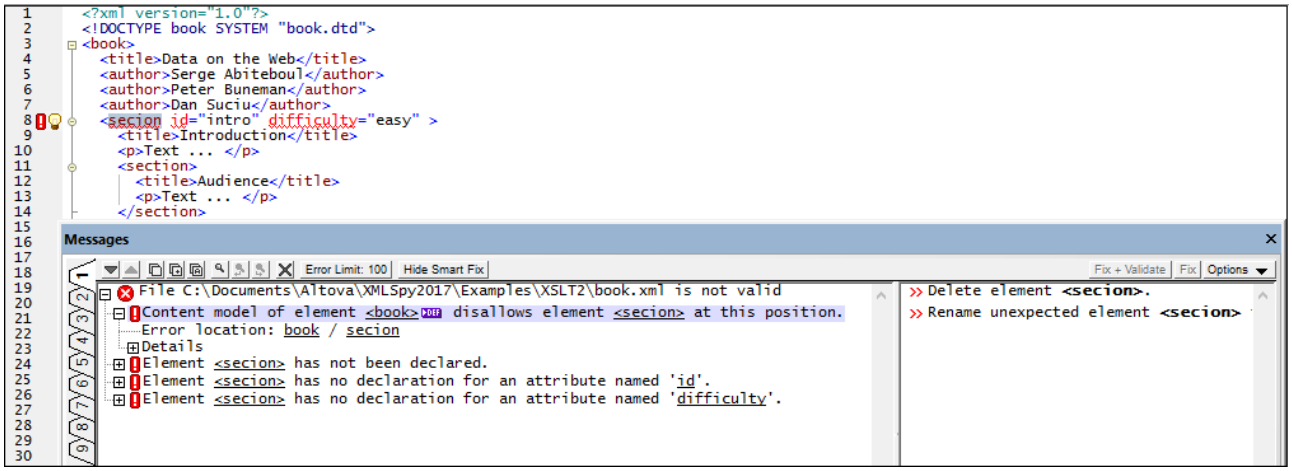
エンティティ入力ヘルパー

インライン(XML ドキュメント内)または外部 DTD で宣言されたエンティティはエンティティ入力ヘルパーに表示されます。3つのビュー(テキスト、グリッド、Authentic)にて、入力ヘルパー内のエンティティをダブルクリックすることで、そのエンティティがカーソル位置に挿入されます。グリッドビューでは、エンティティは追加、挿入、子を追加タブに表示されます。

内部エンティティを追加した場合、そのドキュメントを保存して再度開かないと、エンティティはエンティティ入力ヘルパーに表示されません。

5.7 XML ドキュメントの検証

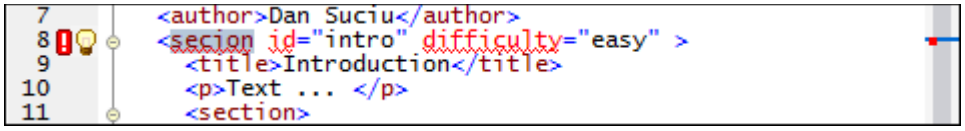
XML | 検証 (F8) コマンド は、ドキュメントを関連したDTD、XML スキーマ、または、他のスキーマに対して検証します。ドキュメントが妥当な場合、検証に成功したことを示すメッセージがメッセージウィンドウに表示されます。それ以外の場合、左側のペイン内にエラーの原因が表示されます (*下のスクリーンショットを参照してください*)。左側のペインで原因が選択されている場合、使用できる場合は、右側のペインに表示されているスマートフィックスを使用します。スマートフィックスの提案は、関連付けられたスキーマ内の情報をベースにしています。スマートフィックスを適用するには以下を行います (i) ダブルク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



エラーインジケーターとスマートフィックス

エラーインジケーター

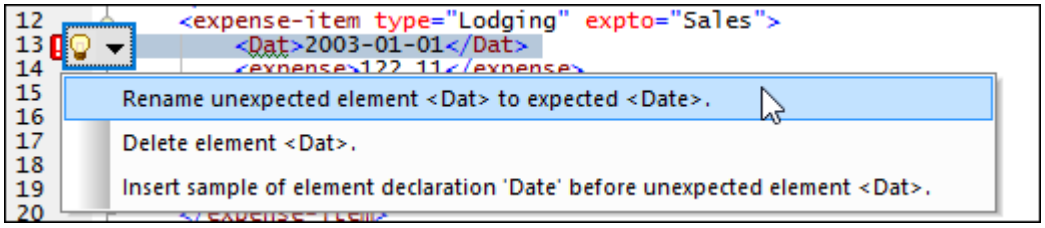
テキストビューでは検証エラーの追加インディケーターが2つ存在します (*下のスクリーンショット参照*): (I) ラインの付番余白内の赤い感嘆符 (ii) (ウィンドウの右側の) 赤色のマーカー正方形。



赤い四角はウィンドウの右側にあるスクロールバーの左横に表示されることに注意してください (*上のスクリーンショットを参照*)。スクロールバーは次の個別の2つの種類のマーカーを表示するためここで説明されています: (I) ハイライトされたテキストの発生個所 (茶色、スクロールバーの左側。*ドキュメントのナビゲート* 137)を参照してください。(ii) 発生個所を確認してください (茶色、スクロールバーの右側 *検索と置き換え* 140 を参照)

スマートフィックス

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



以下の点に注意してください:

- 検証エラーのインジケーターとクイックフィックスはXMLSpy 内で検証されるドキュメントの型、例えばJSON ドキュメントなど、に対して使用することができます。
- 上記の検証エラーインジケーターとスマートフィックスは、**XML** | **検証** (F8) コマンドが実行された場合のみ更新されますが、背景では更新されません。ですから、エラーを修正した後、**検証** (F8) コマンドを再度実行し、エラーが修正されていることを確認してください。

XML ドキュメントの検証に関する詳細は、以下を参照してください: [検証](#) 1227 コマンド。

編集中の検証

編集中に検証 モードはデフォルトでオンに切り替えられています。オンに切り替えられていると、テキストビューと(JSON グリッドビューの)ドキュメントの変更中に整形式チェックと検証チェックが実行されます。(整形式のチェックに加え) 検証を実行するには、DTD またはXML スキーマがXML ドキュメントに割り当てられている必要があります(JSON スキーマはJSON ドキュメントに割り当てられている必要があります)。エラーは左側の余白内のエラーを含むライン上でエラーインジケーターにより表示されています(上記参照)。

編集中に検証 モードは(i) [「XML | 編集中に検証」](#) 1233 メニューコマンド (ii) **編集中に検証** ツールバーボタン または(iii) [オプションダイアログの検証設定](#) 1455 の *編集時* オプションによりオンとオフに切り替えることができます。

5.8 XML フラグメントの挿入

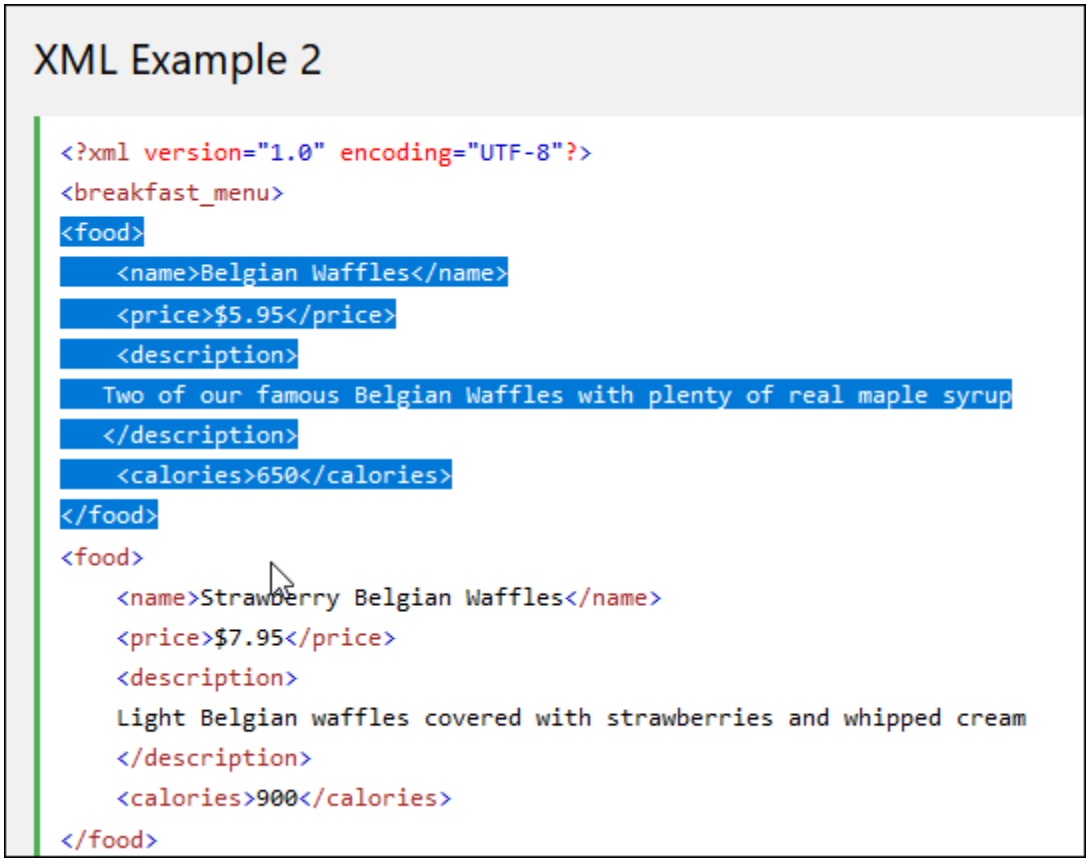
他のアプリケーションとW eb ページからXML フラグメントを挿入することができます。2つの方法を使用してこれらのフラグメントを挿入することができます:

- ドラッグアンドドロップを使用してテキストビューとグリッドビューに挿入する。グリッドビューにドラッグアンドドロップする場合、ドラッグオーバーレイ内で使用可能なインテリジェントな情報 278 がフラグメントをドロップする箇所を決定する手助けをします。
- コピーして貼り付けを使用してテキストビューとグリッドビューに挿入する。

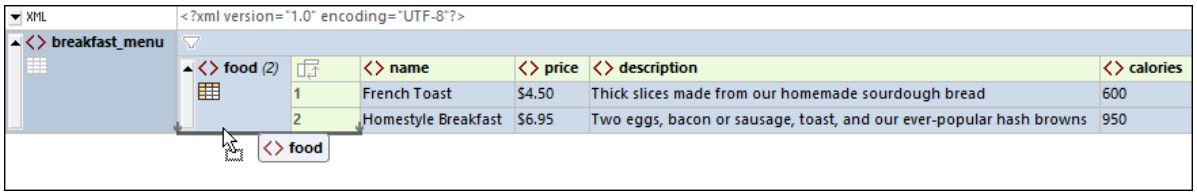
サンプル

次のサンプルはフラグメントをXML ドキュメント内の正確な箇所に素早く追加する方法を示しています。

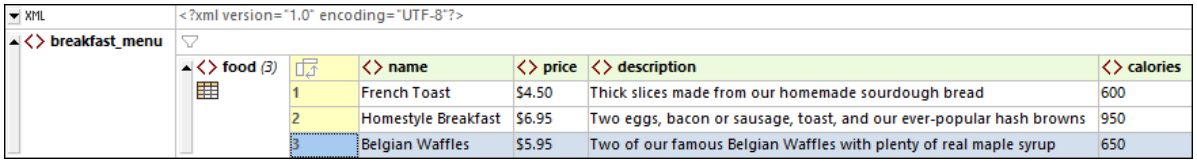
1. 下でハイライトされているフラグメントが(w3schools.com でのXML チュートリアルから) 選択されています。子要素の数値を含む food という名前の要素です。



2. 下のスクリーンショットは2個のfood 要素がテーブルの行として表示されているXML ドキュメントのグリッドビューを示しています。W eb ページからXML フラグメントがfood テーブルにドラッグされる際、ドラッグされたフラグメントがfood 要素としてfood テーブルに最後の行として挿入される情報を含むドラッグオーバーレイ 610 が表示されます。



3. フラグメントがドロップされると、`breakfast_menu` の `food` 子要素として希望する箇所に配置されます（*下のスクリーンショット参照*）。



5.9 XSLT と XQuery による処理

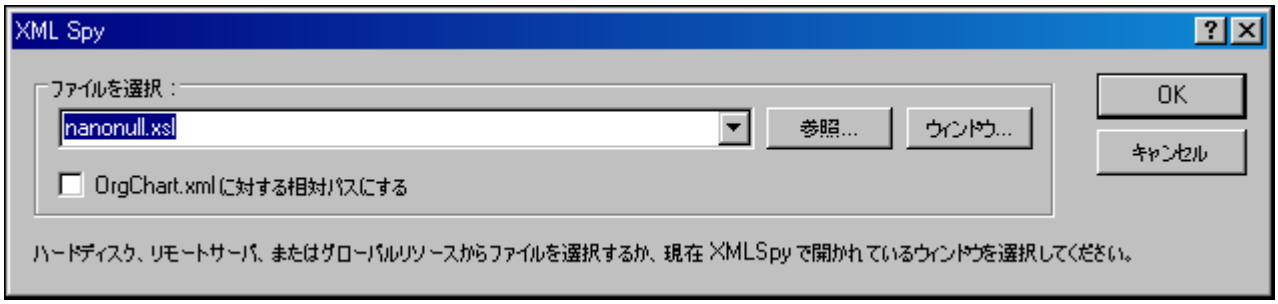
XML ドキュメントと XSLT または XQuery ドキュメントを組み合わせることで、出力ドキュメントを生成することができます。XMLSpy には XSLT 1.0、XSLT 2.0、そして XQuery 1.0 プロセッサーが内蔵されており、GUI から以下のような機能を利用することができます:

- XSLT スタイルシートの割り当て 306
- XSLTへ移動 306
- XSLT パラメーターとXQuery 変数 306
- XSLT 変換 307
- XQuery 実行 307
- RaptorXMLを使ったXML 処理の自動化 307

XSLT スタイルシートの割り当て

「XSLT/XQuery | XSL の割り当て」コマンドを使用(以下のスクリーンショットにあるように、ダイアログにてファイルを指定)して、XSLT スタイルシートを XML ドキュメントに割り当てることができます。割り当ては W 3C により定義された `xml-stylesheet` 処理命令 (PI) として XML ドキュメント内に記述されます。
XSLT 変換が呼び出された際(「XSL/XQuery | XSL 変換」)に、この割り当ては使用されます。

更に、「XSL/XQuery | XSL:FO 変換」コマンドにより(以下のスクリーンショットにあるようにダイアログにてファイルを指定して) XSLT-FO スタイルシートを割り当てることもできます。割り当ては Altova により定義された altova_ xslfo 処理命令 (PI) として XML ドキュメント内に記述されます。
XSLT-FO 変換が呼び出された際(「XSL/XQuery | XS:FO 変換」)に、この割り当ては使用されます。



XSLT ファイルのソースとしてグローバルリソースを選択することもできます。グローバルリソースはファイルまたはフォルダーに対するエイリアスです。アクティブなグローバルリソースの構成をマウス操作(メニューコマンド「ツール | アクティブな構成」)により変更することで、ターゲットファイルまたはフォルダーを変更することができます。グローバルリソースを使うことで、割り当てられた XSLT ファイルを切り替え、試験などに利用することができます。詳しいグローバルリソースの使用方法に関しては、Altova グローバルリソース 934 の項を参照ください。

以前の割り当てにより作成された上記の処理命令が存在する場合、それら既存の割り当てを上書きしても良いか確認されます。

XSLT へ移動

「XSL/XQuery | XSL へ移動」コマンドにより、XML ドキュメントに割り当てられた XSLT ファイルを開くことができます。

XSLT パラメーターとXQuery 変数

「XSL/XQuery | XSLT パラメーター/XQuery 変数」コマンドを選択することで表示されるダイアログにより、XSLT パラメーターならびに XQuery 変数の定義、編集、そして削除を行うことができます。ここで定義されたパラメーター、変数の値は、XMLSpy における全ての XSLT 変換ならびに XQuery 実行により使用されます。しかし、これらの値は MSXML の様な外部エンジンには受渡されません。この機能の詳細については、ユーザーリファレンス 1281 の項を参照ください。

(C) 2015-2021 Altova GmbH

XSLT 変換

2種類のXSLT 変換を利用することができます:

- 標準 XSLT 変換(「XSL/XQuery | XSL 変換」):変換の出力は新規ウィンドウに表示されるか、スタイルシートで指定されている場合、指定された場所にファイルが保存されます。変換のために使用されるエンジンはオプションダイアログ(「ツール | オプション」[1454])のXSL タブ[1479]にて指定することができます。
- XSL-FO 変換(「XSL/XQuery | XSL:FO 変換」):XML ドキュメントは2段階の処理によりPDF へと変換されます。最初のステップでは、オプションダイアログ(「ツール | オプション」[1454])のXSL タブ[1479]で指定されたXSLT プロセッサーによりXML ファイルがFO ドキュメントに変換されます。オプションタブの下部にてFOP の様なFO プロセッサーに同封されたXSLT エンジンも使用することができます。2番目のステップでは、オプションダイアログ(「ツール | オプション」[1454])のXSL タブ[1479]で指定されたFO プロセッサーによりFO ドキュメントからPDF 出力が生成されます。

注意: XSL: FO 変換コマンドにより、(XML ドキュメントの一種である)FO ドキュメントをPDF に変換することができます。変換のソースドキュメントがFO ドキュメントである場合、上で紹介された2 段階処理の2 番目のステップが直接実行されます。

XQuery 実行

「XSL/XQuery | XQuery 実行」コマンドをクリックすることで、アクティブな XML ドキュメント上でXQuery ドキュメントを実行することができます。XQuery ファイルを入力するよう促され、実行結果のドキュメントは新しいウィンドウに表示されます。

RaptorXML を使ったXML 処理の自動化

Altova RaptorXML はXML 検証、XSLT 変換、およびXQuery 実行を搭載したアプリケーションです。コマンドライン、COM インターフェース、Java プログラム、そして.NET アプリケーションから使用することができ、XSLT 変換タスクはRaptorXML により自動化することができます。例えば、RaptorXML によりドキュメントのセットを変換するバッチファイルを作成することができます。詳細については RaptorXML ドキュメンテーション を参照ください。

5.10 PDF フォント

フォーマッターとPDF ビューアーに対するフォントの使用
(例えばFOP といった) フォーマッターによりPDF が作成され、PDF ビューアー(例えばAdobe Reader)にて確認することができます。

PDF のレイアウトを行うには、ドキュメント内で使用されているフォント、その中でも特に、使用されている文字の幅に関する詳細をフォーマッターへ伝える必要があります。この情報を使用することで行の長さやハイフンの位置、行端揃えなどの計算が行われます。この情報はフォントメトリックと呼ばれ、各フォントとともに保管されます。フォーマッターによっては、システムのフォントフォルダーからメトリックを直接読み込むことができますが、(FOP のような) それ以外のフォーマッターに対しては、定められた形式でメトリックを渡す必要があります。フォントのメトリックがフォーマッターに対して与えられると、期待されるかたちでPDF のレイアウトが作成されることになります。そのため、ドキュメントにて使用されている全てのフォントに対するメトリックファイルが、使用しているフォーマッターに対して渡されるようにする必要があります。

フォーマッターはフォントを参照することができるほか、PDF ファイル内部にフォントを埋め込むことができます。フォントが参照された場合、PDF ビューアー(例: Adobe Reader)は通常(base 14 フォントが含まれる) 独自のリソースフォルダーからそのフォントの検索を行い、フォントがそこに無い場合はシステムのフォントフォルダーを検索します。フォントが利用できる場合、PDF が表示される際にそのフォントが使用されます。フォントが見つからない場合、リソースフォルダー上にある代替フォントが使用されるか、エラーが表示されます。代替フォントには異なるメトリックが指定されている場合もあり、表示エラーが生成される可能性もあります。

フォーマッターによりフォントがPDF ファイル内部に埋め込まれた場合、埋め込まれたフォントがPDF ビューアーにより使用されます。フォーマッターはフォントの全文字セットを埋め込むことができるほか、ドキュメント内部で使用されているフォントのサブセットだけを埋め込むことができます。フォントの埋め込み方法により、PDF ファイルの大きさが変わってくるだけではなく、フォントの使用に関する版権にも影響を与える可能性があります(以下のメモを参照ください)。フォーマッターのオプションを指定する際に、これらオプションの選択を指定できる場合があります。

XMLSpy とPDF フォント

XMLSpy では、XSL-FO ドキュメント(以下 FO ドキュメント)からPDF が生成されます。FOP といった外部 FO プロセッサーにより、XSL-FO ドキュメントが処理されます。オプションダイアログでは、FO プロセッサーの場所を指定することができます。これによりXMLSpy のGUI からFO の処理を開始することができます。

XSL-FO ドキュメントそのものはXML ドキュメントをXSLT スタイルシートで処理することにより生成されます。(XMLSpy に搭載されている) Altova のXSLT エンジンを使用することができるほか、外部 XSLT エンジンを使用することができます。

全テキストのフォントプロパティを含むPDF ドキュメントのフォーマットは、XSL-FO ドキュメントにて指定されます。使用しているフォーマッターがフォントのメトリックをフォントから直接読み取ることができる場合、フォントにアクセスするためにフォーマッターのセットアップを行う必要があります。FOP をフォーマッターとして使用している場合、base-14 フォントに含まれないフォントに対して、適切なフォントメトリックファイルを指定する必要があります。

フォーマッターがフォントを利用できるようにする

(FOP を含む) ほとんどのフォーマッターでは既にbase 14 フォントを利用することができ、これらのフォントをフォーマッターへ指定するために使用される名前を知ることが重要になります。base 14 フォントはフォーマッターによりサポートされる基本的なフォントですが、フォーマッター固有の方法に従うかたちで、これらのフォント以外にもフォーマッターが扱うことのできるフォントの数を増やすことができます。以下ではFOP にてフォントの数を増やす方法について記述されます。

FOP に対して新たなフォントサポートを追加するためのセットアップ手順

以下のステップにより、FOP が使用することのできるフォントを追加することができます:

1. PostScript またはTrueType フォントファイルから、フォントのメトリックファイルを生成します。FOP に搭載されているPFM Reader ならびにTTF Reader ユーティリティにより、PostScript ならびにTrueType フォントをXML フォントメトリックファイルへ変換することができます。詳細については、FOP: Fonts ページを参照ください。
2. FOP 構成ファイルをセットアップすることで、目的のフォントメトリックファイルを使用することができるようになります。FOP 構成ファイルにてフォントファイルに関する情報を入力することになります。FOP: Fonts ページを参照ください。
3. `fop.bat` ファイルにて、以下にある最後の行を:

```
"%JAVACMD%" […] org.apache.fop.cli.Main %FOP_CMD_LINE_ARGS%
```

構成ファイルの場所を含むように変更します:

```
"%JAVACMD%" […] org.apache.fop.cli.Main %FOP_CMD_LINE_ARGS% -c conf\fop.xconf
```

メトリックファイルがFOP（FOP 構成ファイル）にて登録され、FOP 実行ファイルが構成ファイルを読み取るようにセットされることで、指定されたフォントをPDF の作成に使用することができるようになります。

FOP 構成ファイルのセットアップ

FOP 構成ファイルは `fop.xconf` という名前で、FOP インストールフォルダー内にある `conf` フォルダー以下に配置されています。XML 形式により記述されるこのファイルを編集することで、FOP によりフォントメトリックファイルが適切に読み取られるように指定する必要があります。ドキュメント内部にある `font` 要素プレースホルダにて示される場所に、レンダリングを行う各要素に対応した `font` 要素を追加する必要があります:

```
<font metrics-url="arial.xml" kerning="yes" embed-url="arial.ttf">
          <font-triplet name="Arial" style="normal" weight="normal"/>
          <font-triplet name="ArialMT" style="normal" weight="normal"/>
</font>
```

上の例は、以下のように理解することができます:

`arial.xml`	メトリックファイルのURL です。絶対パスの使用が推奨されます。
`arial.ttf`	TTF ファイルの名前です（通常 `%WINDIR%\Fonts` 以下に収められています）。
`Arial`	フォントファミリーに`Arial` が指定されている場合、上記のメトリックとTTF ファイルが使用されるように指定します。
`style="normal"`	フォントスタイルが（`italic` などではなく）`normal` として定義されている場合、上記のメトリックとTTF ファイルが使用されるように指定します。
`weight="normal"`	フォントウェイトが（`bold` などではなく）`normal` として定義されている場合、上記のメトリックとTTF ファイルが使用されるように指定します。

フォントの版権に関するメモ: フォントの使用は著作権法の対象となり、使用に関する条件は一定ではありません。フォントの埋め込みを行う場合、特にフォント全体の埋め込みを行う場合、購入したフォントをそのように使用しても問題が無いか確認する必要があります。

文字セット

フォントの文字セットも考慮する必要がある点に注意してください。base 14 フォントでは、ISO-8859-1 により規定される文字の他に、シンボルとZapf Dingbats フォントにある文字がカバーされます。base 14 フォントによりカバーされない文字がドキュメントに含まれる場合、その文字が含まれる文字セットを含んでいるフォントを使用する必要があります。Arial Unicode のようなフォントには、Unicode によりカバーされる文字が含まれます。

5.11 チャート

XMLドキュメントがテキストビューまたはグリッドビューで開かれている場合、XMLドキュメント内のデータを、出力ウィンドウ[108]の1つであるチャートウィンドウに(円グラフや棒グラフのような)チャートという形で出力することができます。チャートはイメージファイルやXSLT、XQueryフラグメントという形でクリップボードへエクスポートすることができます。チャート機能を使うことで、XMLドキュメント内で選択した数値データを、素早く視覚的に表示することができます。

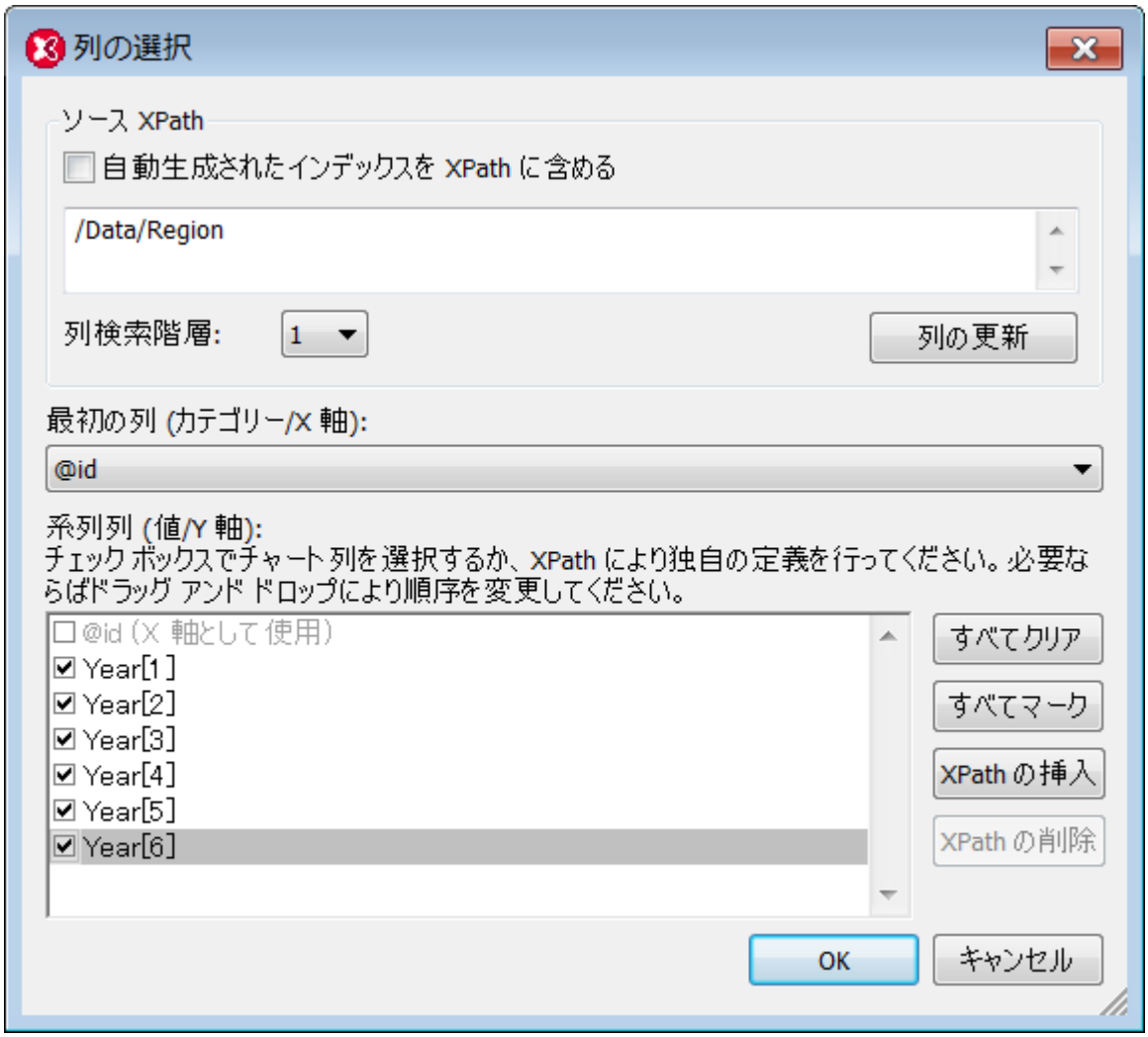
以下の種類のグラフが利用できます:

- 円グラフ(2D、3D)
- 棒グラフ(2D、3D)
- 棒グラフ、グループバー(2D、3D)
- スタックされた棒グラフ
- 折れ線グラフ
- 値付き折れ線グラフ
- エリアチャートならびにスタックされたエリアチャート
- ローソク足チャート
- ゲージチャート(円ならびに棒)
- オーバーレイチャート

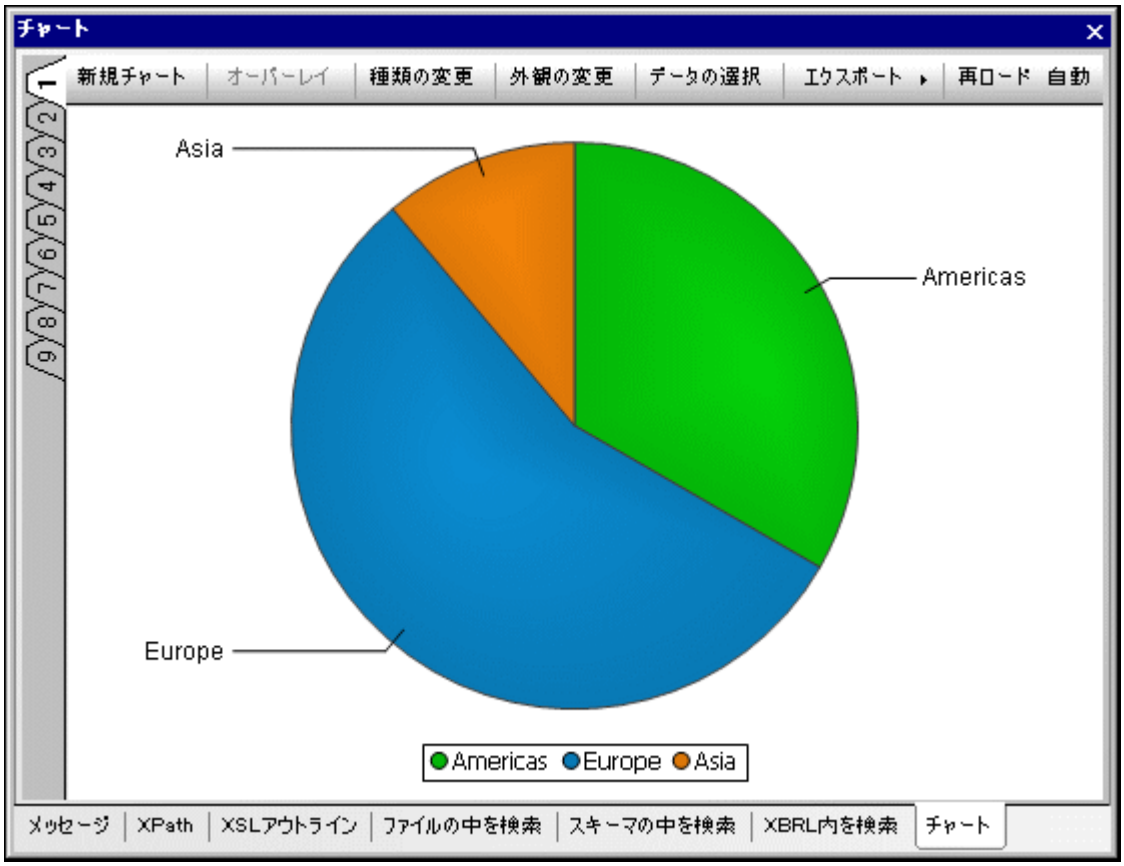
概要:作成からエクスポートまで

チャート作成を行うステップの概要を以下に記します。より詳細な情報については、このセクションのサブセクションを参照ください。

1. テキストビューまたはグリッドビューにて、データ選択のコンテキストノードとして使用したいノードを選択します。ノードの範囲を選択することも可能です。選択の方法については[ソースXPath](317)のセクションを参照ください。
2. 右クリックして表示されるコンテキストメニューから、「**新規チャート**」を選択します。その他にも、チャート出力ウィンドウにて「**新規チャート**」ボタンをクリックすることもできます。この操作により列の選択ダイアログが表示され、[X-軸](320)、[Y-軸](325)データを選択し、[ソースXPath](317)を修正することができます。



3. 「OK」をクリックすることで、チャート ウィンドウにチャートが作成されます(以下のスクリーンショットを参照)。



4. チャートのデータ選択などの設定は、チャートの作成後に編集することができます。ソースXPath 列選択の編集だけではなく、チャートの種類や外観などを指定することもできます。チャートの軸に対するデータの選択は「**データを選択**」ボタンをクリックすることで行うことができます。チャートの種類や外観は、それぞれ「**種類の変更**」ボタンと「**外観の変更**」ボタンにより修正することができます。
5. チャートはイメージファイル、またはXSLT、XQuery フラグメントとしてクリップボードへエクスポートすることができます。

その他の機能

以下の機能により、グラフの使用をより簡単に行うことができます:

- *複数タブ*: 現在のチャートを削除すること無く新規チャートを作成したい場合、(上のスクリーンショットにあるように) 9つあるタブの内、使用されていないタブを選択して新規チャートの作成を行うことができます。XML ドキュメントが閉じられた場合でも、そのドキュメントから生成されたチャートは、チャートウィンドウのタブ内に存在し続けることに注意してください。
- *自動再ロード*: (上のスクリーンショットで示されている)「**自動**」ボタンを有効にすることで、XML ドキュメントのデータが変更されたときにチャートも自動的に再ロードされます。このボタンが有効になっていない場合、「**再ロード**」ボタンをクリックして、手動でチャートの更新を行う必要があります。

サンプルファイル

このセクションならびにサブセクションでは、チャートの動作説明を行うために `YearlySales.xml` XML ファイルが使用されます。このファイルは `C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples¥Tutorial` フォルダーからご利用になれます。

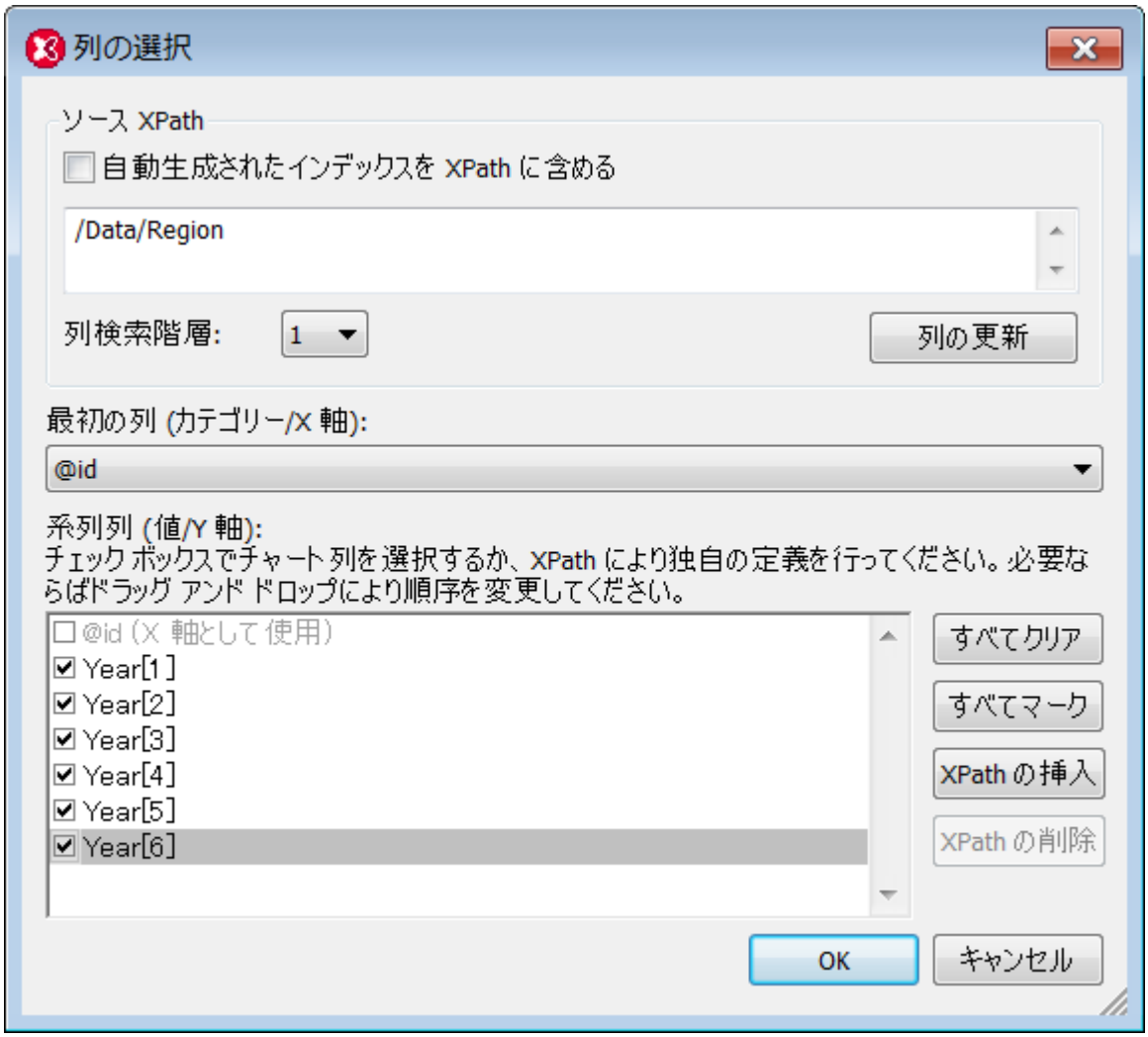
```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
```

```
        <Region id="Americas">
                <Year id="2005">30000</Year>
                <Year id="2006">90000</Year>
                <Year id="2007">120000</Year>
                <Year id="2008">180000</Year>
                <Year id="2009">140000</Year>
                <Year id="2010">100000</Year>
        </Region>
        <Region id="Europe">
                <Year id="2005">50000</Year>
                <Year id="2006">60000</Year>
                <Year id="2007">80000</Year>
                <Year id="2008">100000</Year>
                <Year id="2009">95000</Year>
                <Year id="2010">80000</Year>
        </Region>
        <Region id="Asia">
                <Year id="2005">10000</Year>
                <Year id="2006">25000</Year>
                <Year id="2007">70000</Year>
                <Year id="2008">110000</Year>
                <Year id="2009">125000</Year>
                <Year id="2010">150000</Year>
        </Region>
</Data>
```

5.11.1 チャートの作成

「**新規チャート**」ボタンをクリックすると、(以下のスクリーンショットにある) 列の選択ダイアログが表示され、チャート作成のための3つのデータ選択パラメーターを指定します。これらパラメーターはチャートデータテーブルの作成のために使用されます:

- *ソース XPath* : XPath 条件式は、ダイアログが開かれる際に自動的に入力されます。列の選択ダイアログが開かれたときに選択されていた XML ドキュメント内のノードがここでは入力され、キーボードを使って編集することができます。*インデックス* を XPath に自動追加チェックボックスにより、XPath 内のかっこがフィルタリングに使用されるかが決定されます (詳細はソース XPath 317 を参照ください)。ソース XPath にて選択されたノードの子孫は、X-軸、Y-軸データ列として使用することができるようになります。列検索階層コンボボックスでは、X-軸、Y-軸のデータ選択に使用するためのノードを検索する階層を指定することができます。ソース XPath を編集するか、新たな列検索階層を選択した後は、「**列の更新**」ボタンをクリックして、ダイアログにある X-軸、Y-軸リストの更新を行う必要があります。
- *X-軸* : このコンボボックスでの選択により X-軸として使用されるノードを指定することができます。この選択に取得されるシーケンスにより X-軸に表示されるラベルが決定されます。コンボボックスの *自動列挙* オプションにより X-軸には番号が与えられます。X-軸コンボボックスでは Y-軸のために作成された XPath 条件式も利用できることに注目してください。
- *Y-軸* : このペインでチェックされたエントリが、Y-軸で表示される数値のノードになります。「**全てマーク**」と「**全てクリア**」ボタンにより、Y-軸ペインにある全てのアイテムの選択とその解除を行うことができます。「**XPath の挿入**」ボタンにより、ソース XPath から得られたノードの子孫では無い系列を生成することができます。X-軸で選択されたノードや XPath 条件式は、Y-軸に対して使用することができず、グレーで表示されます。



チャートデータテーブルの作成方法

チャートにて使用されるデータは列のセなくダイアログにおける選択により決定されます。ここでは例を使用しながらチャートのデータを選択する方法を説明します。この例で使用される XML ドキュメントには3つの `Region` 要素が含まれるため、ソース XPath の `/Data/Region` により、それら3つの全てが返されます。各 `Region` 要素をコンテキストノードとして、データの列が生成されます。ソース XPath により選択された各 `Region` 要素に対して:

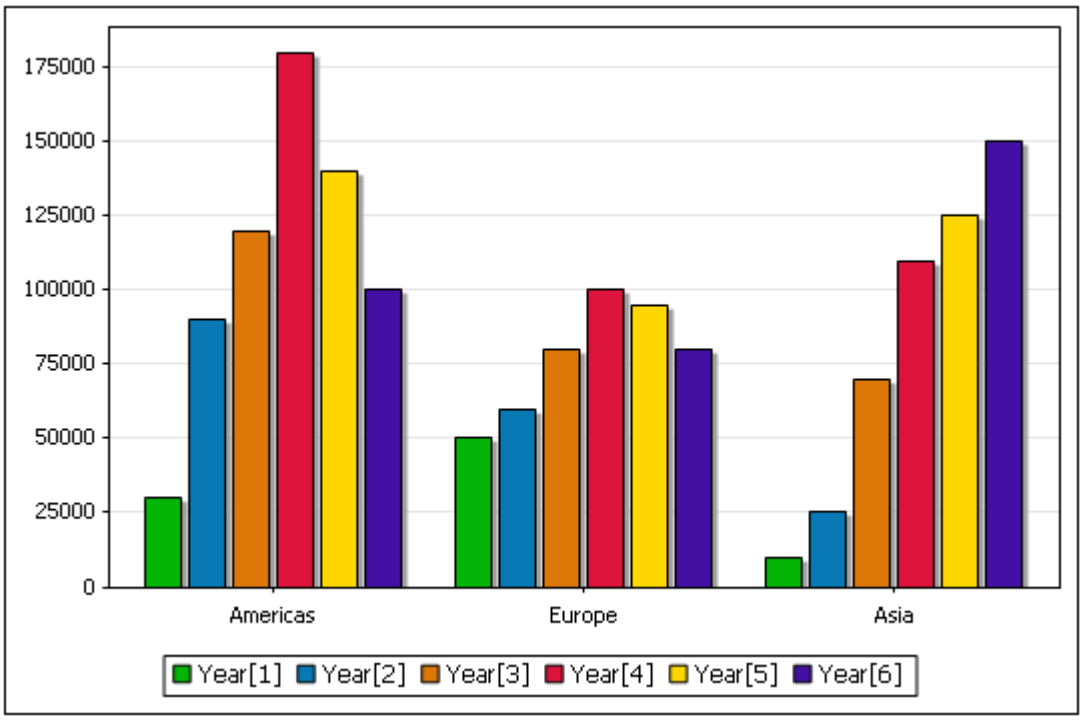
1. X-軸表現により最初の列が生成されます(デフォルトでは、この列が X-軸のラベルとして使用される列になります)。
2. 各系列(Y-軸選択)に対して、列が生成されます。

上のスクリーンショットにある列の選択ダイアログから生成されたチャートデータは、以下のテーブルのようになります。

ソース XPath	X-軸	Y-軸(系列列)					
Region[1]	@id	Year[1]	Year[2]	Year[3]	Year[4]	Year[5]	Year[6]
Region[2]	@id	Year[1]	Year[2]	Year[3]	Year[4]	Year[5]	Year[6]
Region[3]	@id	Year[1]	Year[2]	Year[3]	Year[4]	Year[5]	Year[6]

(C) 2015-2021 Altova GmbH

このデータから生成された棒グラフは以下のようになります:



以下の重要なポイントに注意してください:

- X-軸の要素の数は、ソースXPath 条件式により得られたシーケンスのサイズにより決定されます(この例の場合では3つになります)。
- ソースXPath から得られたノードはコンテキストノードとなり、X-軸の各要素で使用される2種類のデータ生成に使用されます:(i) X-軸の要素ラベル(X-軸選択により作成)と、(ii) 要素に対する系列のプロット(これらの系列はY-軸選択とともに選択されます)。X-軸ならびにY-軸に対して入力されたXPath 条件式は、これら(ソースXPath)ノードのコンテキスト内で評価されます。
- X-軸選択により得られたシーケンスが、それぞれ各要素のラベルになります。実際にある要素よりもラベルの数が少ない場合、いくつかの要素はラベルなしで表示されます。
- 各系列(例えば`Year[1]`)は各コンテキストノードに対して1回評価されます。円グラフのようなチャートによっては、使用される系列を1つにすることができます。
- 凡例は系列アイテムの名前から取得されます。

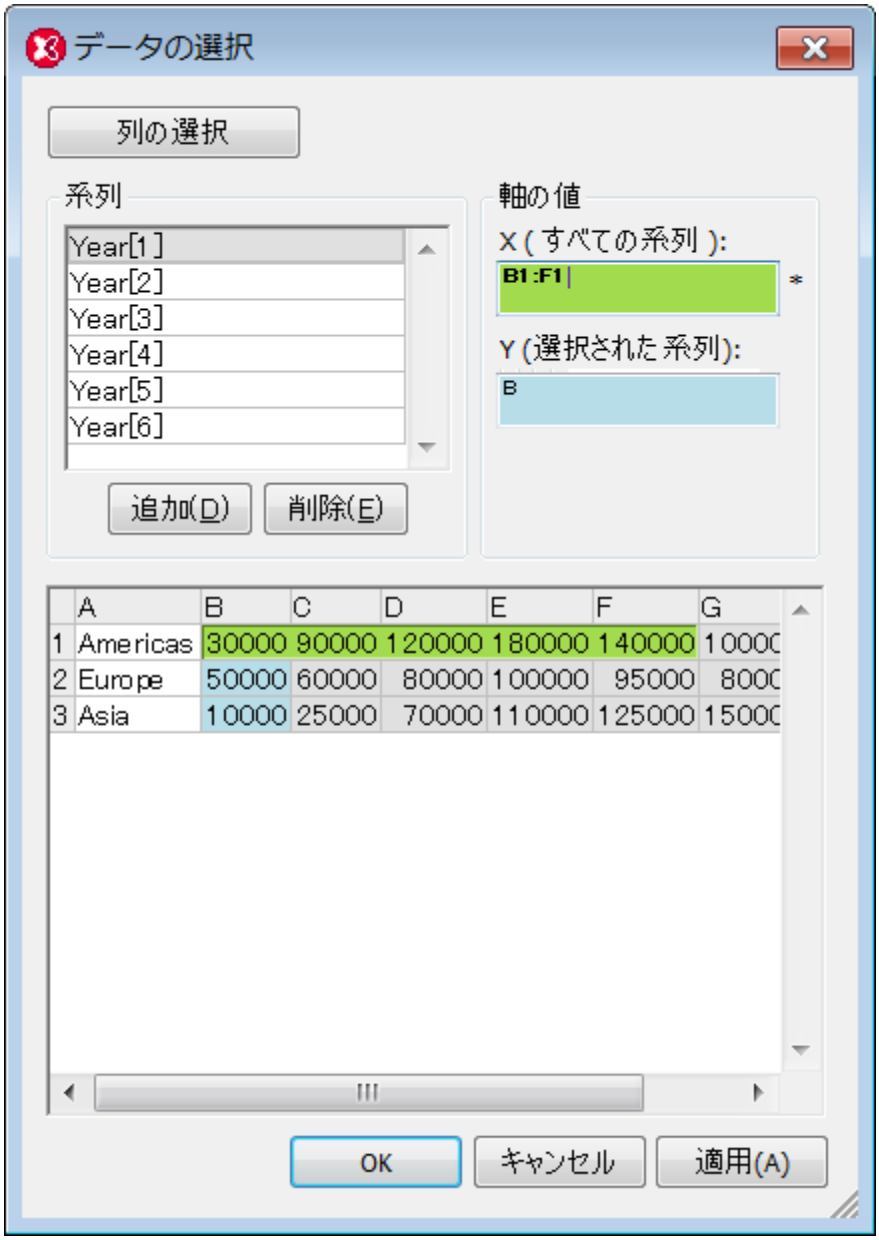
ここで使用したサンプルのXMLドキュメントを示します。このファイルは`YearlySales.xml`というファイル名で`C:\Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples¥Tutorial`に収められています。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
       <Region id="Americas">
              <Year id="2005">30000</Year>
              <Year id="2006">90000</Year>
              <Year id="2007">120000</Year>
              <Year id="2008">180000</Year>
              <Year id="2009">140000</Year>
              <Year id="2010">100000</Year>
       </Region>
       <Region id="Europe">
              <Year id="2005">50000</Year>
```

```
                <Year id="2006">60000</Year>
                <Year id="2007">80000</Year>
                <Year id="2008">100000</Year>
                <Year id="2009">95000</Year>
                <Year id="2010">80000</Year>
        </Region>
        <Region id="Asia">
                <Year id="2005">10000</Year>
                <Year id="2006">25000</Year>
                <Year id="2007">70000</Year>
                <Year id="2008">110000</Year>
                <Year id="2009">125000</Year>
                <Year id="2010">150000</Year>
        </Region>
</Data>
```

データの選択ダイアログに表示される上のデータの選択を(以下のスクリーンショットにある)テーブルの中で見ることができます。データの選択ダイアログへはチャート出力ウィンドウの「**データの選択**」ボタンをクリックすることでアクセスすることができます。

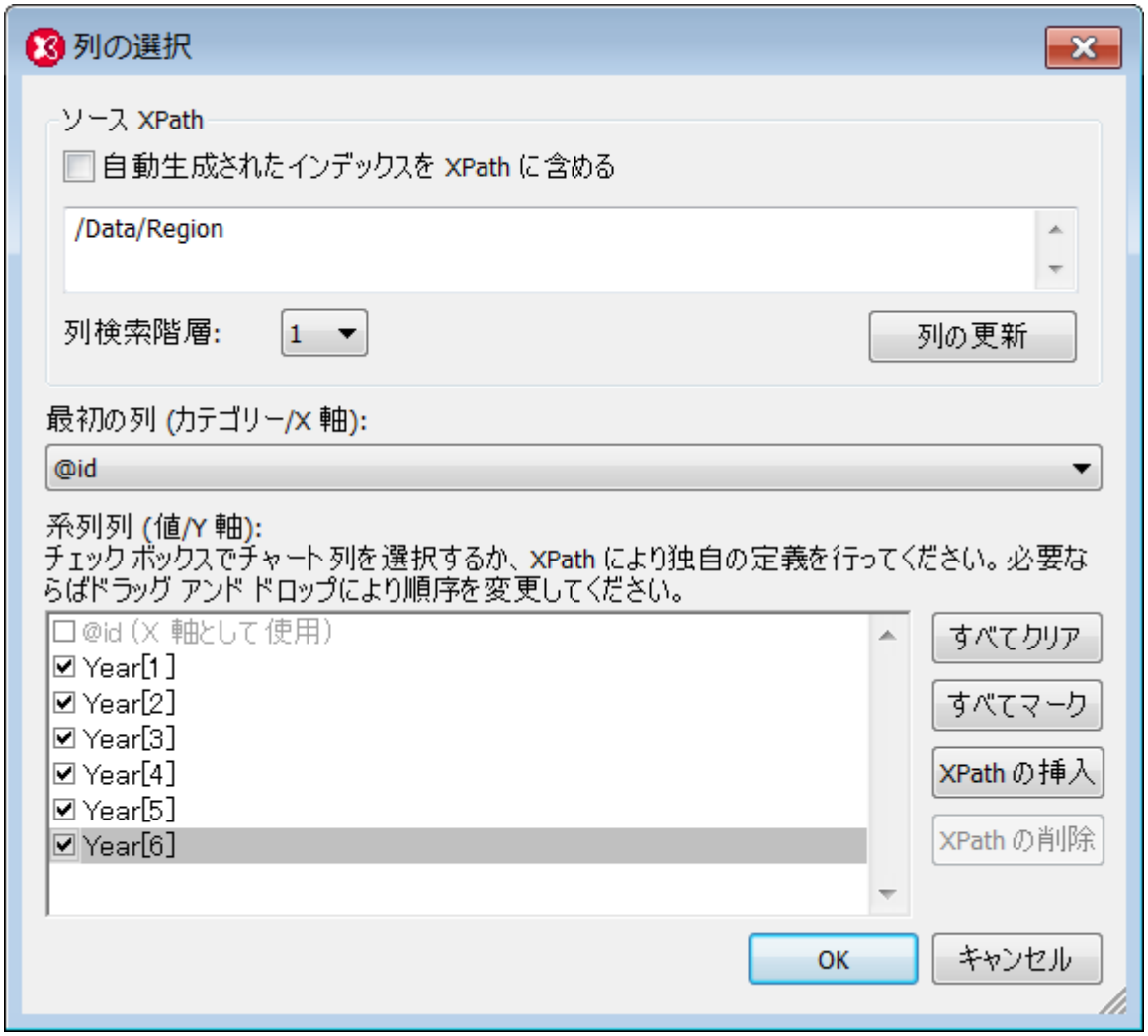
(C) 2015-2021 Altova GmbH



列の選択ダイアログにある個別のパラメーターの詳細については、各セクションを参照ください: ソースXPath 317、X-軸選択 320、Y-軸選択 325、チャートデータ 329。

5.11.2 ソース XPath

ソースXPath は列の選択ダイアログにて指定することができます。ソースXPath により、ドキュメント内のどのノードが、X-軸、Y-軸のデータとして使用できるかが決定されます。列検索階層コンボボックスでは、X-軸、Y-軸のデータとして使用されるノードの検索の深さが指定されます。



ソースXPath または列検索階層を選択した後、「**列の更新**」ボタンにより、(ダイアログ内で)X-軸とY-軸に使用可能な選択肢を更新します。かっこがXPath 条件式に含まれている場合(例: `/data/Region[1]` では `[1]` というかっこ表現により最初の`Region` 要素が選択されます)、自動生成されたインデックスを含めるチェックボックスを有効にする必要があります。ソースXPath 条件式を修正した場合、「**列の更新**」ボタンをクリックするようにしてください。

カーソル位置からのソースXPath

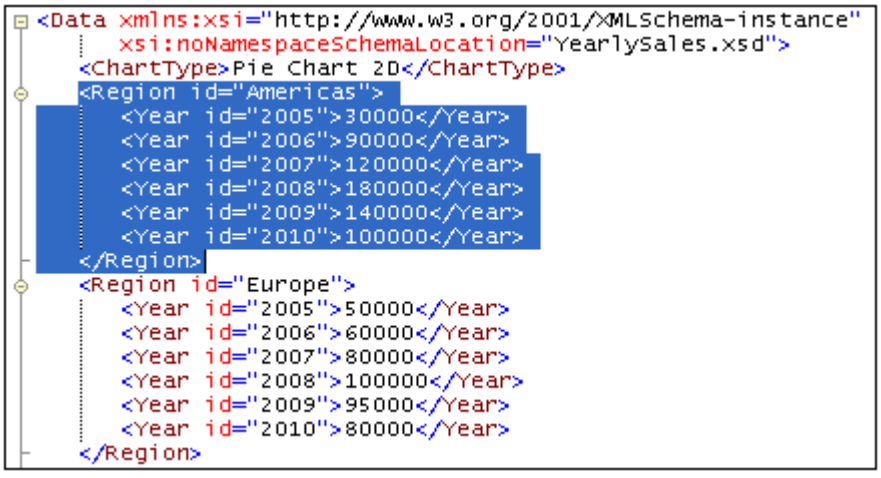
列の選択ダイアログが開かれたときに選択されるソースXPath について説明するために、以下に示されるドキュメントを使用します。このファイルは`YearlySales.xml` というファイル名で C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント\Altova\XMLSpy2021\Examples\Tutorial に収められています。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
        <Region id="Americas">
                <Year id="2005">30000</Year>
                <Year id="2006">90000</Year>
                <Year id="2007">120000</Year>
                <Year id="2008">180000</Year>
                <Year id="2009">140000</Year>
```

```
            <Year id="2010">100000</Year>
        </Region>
        <Region id="Europe">
            <Year id="2005">50000</Year>
            <Year id="2006">60000</Year>
            <Year id="2007">80000</Year>
            <Year id="2008">100000</Year>
            <Year id="2009">95000</Year>
            <Year id="2010">80000</Year>
        </Region>
        <Region id="Asia">
            <Year id="2005">10000</Year>
            <Year id="2006">25000</Year>
            <Year id="2007">70000</Year>
            <Year id="2008">110000</Year>
            <Year id="2009">125000</Year>
            <Year id="2010">150000</Year>
        </Region>
</Data>
```

以下のケースが考えられます:

- カーソルが `Data` 要素の開始タグ(属性やその値を含む)や終了タグ内にある場合、または `Data` 要素内部(但し、`Data` の子要素内部ではない)にカーソルがある場合:ソースXPath は `/Data` となります。
- カーソルが `Region` 要素の開始タグ(属性やその値を含む)や終了タグ内にある場合、または `Region` 要素内部(但し、`Region` の子要素内部ではない)にカーソルがある場合:ソースXPath は `/Data/Region` となります。
- カーソルが `Year` 要素の開始タグ(属性やその値を含む)や終了タグ内にある場合、または `Year` 要素内部にカーソルがある場合:ソースXPath は `/Data/Region[N]/Year` となります。かっこによるフィルター `[N]` により、選択された `Year` 要素を含んでいる `Region` 要素が指定され、X-軸、Y-軸のデータ選択は、この `Region` 要素内にある `Year` 要素に限定されます。
- 個別の `Region` 要素(例: `/Data/Region[1]`)を選択したい場合、以下のスクリーンショットにあるように、最初の `Region` 要素をハイライトします。



- 同様に、2 つの `Region` 要素をハイライトすることで、これら2つの `Region` 要素だけを選択するXPath 条件式が生成されます。

ソースXPath 条件式は、後からでも列の選択ダイアログにて編集することができます。

グリッドビューでは、ノードをクリックし範囲をマークすることで、選択を行うことができます。グリッドビューの選択から生成されるソースXPath は、上のテキストビューのものと同様のものになります。

XPath のインデックスを追加

*インデックスを追加するチェックボックス*により、XPath 条件式内にあるかっこを使ったフィルタリングが行われるか、列の選択ダイアログ呼び出された際に、これらかっこフィルタリングを自動的に入力するか、または手動で入力するかが決定されます。例えば、カーソルが`Region` 要素の最初の子孫要素内にあり、*インデックスを追加するチェックボックス*が有効になっている場合、自動的に入力される XPath 条件式は、例えば `/Data/Region[1]/Year` となります。*インデックスを追加するチェックボックス*が有効になっていない場合、条件式は `/Data/Region/Year` のようになります。

*インデックスを追加するチェックボックス*では、手動で入力されたかっこを保持するかどうかも決定されます。ソース XPath 条件式のかっこを使用したい場合、*インデックスを追加するチェックボックス*を有効にする必要があります。

ソース XPath 選択の意味

ソース XPath 選択に関して以下の点に注意してください。

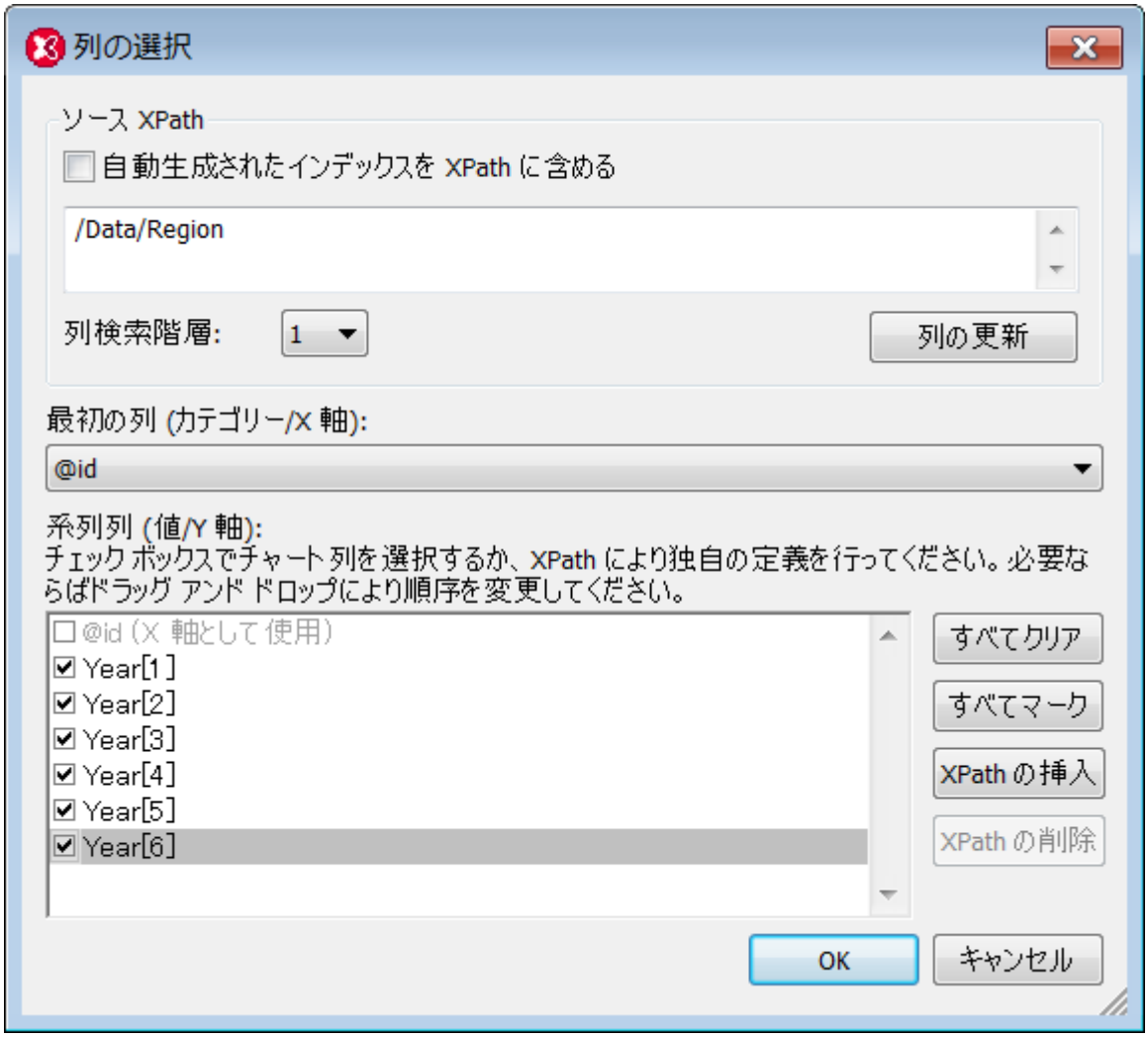
- ソース XPath により得られた*シーケンス*のアイテム数により、X-軸の要素の数が決定されます。ソース XPath を修正する以外に、X-軸の要素数を変更する方法は1つしかありません。要素の数より1大きな系列の番号を選択するというものです。この方法の詳細に関してはX-軸選択 320 を参照ください。
- ソース XPath ノードは、X-軸、Y-軸のデータ選択で利用可能な全てのノード、ならびに XPath 条件式の祖先ノードとなります。
- 上の2つの結果、ソース XPath 条件式に対する変更は、X-軸の要素数だけではなく、チャートに関連した XPath 条件式のコンテキストにも影響を与えてしまうことに注意してください。

以下に、上の例で示された XML ドキュメントに対して適用できる XPath 条件式とその意味を記します:

- **`/Data/Region`** :3つの`Region` 要素を返し、X-軸には3つの要素が与えられます。各 `Region` 要素が XPath 条件式のコンテキストノードになります。
- **`/Data/Region/Year`** :18の`Year` 要素を返し、X-軸の要素数は18になります。各 Year 要素が XPath 条件式のコンテキストノードになります。
- **`/Data/Region[1]/Year`** : 最初の`Region` 要素の子要素である6つの`Year` 要素が返され、X-軸の要素数は6になります。最初の`Region` 要素の子要素である各 `Year` 要素が XPath 条件式のコンテキストノードとなります。
- **`distinct-values(//Year/@id)`** :6つのアイテムが返されます(`Year/@id` 属性の`distinct` 値:2005、2006、2007、2008、2009、2010)。しかしこの XPath 条件式は*ノードアイテム*を返さないため、XPath 条件式のコンテキストノードとなることはできません。(例えば`current()` 関数を使用するなどして)この*シーケンス*内のアイテムが XML ドキュメント内のターゲットノードに使用されるのであれば、この XPath 条件式をドキュメントのルートからスタートするように書きなおして、コンテキストノードを確立する必要があります(例: `/Data/Region[1]/Year[@id eq .]`)。

5.11.3 X-軸の選択

X-軸の選択は、列の選択ダイアログにて指定されます(以下のスクリーンショットを参照)。この選択により X-軸に表示されるラベルが決定されます。ラベルはデータの選択ダイアログにて編集することができます(以下を参照)。



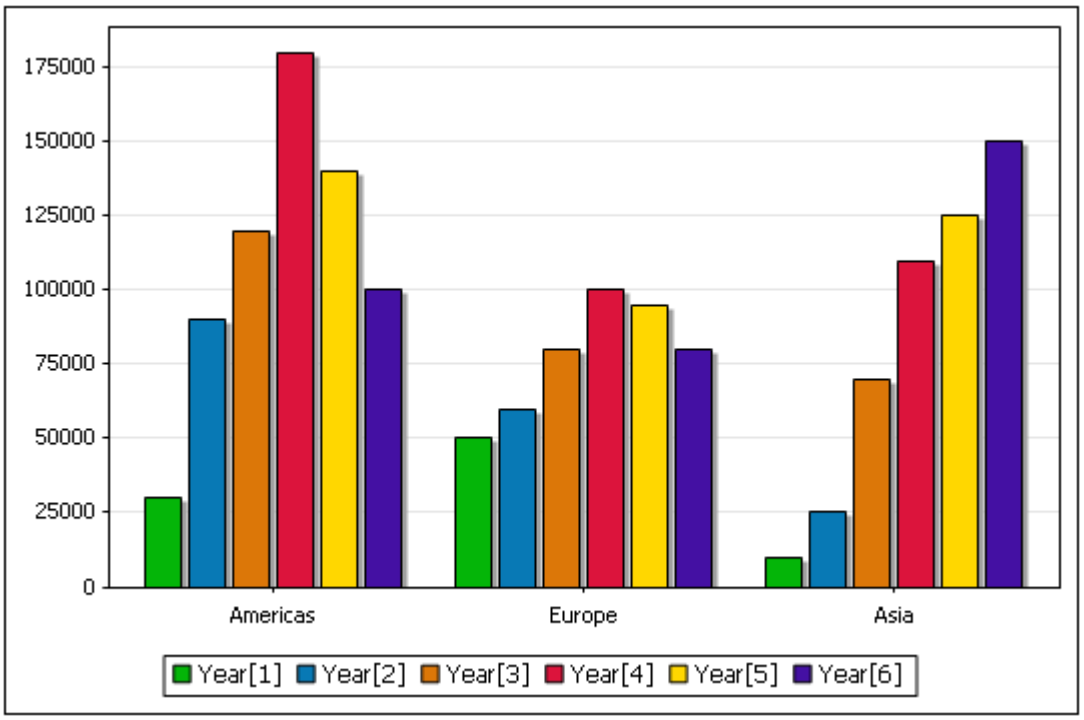
以下のXMLドキュメントを例にして考えてみましょう(このファイルは`YearlySales.xml`というファイル名で`C:\Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples¥Tutorial`に収められています)。カーソルが最初の`Region`要素の開始タグ内に位置している状態で、チャート出力ウィンドウの「**新規チャート**」ボタンをクリックします。列の選択ダイアログがソースXPathと共に表示されます: `/Data/Region`。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
       <Region id="Americas">
              <Year id="2005">30000</Year>
              <Year id="2006">90000</Year>
              <Year id="2007">120000</Year>
              <Year id="2008">180000</Year>
              <Year id="2009">140000</Year>
              <Year id="2010">100000</Year>
       </Region>
       <Region id="Europe">
              <Year id="2005">50000</Year>
              <Year id="2006">60000</Year>
              <Year id="2007">80000</Year>
```

```
            <Year id="2008">100000</Year>
            <Year id="2009">95000</Year>
            <Year id="2010">80000</Year>
        </Region>
        <Region id="Asia">
            <Year id="2005">10000</Year>
            <Year id="2006">25000</Year>
            <Year id="2007">70000</Year>
            <Year id="2008">110000</Year>
            <Year id="2009">125000</Year>
            <Year id="2010">150000</Year>
        </Region>
</Data>
```

ソースXPath 317 のセクションで説明したように、このソースXPath により、X-軸に3つの要素があるチャートが作成されます(ソースXPath は`Region` 要素の数である3つのアイテムを返すため)。これら3つの要素のラベルを3つの地域の名前で表したいので、X-軸選択のコンボボックスから`@id` 属性を選択します(上の列の選択ダイアログのスクリーンショットを参照)。

各要素に対してチャートデータを生成するには、各`Region` 要素の評価を行わなければなりません。各`Region` 要素に対して、`id` 属性によりX-軸属性のラベルが適切に生成されます。X-軸は以下のスクリーンショットのように表示されます。

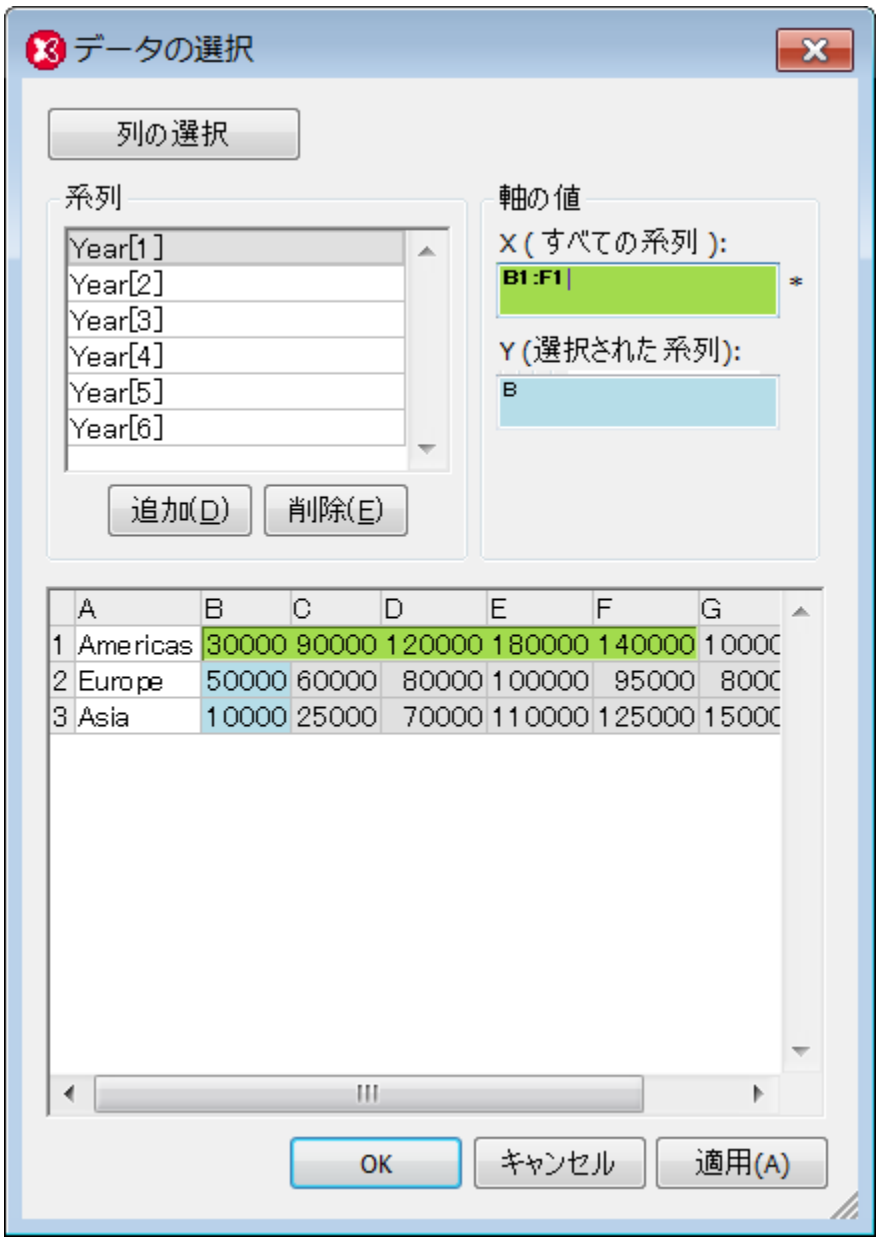


他のXPath 条件式がX-軸コンボボックスで選択された場合、その条件式が`Region` 要素コンテキスト内で評価され、評価の結果が対応する要素のラベルとなります。*自動列挙* オプションでは要素の番号に対応した番号のシーケンスを生成します: 最初の要素が1となり、2番目の要素が2となります。

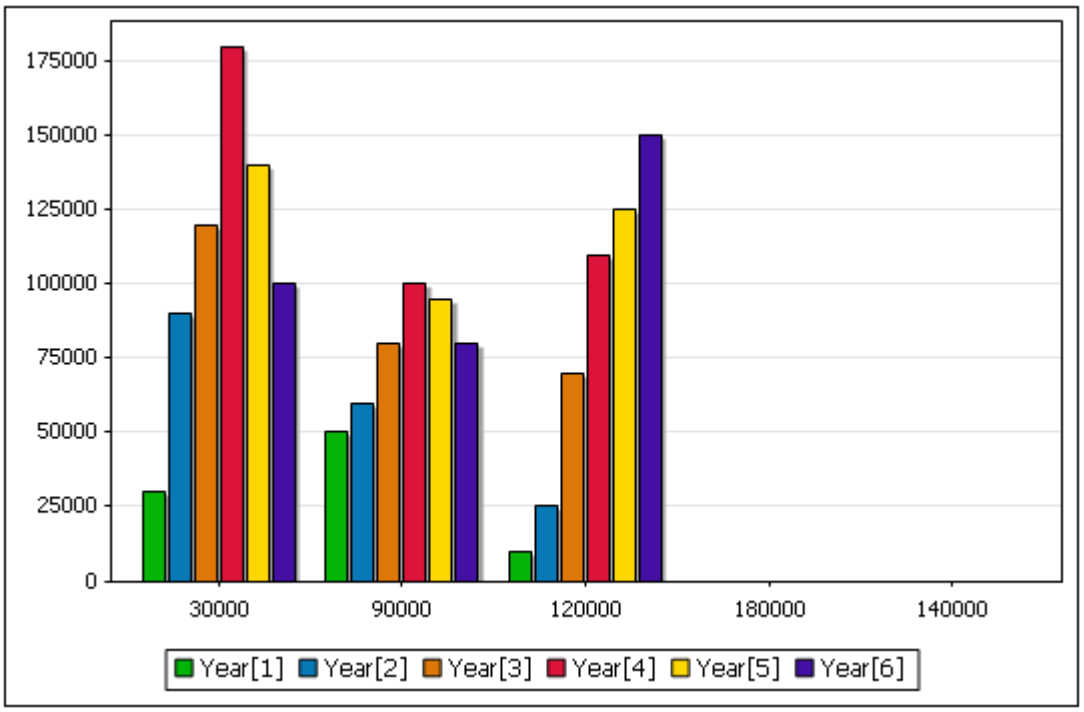
X-軸のラベルとX-軸要素数の修正

X-軸に対するラベルの選択はデータの選択ダイアログから行うことができます(チャート出力ウィンドウの「データの選択」ボタンをクリックすることでアクセスできます)。

以下のスクリーンショットで示されるデータの選択ウィンドウにおいて、軸の値ペイン内のX-軸テキストボックスをクリックします。これでX-軸の選択を修正することができます。そこで B1 フィールドをクリックして、F1 までマウスをドラッグし、選択を B1:F1 とします。「OK」をクリックして新たなチャートを確認します。



この選択により、以下のスクリーンショットにあるような要素に対するラベルが作成されます。新しい選択では5つのアイテムが含まれるため、5つの要素が生成されたことにも注目してください。実際に表示されているのは最初の3つだけですが、これはソースXPath が返したノード数は3つで、これら3つのノードがチャートに表示されるためです。これら3つのノードは、データの選択ダイアログにて表示されている[テーブルの行](314)に対応します。テーブル内の行はソースXPath を変更することでしか修正できないということご注意してください。

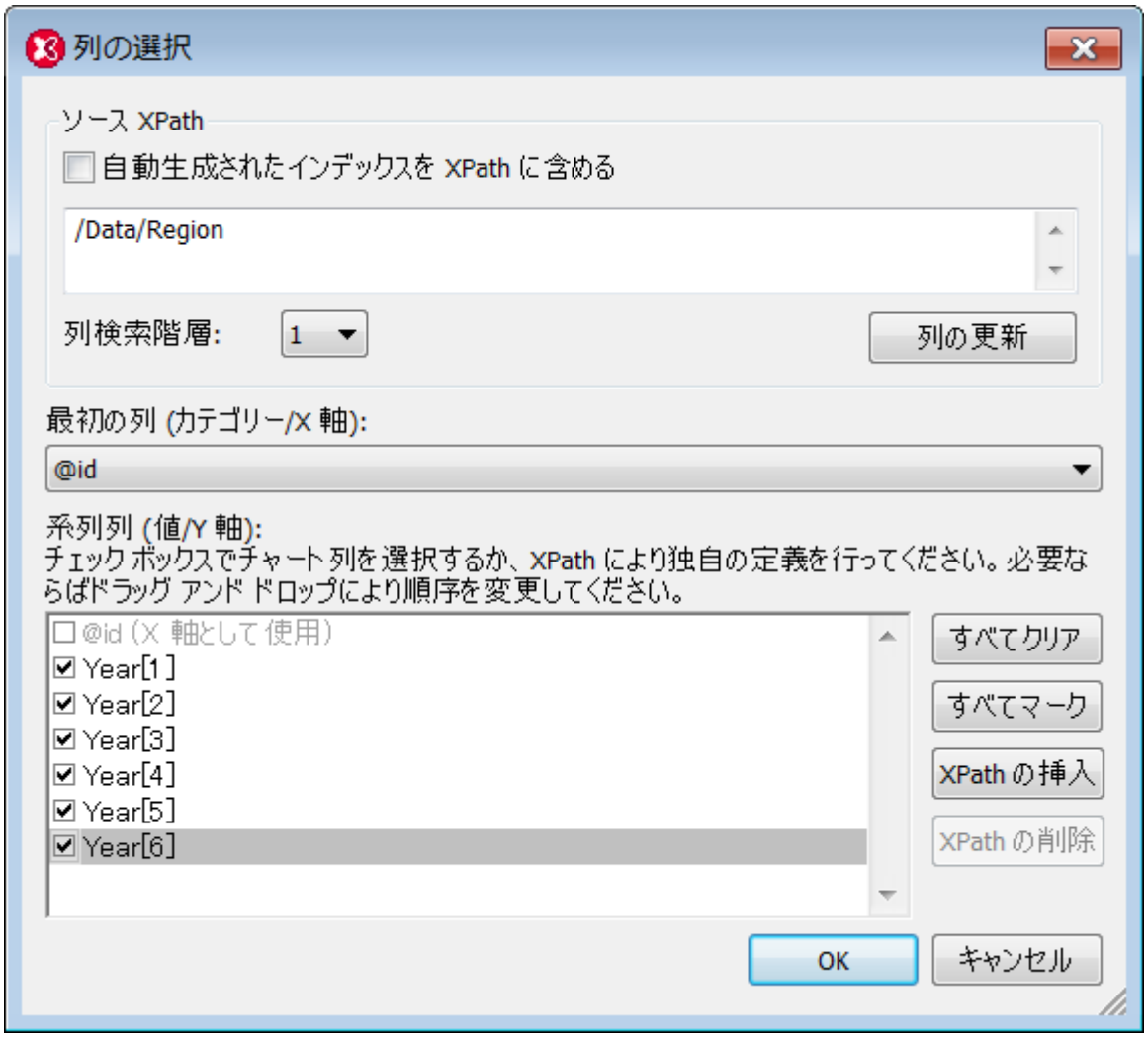


ソースXPath ならびにX-軸の選択に関するより詳細については、チャートデータの作成方法[314]を参照ください。

(C) 2015-2021 Altova GmbH

5.11.4 Y-軸の選択

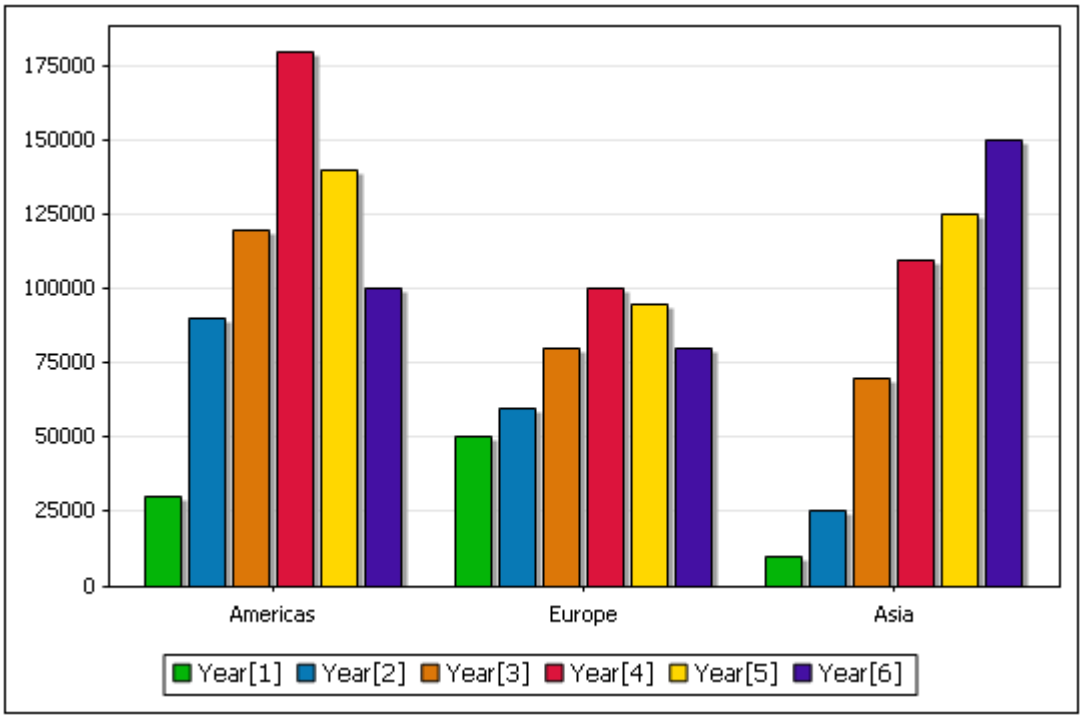
以下のスクリーンショットにあるY-軸は系列軸とも呼ばれます。



この軸で行われる選択により、各 X-軸要素に対してどれだけの系列がプロットされるのか決定されます。1つの系列だけが選択された場合、各 X-軸が要素となり、その系列のXPath 条件式により得られた値がプロットされます。2つ以上の系列が選択された場合、以下のスクリーンショットにあるように複数の系列が表示されます。チャートデータは以下のテーブルに示されるようになります。Y-軸のデータ選択に対するコンテキストノード[314]は各 Region 要素となります。

ソースXPath	X-軸	Y-軸(系列列)					
Region[1]	@id	Year[1]	Year[2]	Year[3]	Year[4]	Year[5]	Year[6]
Region[2]	@id	Year[1]	Year[2]	Year[3]	Year[4]	Year[5]	Year[6]
Region[3]	@id	Year[1]	Year[2]	Year[3]	Year[4]	Year[5]	Year[6]

結果として表示されるチャートは以下のようなものになります:



3つのX-軸要素があり、各 `Region/@id` 属性の値によりラベルが表示されます。X-軸の各要素にて、各系列のXPath 条件式が評価されます。この例の場合、X-軸の各要素に対して、6つある各系列が評価、プロットされています。例えば、最初の系列(`Year[1]`)は、3つある地域の全てに対してプロットされており、同じことが`Year[6]` まで当てはまります。

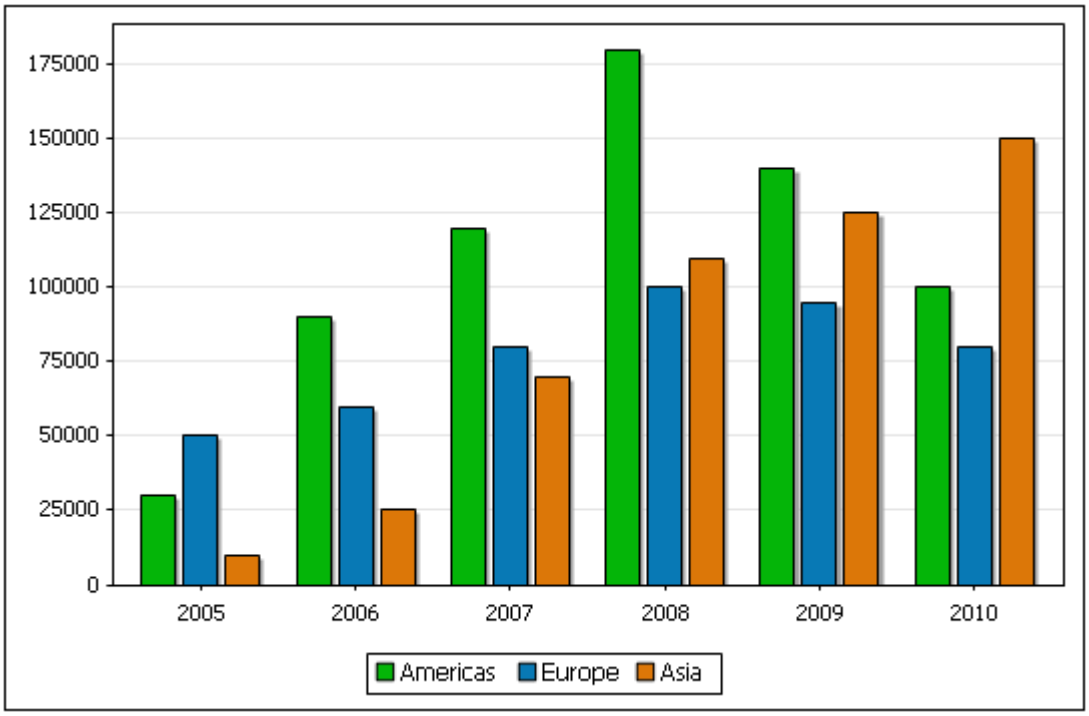
メモ: 円グラフや、各要素に対して1つの系列しかない様なグラフでは、1つの軸しか使用されません。例えば、各要素に対して1つの系列しか存在しないグラフでは、X-軸の各要素には1つの棒しか表示されず、各棒が1つある系列を表しています。円グラフの場合、単一の系列の値の合計が100% になるように、X-軸の各要素が割り当てられます。

Y-軸の凡例

チャート下部に表示される凡例は、各系列の名前になります。これら名前は[データの選択ダイアログ](#) 329 にて修正することができます。

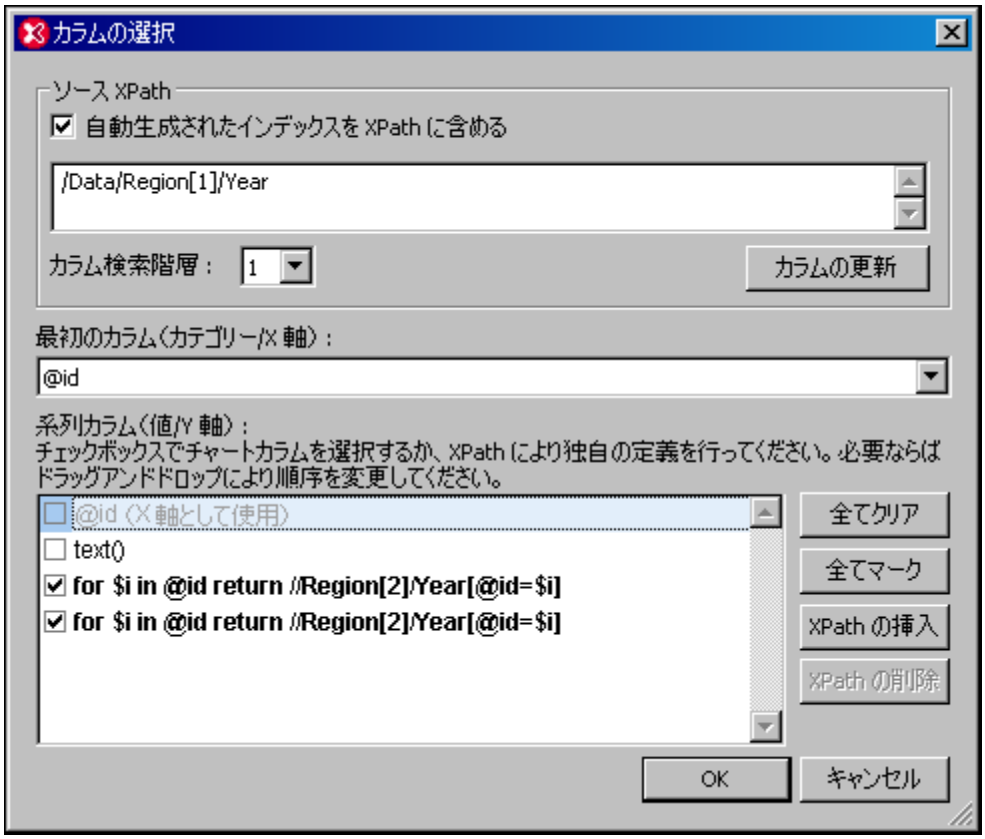
X-軸とY-軸の選択を切り替える

上の例では地域がX-軸となり、各地域の年間売上がY-軸にプロットされていました(つまり`Year` 要素が系列になっていました)。しかし年をX-軸にプロットし、棒グラフを使って毎年の年間売上を地域ごとに比較したい場合はどうすればよいのでしょうか? (ソースXPath の選択から得られる)6つのX-軸系列を入手し、X-軸の系列を対応する年によりラベル付けし、各 X-軸系列で描画される3つの(地域)系列を選択する必要があります。以下のスクリーンショットにある列の選択ダイアログでは、このようなデータの選択を行う方法が示されています。



系列の選択

ノードを系列として選択するためには、ノードのチェックボックスにチェックを入れます。ソースXPath と列検索階層を修正して、目的のノードが系列ペインに表示されるようにします。他にも、(以下のスクリーンショットにあるように)XPath 条件式を追加してノードを選択するという方法もあります。このような使用に関する詳細については[チャートの例: 詳細](#) 360 を参照ください。



メモ

このセクションで使用されたXMLドキュメントのサンプルは以下のようなものになります。このファイルは`YearlySales.xml`というファイル名でC:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples¥Tutorial に収められています。

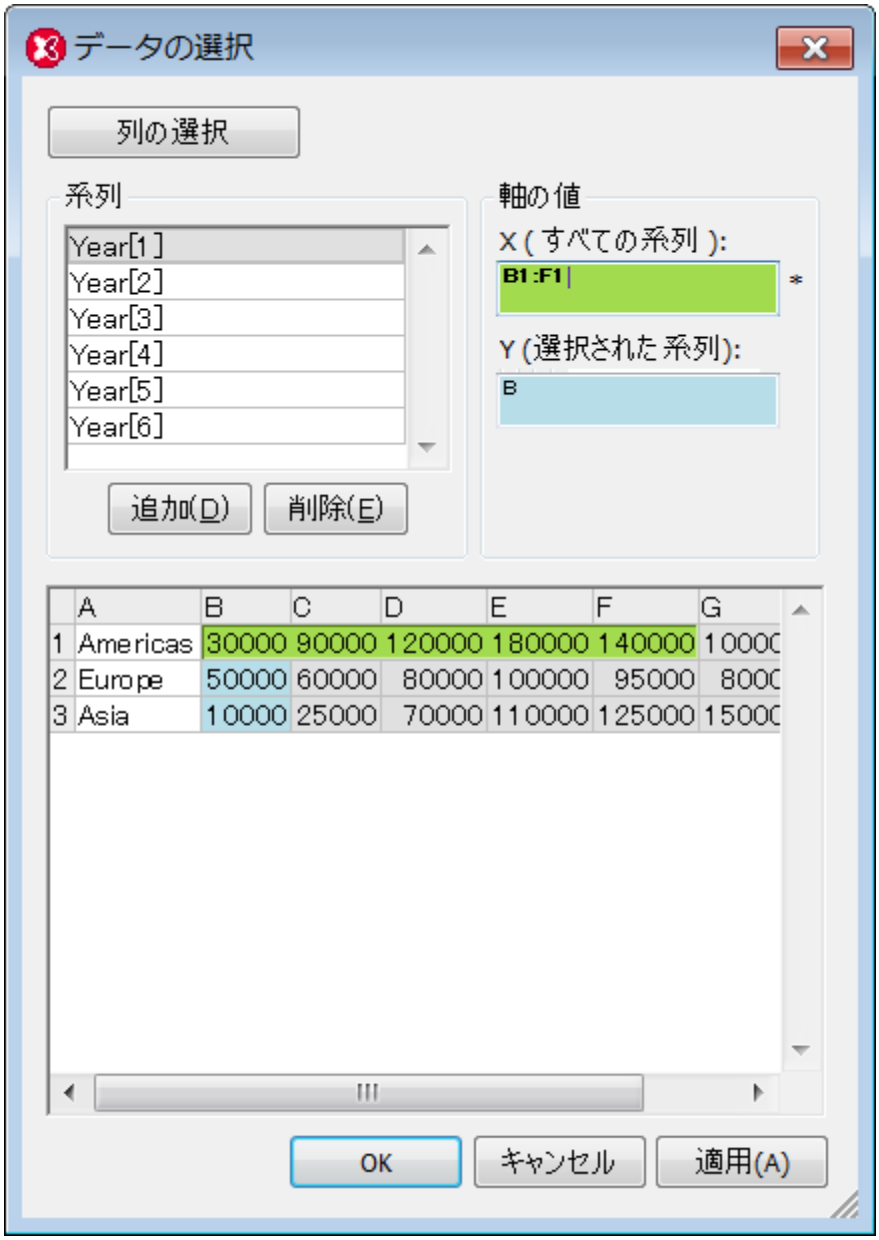
```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
       <Region id="Americas">
              <Year id="2005">30000</Year>
              <Year id="2006">90000</Year>
              <Year id="2007">120000</Year>
              <Year id="2008">180000</Year>
              <Year id="2009">140000</Year>
              <Year id="2010">100000</Year>
       </Region>
       <Region id="Europe">
              <Year id="2005">50000</Year>
              <Year id="2006">60000</Year>
              <Year id="2007">80000</Year>
              <Year id="2008">100000</Year>
              <Year id="2009">95000</Year>
              <Year id="2010">80000</Year>
       </Region>
       <Region id="Asia">
```

```
            <Year id="2005">10000</Year>
            <Year id="2006">25000</Year>
            <Year id="2007">70000</Year>
            <Year id="2008">110000</Year>
            <Year id="2009">125000</Year>
            <Year id="2010">150000</Year>
        </Region>
</Data>
```

5.11.5 チャートのデータ

「**データの選択**」ボタンをクリックすることで、(以下のスクリーンショットにある) データの選択ダイアログが表示されます。ダイアログは、(i) 系列ペイン、(ii) 軸の値ペイン、そして (iii) チャートデータテーブルの3つのペインから構成されます。

「**列の選択**」ボタンをクリックすることで、列の選択ダイアログ[313]が表示され、ソースXPath やX-軸、Y-軸のデータ選択を変更することができます。



系列ペイン

系列ペイン内で最初から選択されている系列は、[列の選択ダイアログ](#)[313]にて選択したものになります。「OK」ボタンをクリックしたときに、このペインで選択されているものがチャート内に表示される系列になります。系列ペインでは3つの操作を行うことができます:

- **系列の追加と削除** : チャート内に表示される系列の数を変更することができます。
- **系列の名前変更** : 系列の名前はチャート内の凡例として表示されます。
- **各系列のデータを選択** : 系列ペインにて系列が選択された状態だと、X-軸とY-軸のデータを、軸の値ペインで指定することができます。その方法については以下を参照ください。

(C) 2015-2021 Altova GmbH

軸の値ペイン

X-軸ならびにY-軸のデータは、軸の値ペインの対応するテキストボックスから入力することができます。どちらかのテキストボックスをクリックして、その中の値を編集することができ、編集状態であることがテキストボックスの右側に、アスタリスクとして表示されます。データの選択は、チャートデータテーブルからデータの範囲を選択することでも行うことができます。選択するデータの範囲は、列や行全体、またはそれらの一部として行うことができます。データ範囲の指定は、キーボードによる入力(例えば、`A` や `3`、`B1:F1` など)もしくはチャートデータテーブルの範囲をマークすることで行うことができます。データの範囲を選択するには、範囲の最初のセルを選択し、カーソルを最後のセルまでドラッグします。列や行全体をマークするには、列または行のヘッダーをそれぞれマークしてください。

X-軸の選択により、X-軸ノードのラベルが決定され、全ての系列に対して適用されます。X-軸における要素の数は変更されません。

Y-軸の選択により、選択された系列に対して使用されるセルの範囲が決定されます。選択されたセルの数がX-軸の要素より少ない場合、対応する要素を持たないX-軸の系列は表示されません。選択されたセルの数がX-軸の要素より多い場合、余った分だけ新たに要素が(対応する値とともに)作成されます。

チャートデータテーブル

(データの選択ダイアログ下部に表示される)チャートデータテーブルの構造は、列の選択ダイアログ[313]の選択から取得されます。

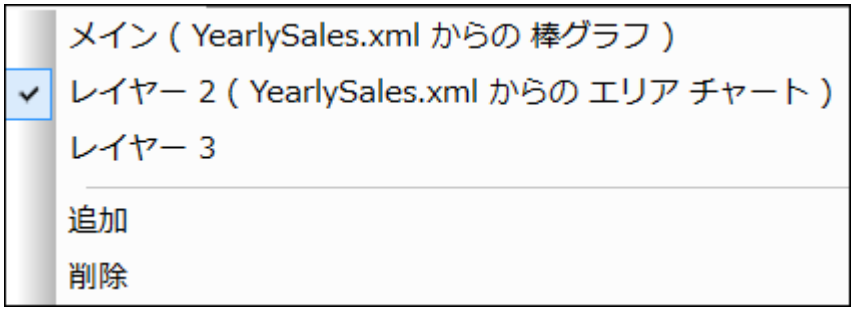
- テーブル内にある行の数は、ソースXPath から得られたシーケンス内にあるアイテム数に対応します。
- 列はアルファベットのA から順に名前がふられています。この列名を使って、軸の値ペインから選択を行うことができます(例: `B1:F1`)。
- 最初の列は、ソースXPath 条件式から得られたノードをコンテキストに、列の選択ダイアログ[313]にて選択されたX-軸を評価することで取得しています。
- 最初の列を除く、その他全ての列は、列の選択ダイアログ[313]にて選択されたY-軸を評価することで得られています。列の選択ダイアログ[313]のY-軸の選択から得られた各系列は、チャートデータテーブルの列に対応します。

チャートデータテーブルは、列の選択ダイアログ[313]にて選択されたデータのスーパーセット(上位集合)として見ることもできます。このスーパーセットから、(軸の値ペインにより)目的のデータの範囲、または個別の系列を選択することができます。

5.11.6 オーバーレイ

オーバーレイとは、ベースチャートの上に置かれるチャートのことです。オーバーレイを以下の方法で追加することができます:

1. **オーバーレイボタン**をクリックして、オーバーレイメニューを開きます(*下のスクリーンショット参照*)。



2. **追加** をクリックします。新規のレイヤーが追加されます。
3. 新規のレイヤーが選択されると、**新規チャート** ボタンをクリックして、チャート作成[313]内の選択で指定されているとおりデータを選択します。
4. チャートの種類と外観を変更するには、**種類の変更** と **外観の変更** をそれぞれクリックします。

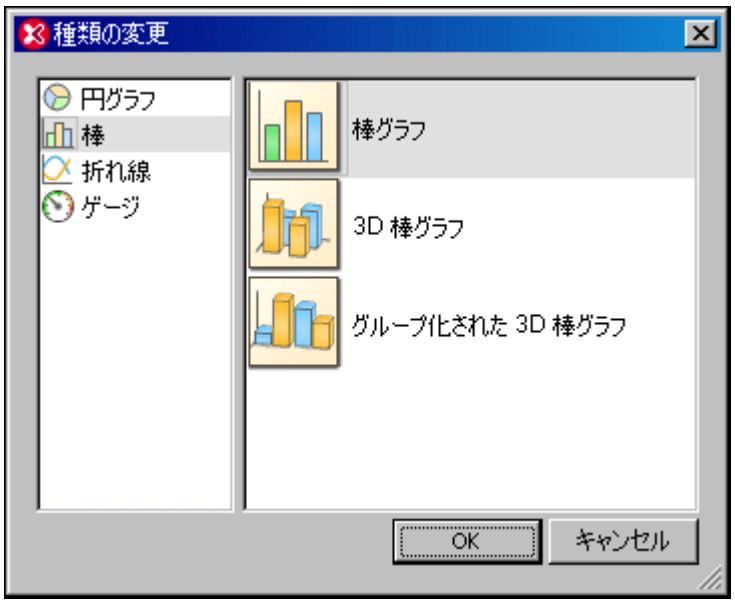
追加できるオーバーレイの数量に制限はありません。各新規のレイヤーはダイアグラム内で既存のレイヤーに追加され重ねて表示されます。レイヤーの順序を変更する場合は、順序の変更を使用して変更します。現在選択されているレイヤーを、**現在選択されている項目**をクリックして削除することができます。

メモ　各オーバーレイは、下のレイヤーの上に追加されます。エリアチャートのみが透明に設定することができ、他のレイヤーの設定は最適では内可能性があります。たとえば、棒グラフーが折れ線グラフの上に置かれている場合、折れ線グラフの外観を損なう可能性があります。上記の点を考慮して、レイヤーの順序を計画してください。

5.11.7 チャートの設定：クイックリファレンス

チャートの種類

（以下のスクリーンショットにある）種類の変更ダイアログにて、使用したい種類のグラフを選択して、「OK」により確定します。種類の変更ダイアログへは、「**種類の変更**」ボタンからアクセスすることができます。

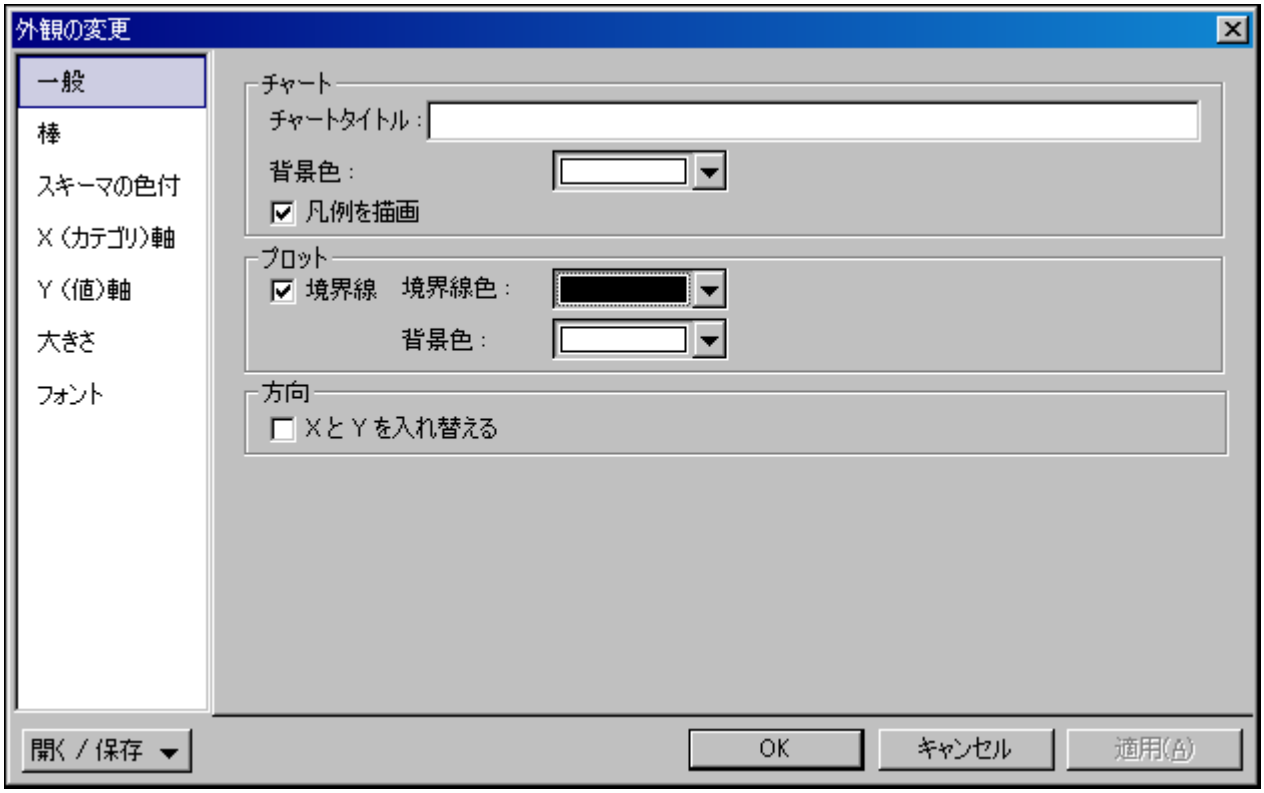


チャートの種類が変更された後は、チャートの設定（タイトルや高さ、幅など）を、以下のスクリーンショットで示される外観の変更ダイアログから行う必要があり、チャートデータも指定する必要があります（以下のスクリーンショットを参照）。各グラフの種類に対してデータの選択を行う方法については、チャートの作成[313]ならびにチャートデータ[329]のセクションを参照ください。

チャートの外観

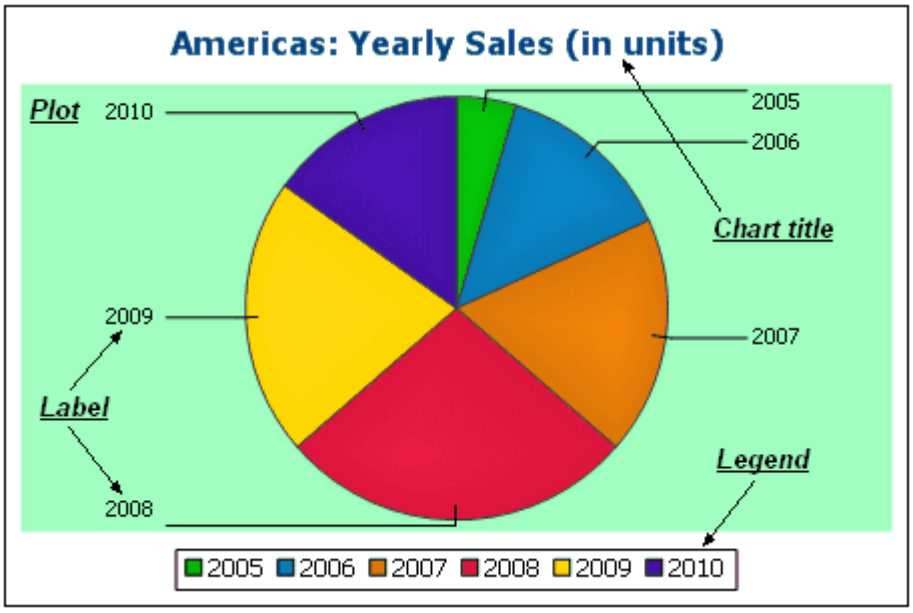
以下のスクリーンショットは棒グラフの設定ダイアログのものです。チャートのタイトルやスキーマの色付、フォントサイズといったチャートの設定は、外観の変更ダイアログにて行ないます。このダイアログへは「**外観の変更**」ボタンからアクセスすることができ、そこで設定することのできる内容もチャートの種類によって変わってきます。

(C) 2015-2021 Altova GmbH



以下の設定は全ての種類のチャートに対して適用されます:

- *一般* : チャートのタイトルや、タブ、チャートの背景色、プロットの境界線やその背景色などを編集することができます(以下のスクリーンショットを参照)。以下のスクリーンショットでは、淡い緑色の背景色が与えられています。チャートの下部に表示され、チャートに表示されている色の説明を行っている凡例を表示するかどうかも、この設定画面から決定することができます。



- *塗りつぶし*：ユーザー定義の塗りつぶし定義に加えて、4つの定義済み塗りつぶし定義が利用可能です。色の追加や削除を行うことで、お好きな塗りつぶしの定義を修正することができます。塗りつぶしタブにて選択された色の定義がチャートで使用されます。
- *大きさ*：チャートが持つ様々な特徴の大きさを、ピクセルまたはパーセンテージによる比率で指定することができます。
- *フォント*：このタブではチャートの凡例、タイトルそしてラベルのフォントプロパティを指定することができます。大きさはピクセル、またはチャートに対するパーセンテージにより指定します。

更に、各種類のチャートは以下のような独自の設定項目を持っています:

- *円グラフ*：(i) 最初の項目が開始される角度を指定します。(ii) 項目が並べられる向きを指定します。(iii) 輪郭の色を指定します。(iv) 色をハイライトするか指定します(3D 円グラフでは、影と透明度を指定することができます)。(v) ラベルを表示するかどうか指定します。(vi) 値やパーセンテージをラベルに追加するか、そしてパーセンテージに表示される桁数を指定します。
- *棒グラフ*：(全般にて) X-軸とY-軸を入れ替えることで、水平の棒グラフを作成することができます(2次元棒グラフのみ)。(棒にて) 輪郭の描画と影付けを行ないます(輪郭の描画は2次元棒グラフのみ)。(X-軸にて) X-軸のラベルと色、そして垂直の補助線をセットします。(Y-軸にて) Y-軸のラベルと色、水平の補助線、表示する値の範囲、そしてY-軸の目盛り間隔を指定します。(Z-軸にて) Z-軸のラベルと色を指定します(3次元棒グラフのみ)。(3D にて) 垂直の傾き、水平の回転、そして視界の広さを指定します。
- *折れ線グラフ*：(全般にて) X-軸とY-軸を入れ替えます。(ラインにて) プロットの点をどのように表示するか指定します。(X-軸にて) X-軸のラベルと色、そして垂直の補助線をセットします。(Y-軸にて) Y-軸のラベルと色、水泳の補助線、表示する値の範囲、そしてY-軸の目盛り間隔を指定します。
- *ゲージ*：(i) ゲージが開始される角度(アングル)と、ゲージの値が表示される角度を指定します(円ゲージのみ)。(ii) 表示される値の範囲を指定します(円ゲージならびに棒ゲージ)。(iii) 大きな目盛りの間隔と小さな目盛りの間隔を指定します(円ゲージならびに棒ゲージ)。(iv) ダイヤルや針、境界線などの色を指定します。

5.11.8 チャートの設定と外観

チャートの設定は次の通り構成されています:

- 基本チャート設定[334]: 最も基本的なチャートの種類の設定です。チャートの種類を選択するには、チャートウィンドウのツールバー内の[種類の変更]をクリックします。種類の変更ダイアログ[334]が表示されます。
- チャートの詳細設定[339]によりチャートの外観(タイトル、凡例、色、フォントなど)を変更することができます。詳細設定は外観の変更ダイアログ[339]で定義することができます。このダイアログにアクセスするには、チャートウィンドウのツールバー内の[外観の変更]をクリックします。

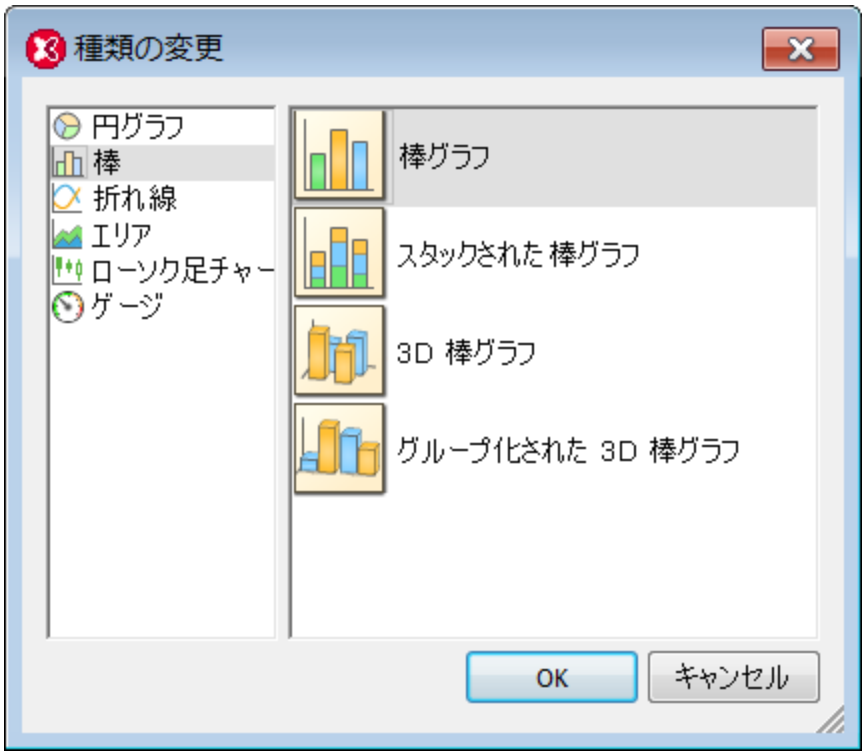
5.11.8.1 チャートの基本設定

このセクション:

- チャートの種類の設定[334]
- チャートの種類のリスト[335]
- 他の基本設定[339]

チャートの種類の設定

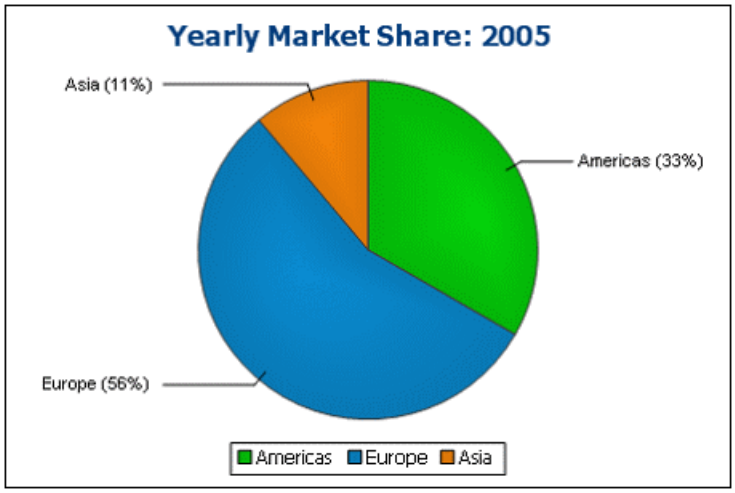
最も基本的な設定はチャートの種類です。チャートの種類を選択するには、チャートウィンドウのツールバーの「種類の変更」をクリックします。



チャートの種類は、チャートウィンドウにある「種類の変更」ボタンをクリックすることで表示される、種類の変更ダイアログ[334] ボックスにて定義されます。(上のスクリーンショット)、チャートの種類を選択して、**[OK]** をクリックします。

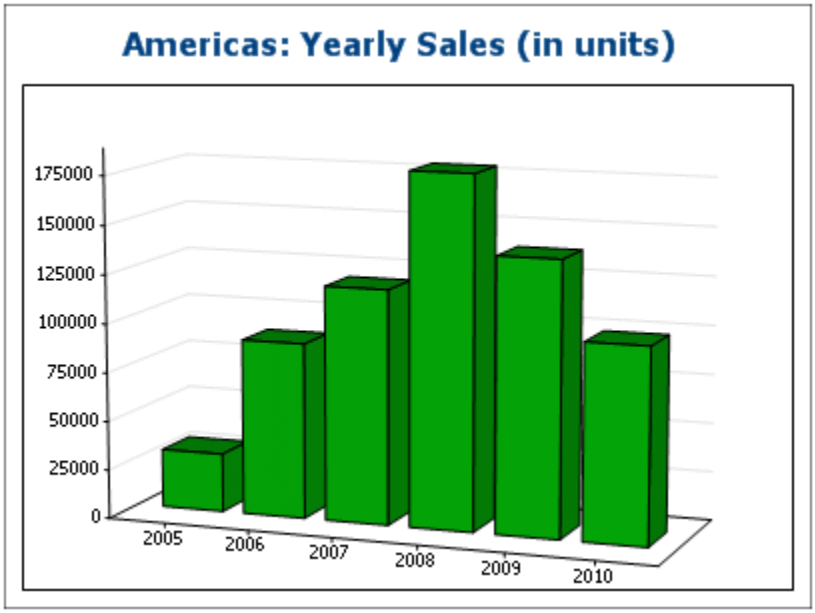
- *円チャート*

  円グラフでは、単一軸の値を表示し、別の軸により値のラベル付けを行います。

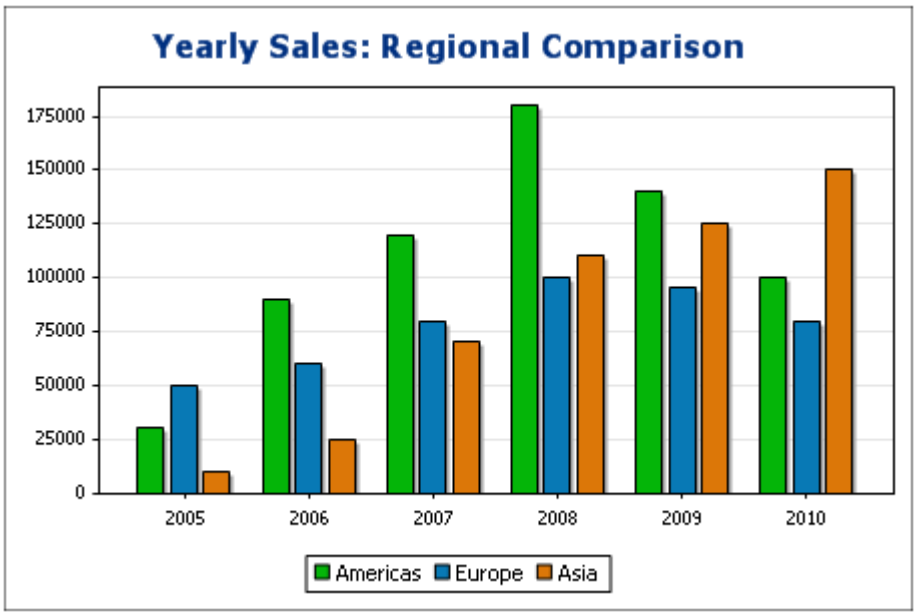


- *棒グラフ*

  棒グラフでは、X-軸に数値以外の値を使用することができます。

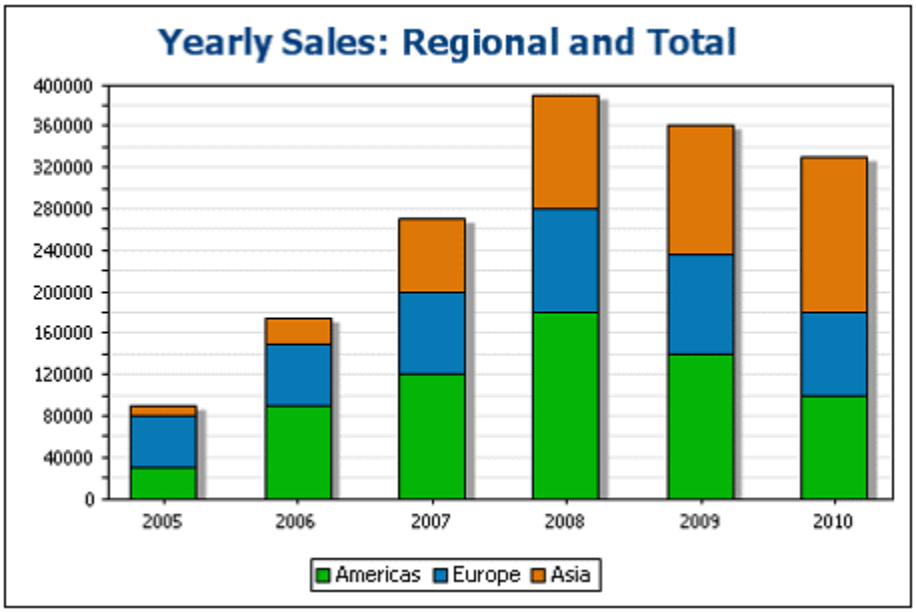


上のサンプルで表示されるように次のセットの3つの値を使用することもできます: (i) 地域、(ii) 年度、(iii) 売り上げ。 棒グラフは2D ならびに3D にて表示することができます。



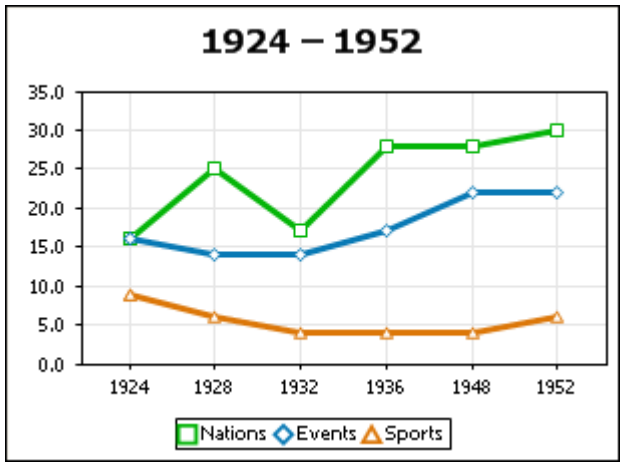
更に、スタックされた棒グラフを使用することで、チャートに含まれている系列の合計を表示することができます。

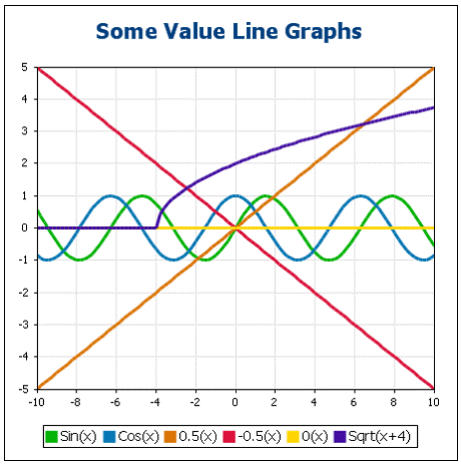
(C) 2015-2021 Altova GmbH



▼ *折れ線グラフ*

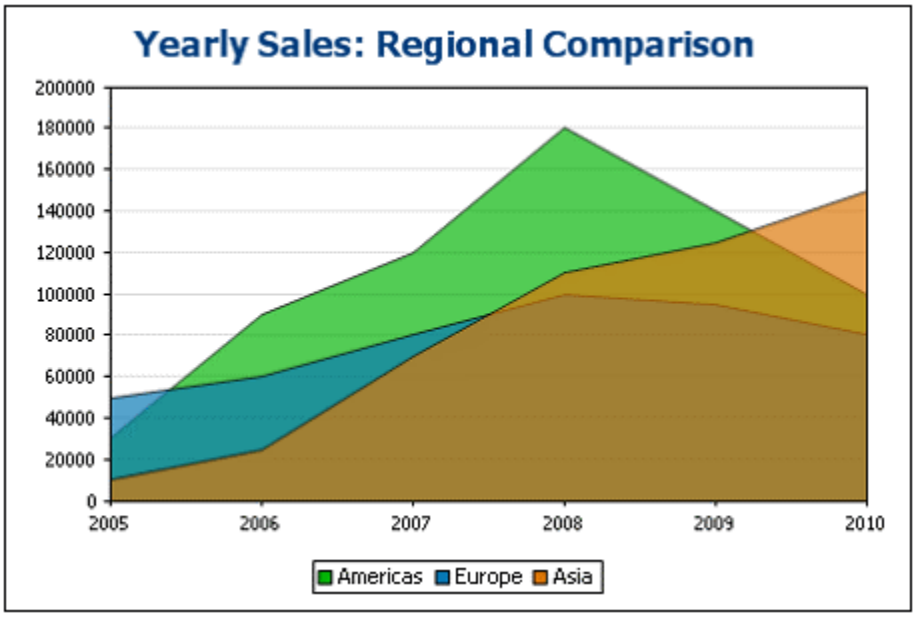
通常の折れ線グラフ(下左のスクリーンショット)と値付き折れ線グラフ(下右のスクリーンショット)の違いは、値付き折れ線グラフでは、X-軸に数値カラムしか使用することができない点にあります。折れ線グラフのX-軸に文字列を使用したい場合には、[種類の変更]ダイアログボックスの右ペインから折れ線グラフを選択してください。





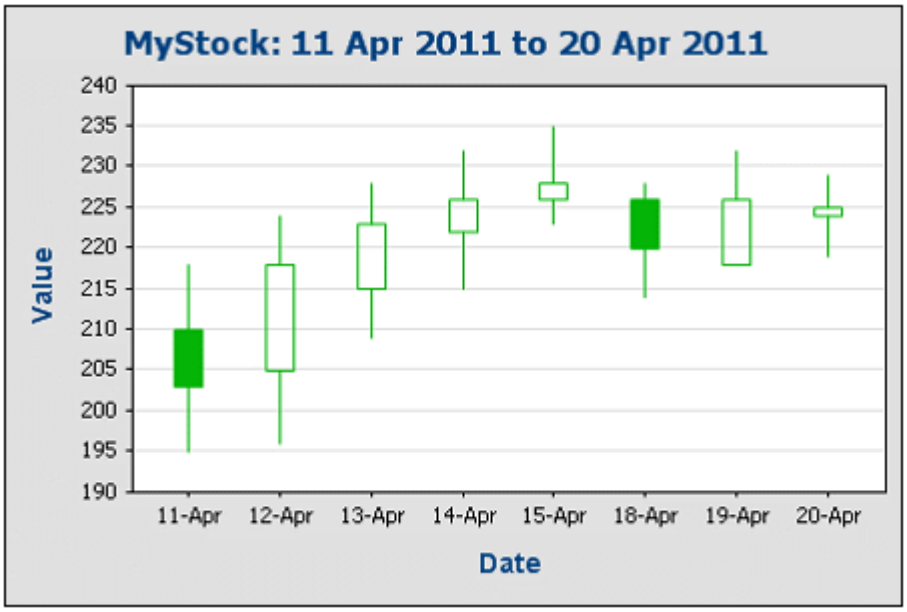
▼ *エリアチャート*

エリアチャートとは、線の下の面が色付けされる折れ線グラフの1種類です。エリアチャートはスタックすることができます。



▼ *ローソク足チャート*

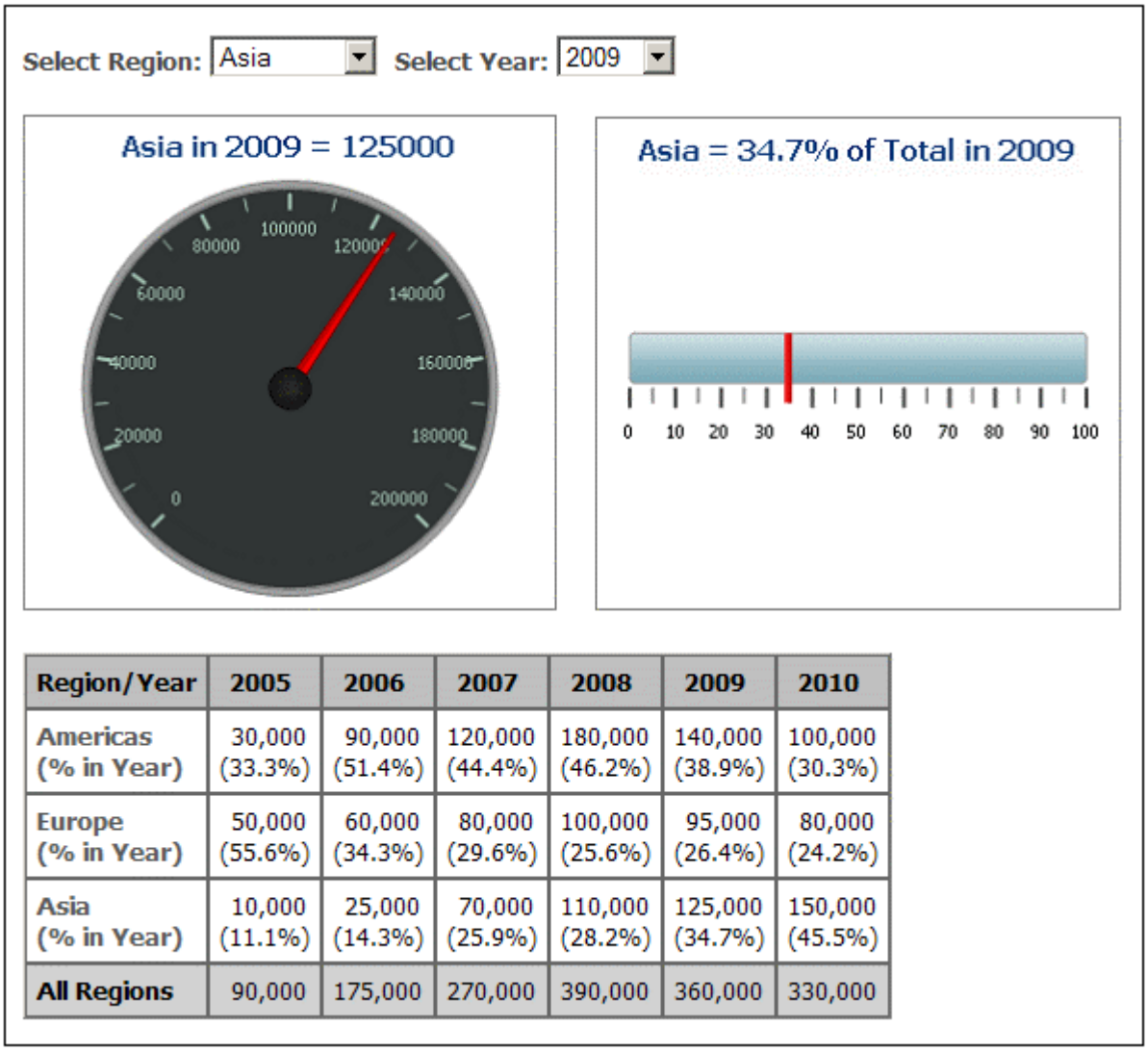
ローソク足チャートは、ある期間内における証券やデリバティブ、または通貨といった値の動きを表示するために使用されます。チャートには資産の動きが表示され、毎日の終値だけでなく、高値、安値、そしてオプションとして始値を表示することができます。4つの系列(始値、高値、低値、終値)を定義すると、チャートは以下のように表示されます。



▼ *ゲージチャート*

ゲージチャートには1つの値が最低ならびに最高値とともに表示されます。

(C) 2015-2021 Altova GmbH



Region/Year	2005	2006	2007	2008	2009	2010
Americas (% in Year)	30,000 (33.3%)	90,000 (51.4%)	120,000 (44.4%)	180,000 (46.2%)	140,000 (38.9%)	100,000 (30.3%)
Europe (% in Year)	50,000 (55.6%)	60,000 (34.3%)	80,000 (29.6%)	100,000 (25.6%)	95,000 (26.4%)	80,000 (24.2%)
Asia (% in Year)	10,000 (11.1%)	25,000 (14.3%)	70,000 (25.9%)	110,000 (28.2%)	125,000 (34.7%)	150,000 (45.5%)
All Regions	90,000	175,000	270,000	390,000	360,000	330,000

他の基本設定

チャート設定ペインでは、チャートのタイトルを設定することができます（*下のスクリーンショット参照*）。

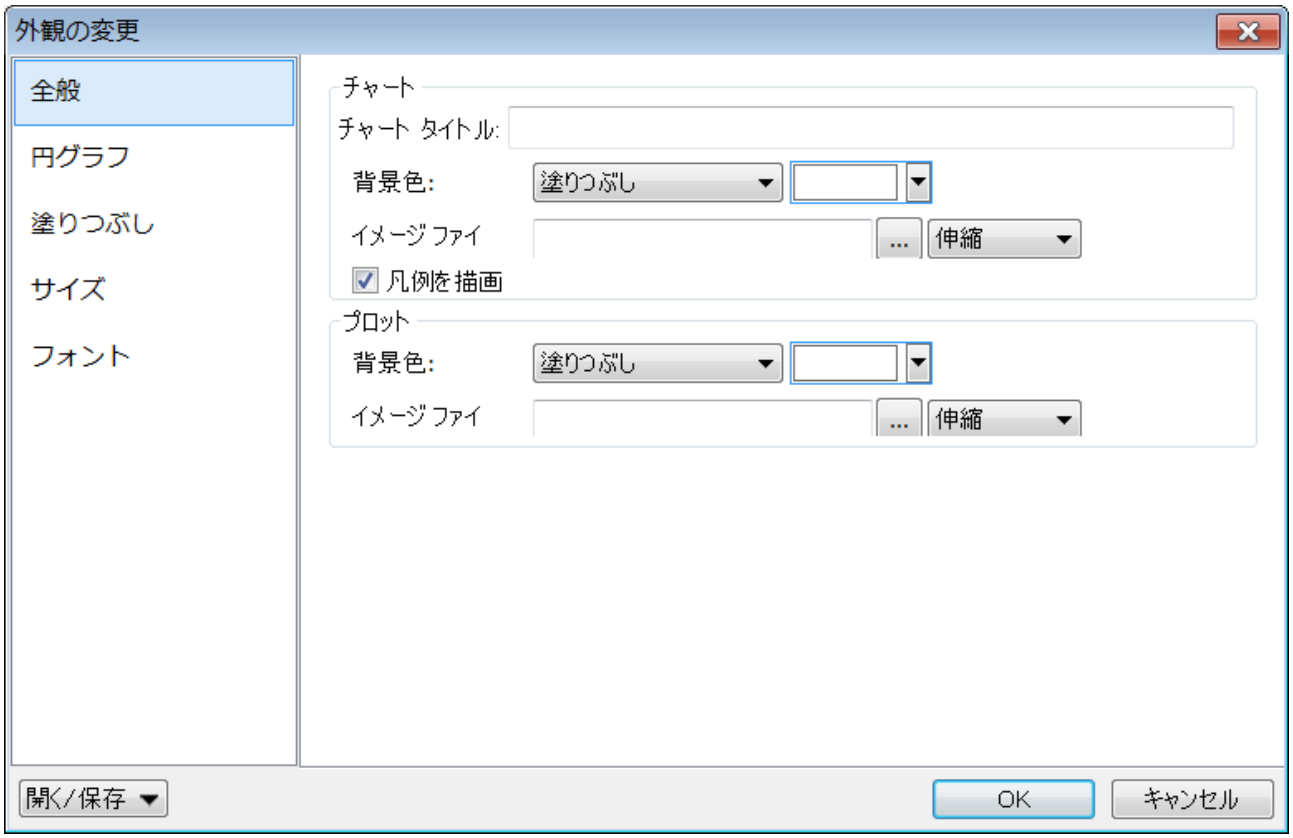
5.11.8.2 チャートの詳細設定

このセクション:

- 詳細設定へのアクセス 340
- 詳細設定の概要 340
- チャート設定のロード、保存、リセット 342

詳細設定へのアクセス

チャートの詳細設定にアクセスするには以下を行います: チャートウィンドウのツールバー内の「外観の変更」をクリックします。これにより特定のチャート種類のための外観の変更ダイアログが表示されます。 (*下のスクリーンショットは円グラフの外観の変更ダイアログを表示しています*)。



詳細設定の概要

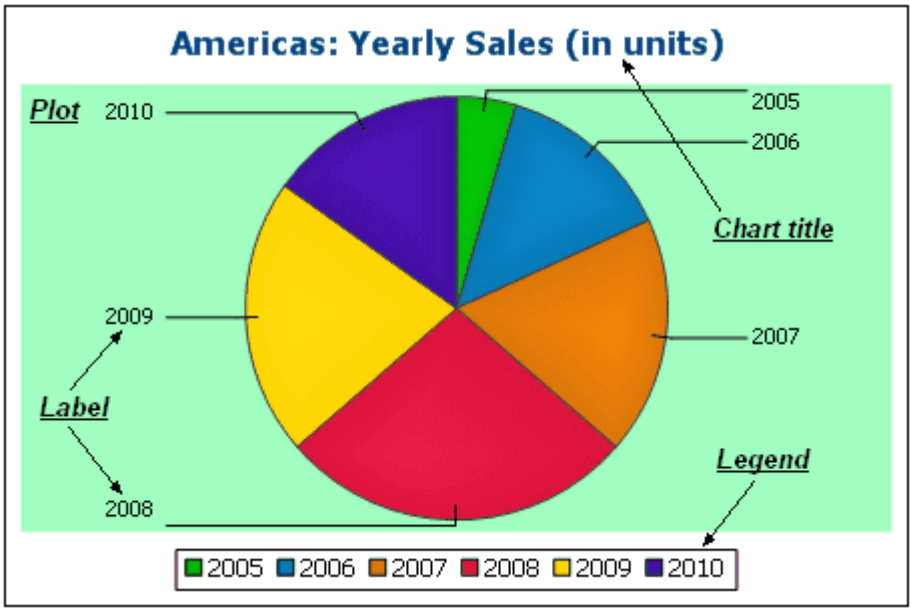
詳細設定はすべてのチャートの種類に共通するタブとチャートの種類に特有のタブにより整理されています。

チャート設定全般

▼ *全般*

チャートのタイトル (*下のスクリーンショット参照*) は基本設定と同じですが (*上を参照*) 詳細設定でも編集することができます。このダイアログの他の設定は、チャートの背景色とプロットです。下のスクリーンショットでは、プロットにはお薄い緑色の背景色が与えられています。チャートおよび/またはプロットの背景のイメージとしてイメージファイルを設定することもできます。このイメージはチャートまたはプロットのエリア全体をカバーするように拡大、(チャートまたはプロットの) 2 つのディメンションにズームがマッチするようにズーム、中央揃え、並べて表示することができます。凡例はチャートの配色コードのキーとなりオン/オフすることができます。

(C) 2015-2021 Altova GmbH



- *配色パターン*

  4つの設定済み配色パターンとユーザーにより定義される配色パターンを使用することができます。色を配色パレットに追加または削除することにより配色パターンを変更することができます。このタブにより選択される配色パターンはチャートで使用されます。

- *サイズ*

  ピクセルまたはパーセンテージ率としてチャートのさまざまな部分のサイズを設定することができます。

- *フォント*

  チャートのタイトル、凡例、ラベルのプロパティをこのタブで指定することができます。サイズはチャートのサイズのパーセンテージまたは絶対ポイントとして設定できます。

- *ロード/保存ボタン*

  設定はXML ファイルに保存され、正しい構成を持つXML ファイルからロードされることができます。構成を確認するには、チャートの設定を保存し XML 、ファイルを開きます。このボタンをクリックして、チャートの設定をディフォルトに再設定することもできます。

特定の種類のチャートの設定

- *円グラフ*

  設定: (i) 最初の扇形が描かれる角度; (ii) 扇形が描かれる方向; (iii) アウトラインの色; (iv) 色に対するハイライト有無(3D 円グラフ. ドロップシャドウ、透明度の使用); (v) ラベルの有無; および(vi) 値とパーセンテージがラベルに追加されるか、およびパーセンテージへの小数点の数。

- *棒グラフ*

  設定: *(全般)* X とY 軸を交換して横向きの棒グラフを生成する; *(棒)* 棒のアウトラインおよびドロップシャドウ(ドロップシャドウは 2D 棒グラフのみ); *(X-軸)* X-軸のラベルと配色と垂直方向の枠線; *(Y-軸)* Y-軸のラベル、配色と水平方向の枠線、値の範囲の表示、Y-軸の目盛; *(Z-軸、3D のみ)* Z-軸のラベルと配色; *(3D)* 垂直方向への傾斜、水平方向への回転、ビューの幅。

- *折れ線グラフ*

  設定: *(全般)* X とY 軸を交換してグラフの生成; *(ライン)* プロットポイントの有無; *(X-軸)* X-軸のラベルと配色と垂直方向の枠線; *(Y-軸)* Y-軸のラベル、配色と水平方向の枠線、値の範囲の表示、Y-軸の目盛。

- *ゲージチャート*

  設定: (i)ゲージが開始する角度とスケールの描引角度; (ii) 表示される値の範囲; (iii) メジャーおよびマイナーなメモリの間隔と配色; (iv) ダイアル、針、境界線の配色。

- *エリアチャート*

  エリアの透明度は0 (透明度無し) から255 (最大透明度) の値で設定することができます。この場合、スタックされていないエリアチャートでの透明度の使用は他のエリアの下にあるエリアをビューアが可視することができます。エリアのアウトラインを指定することができます。

- *ローソク足チャート*

  塗りつぶしの色は2つの場面で使用することができます: (i) 終値が始値よりも高い場合、(ii) 始値が終値よりも高い場合。後者の場合、系列の配色もオプションとして使用することができます。系列の色は、外観の変更ダイアログの配色パレットタブで指定することができます。

チャート設定のロード、保存、リセット

チャート設定は、XML ファイルに保存することのできるデフォルトの設定と異なります。これらの設定は、後にチャートの設定としてロードすることができ、時間と動力をセーブすることができます。「**ロード/保存**」ボタン (このセクションの最初のスクリーンショット参照 339) をクリックすると以下のオプションを使用することができます:

- **デフォルトとして設定**: 設定に加えられた変更を拒否し、**すべての** 設定セクションのデフォルトの設定を復元します。
- **ファイルからロード**: XML ファイルに以前保存された設定をインポートすることができます (*次のコマンドを参照*)。コマンドは、必要なファイルの場所を入力することのできる「参照」ダイアログを表示します。
- **ファイルに保存**: 「名前を付けて保存」ダイアログボックスを開きます。保存するXML ファイルを指定することができます。このファイルはデフォルト設定と異なる設定のファイルをリストします。

5.11.8.2.1 一般

「外観の変更」ダイアログボックスの一般セクションでは、チャートのタイトル、凡例の追加や削除、背景イメージ、更に棒グラフ、折れ線グラフ、エリアチャート、ローソク足チャートに対してはチャートの方向を指定することができます。

チャート

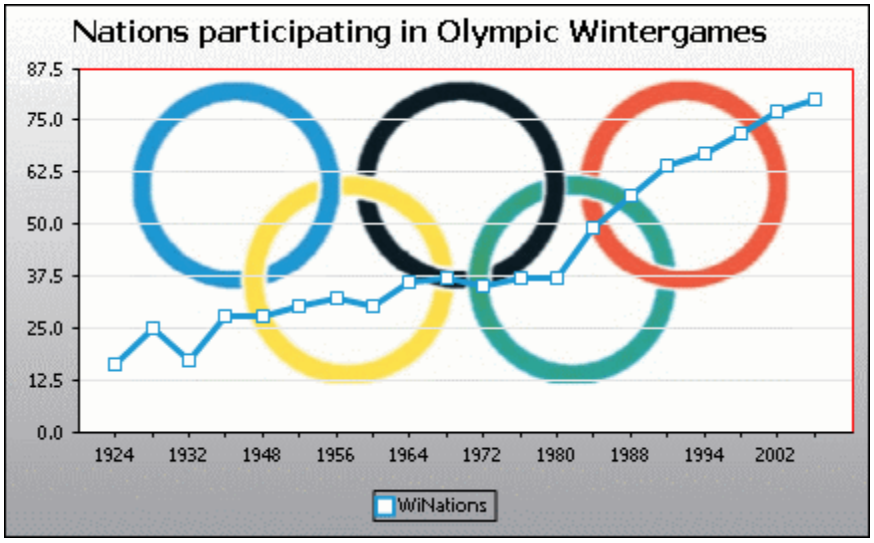
チャートタイトルフィールドにチャートを説明する様なタイトルを入力し、チャート全体に適用される背景色をドロップダウンリストから選択します。塗りつぶしや、垂直または水平グラデーション、そしてグラデーションでは2つの色を選択します。更に、背景色の代わりに背景イメージを指定し、ドロップダウンリストから表示オプションを選択することもできます:

- 伸縮: イメージがチャートの高さならびに幅に伸ばされて表示されます
- 合わせる: イメージが、アスペクト比を保ったままチャートのフレームに合うように表示されます
- 中央に表示: イメージがオリジナルのサイズで中央に表示されます
- 並べて表示: イメージがチャートより小さい場合、イメージの複製が背景エリアに表示されます

凡例を描画チェックボックスはデフォルトでオンになっており、チャート内に凡例を表示したくない場合はチェックを外してください。

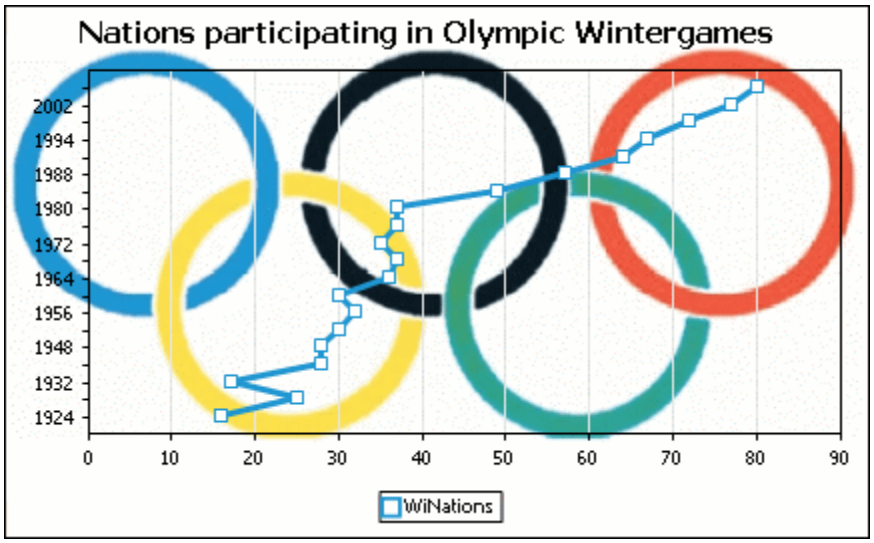
プロット

プロットとは実際のチャートが表示される領域です。プロットの周りに線をひいたり、プロット領域に対して背景色やイメージを指定することができます。以下のスクリーンショットでは、プロット領域の色は白のままでチャートの背景色がグレー(垂直グラデーション)に、境界線の色が赤に変更され、背景イメージが加えられています。



方向

少数の系列と、多くの値がデータに存在する場合、X-軸とY-軸を入れ替えた方が、より見やすい表示となることもあります。以下のスクリーンショットでは、プロットの背景色が「透明」にセットされ、イメージがチャートの背景に適応されていることにも注目してください。



このオプションは円グラフならびにゲージチャートにお適用できないことにご注意してください。

5.11.8.2.2 各チャートの機能

「外観の変更」ダイアログボックスでは、チャートの種類に関連した機能の定義を行うことができます。

(C) 2015-2021 Altova GmbH

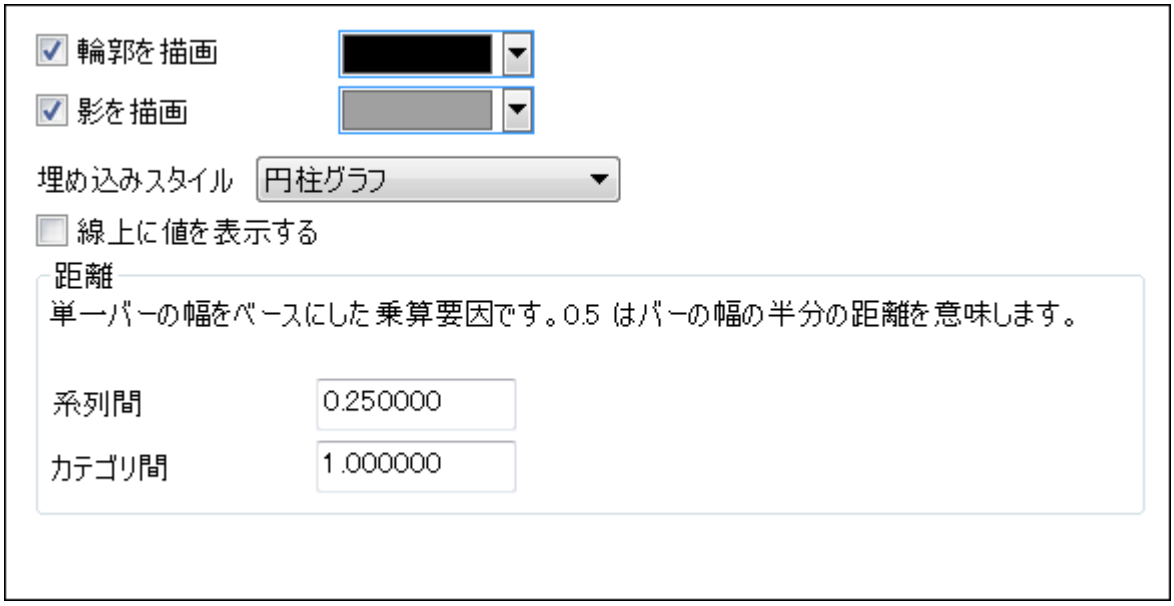
円グラフ

2D ならびに3D バージョンで行うことのできる設定にそこまで違いはありません。2D の円グラフではハイライトを付与することができます。

3D の円グラフでは、影の描画や透明度、3D の傾きを指定することができます。

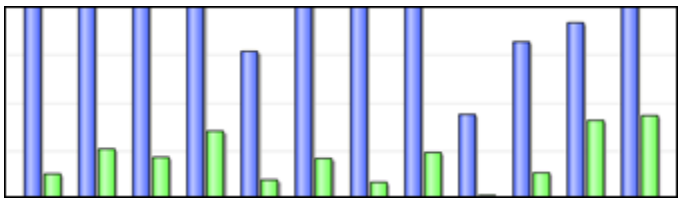
[開始角度] の値により、選択されたカラム内にある最初の行が表示されるチャートの位置を指定します。0° の角度が時計の12時方向に対応します。
凡例やラベルを表示したり、値やパーセントをラベルに付与したり、パーセントに使用される桁数を指定することができます。
[輪郭を描画] チェックボックスの隣から選択できる色により、チャートや個々の円グラフセグメントの周りに描かれる境界線の色を指定することができます。時計回りのチェックボックスにより、行の表示を時計回りの、または反時計回りに行うかを指定できます。
3D の円グラフでは、影の描画やその色、チャートの透明度、そして3D の傾きを指定することができます。2D のチャートでは、ハイライトを描画オプションにより、更なる側面をチャートに付与することができます。

棒グラフ



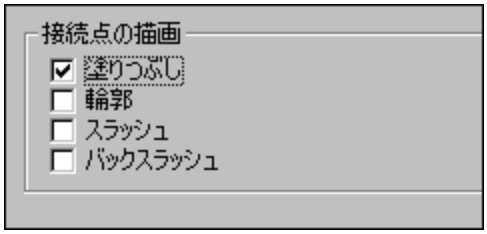
棒グラフのために以下を設定することができます:

- 棒に輪郭を追加し、色を定義することができます
- 2D の棒グラフでは、影を付けることができ、色を定義することもできます(3D 棒グラフではこのオプションを使用することはできません)。
- デフォルトで、棒の形式は、円柱ですが、「グラデーション(縦方向)」または「塗りつぶし」をドロップダウンリストから選択することができます。(このオプションは2D 棒グラフでのみ使用可能です)。
- (Y-軸上のバーの高さに対応する) 棒の値を棒上に表示することができます。値のフォントはフォント 設定内で指定することができます。(このオプションはスタックされたグラフではなく2D 棒グラフでのみ使用可能です)。
- 棒グループの系列と棒グループの距離は、単一の棒の幅の小数として指定することができます。例えば、下のスクリーンショットでは、青い系列と緑の系列から構成される棒グループが表示されており、系列間の距離は、棒の幅の25% (=0.25) に設定されています。棒グループの距離は棒の幅の100% (=1.0) に設定されています。(このオプションは2D 棒グラフでのみ使用可能です)。



(C) 2015-2021 Altova GmbH

折れ線グラフ

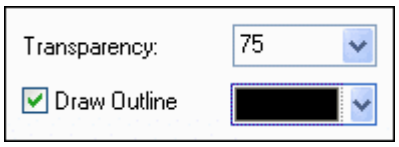


チャート内にある値を結びつける接続点を描画するには、接続点の描画グループボックスにあるチェックボックスのうち少なくとも1つにチェックを入れる必要があります。系列をマークするために、最大5つの形が使用されます: 正方形、ひし形、三角形、逆三角形、円。6つ以上の系列がチャート内に存在する場合、接続点のオプションを2つ以上選択することで、接続点の形を組み合わせて使用することができます。以下のスクリーンショットでは、塗りつぶしとスラッシュの2つが選択されており、6つ目以降の系列にはスラッシュが使用されます。

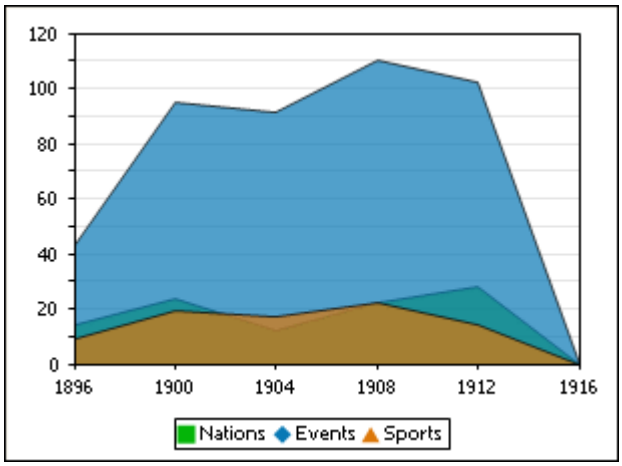
線を描くオプション (i) 接続点のみ (ii) 接続点と線により線を使用して描くことができます。

接続点は棒グラフと値付き棒グラフの両方で使用することができます。

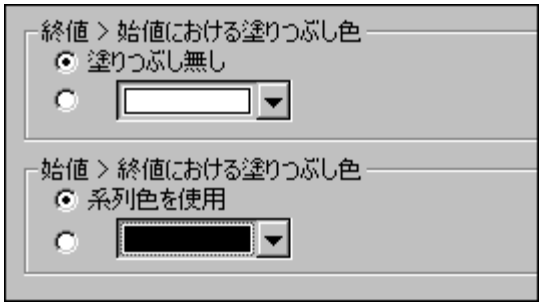
エリアチャート



エリアチャートでは透明度を指定することができ、1つの系列がチャートにある他の系列により隠れてしまうのを防ぐことができます。更に、データエリアに対して輪郭を描画、ならびに色を指定することができます。(*下のスクリーンショット参照*)。



ローソク足チャート



始値と終値の値が系列として既に定義されている場合、使用される色と、終値が始値よりも大きい場合、ローソクの塗りつぶしを行うかを指定することができます。

ゲージチャート

角グループボックスにある「開始」値により 0 のある位置と、「可動角度」によりチャートの表示に使用される角度が指定されます。値の範囲グループボックスでは、表示する最小ならびに最大値を指定することができます。大きな目盛り間隔では目盛りが値とともに、小さな目盛り間隔では価なしで表示され、それぞれに異なる色を指定することができます。色のグループボックスでは、ダイアルや針、チャートを囲む境界線や針のベース(中央部にある部分の針)の色を指定することができます。現在の値と追加ラベルを希望する角度で表示することができます。

5.11.8.2.3 色

選択されたチャートの種類により XMLSpy ではチャート内で使用される色の指定を行う2つのセクションが用意されます:

- 円グラフ、折れ線グラフ、エリアチャート、ローソク足チャートに対して使用される塗りつぶし
- ゲージチャートに対して使用される色の範囲

色スキーマ

「外観の変更」ダイアログボックスにある塗りつぶしセクションでは、(デフォルト、グレイスケール、カラー、パステルから成る) 4つの定義済み塗りつぶしセットが用意されており、カスタマイズすることができます。更に、独自の色を最初から作成することもできます。

最初の色が最初の系列に対して使用され、2 つ目の色が2 つ目の形式に対して使用されます。色を選択してマウスのドラッグ操作により別の位置に移動することで色の順番を変更することができます。新たな色を追加したり既にある色を削除するには、それぞれ対応するボタンをクリックしてください。ローソク足チャートでは、最初の色だけが使用されます。

1 つ以上のオーバーレイチャートレイヤーをチャートウィンドウに追加した場合、外観の変更ダイアログボックスに、デフォルトで選択されている [前のチャートレイヤーから継続して色を使用する] ラジオボタンが表示されます。

このラジオボタンが選択されていると、前のレイヤーから色が継承され、そのオーバーレイに対して独自の色を選択できなくなります。アクティブなレイヤーにおける系列は、その前のレイヤーで使用された色の集合が継続して使用されます。これにより、チャートウィンドウ内に表示される系列全てに異なる色を持たせ、より簡単に系列が識別できるようになります。

このオプションは任意のレイヤーで無効にすることができ、異なる色の集合を指定して、更に続くレイヤーにてそれらの色を継承することもできます。

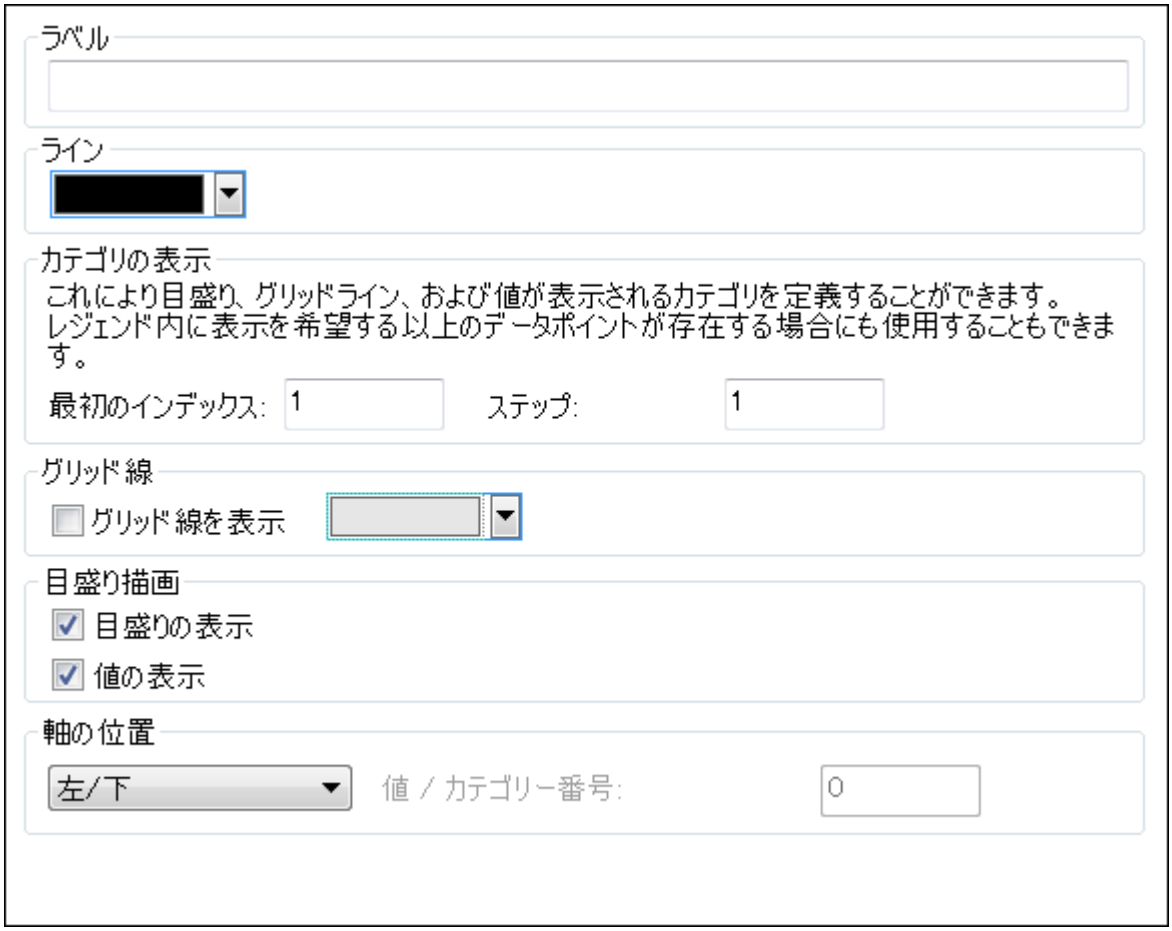
色の範囲

ゲージチャートでは、値の範囲に対して色を指定することで、ゲージの外観をカスタマイズすることができます。

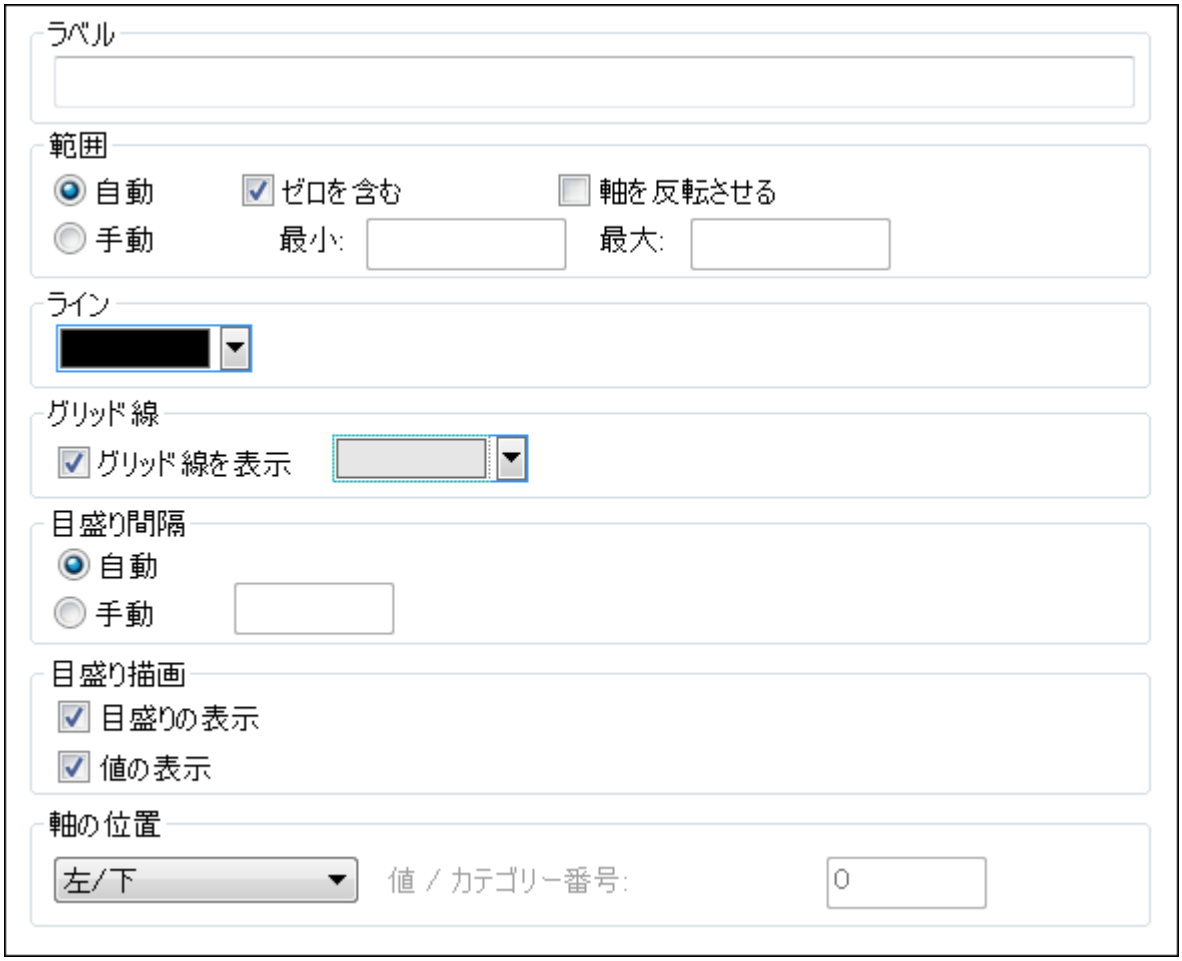
上のスクリーンショットで定義された設定により、ゲージチャートは以下のような外観になります:

5.11.8.2.4 X-軸

「外観の変更」ダイアログボックスのX-軸セクションでは、軸のラベルの入力や、軸ならびにグリッド(表示されている場合)の色を定義、目盛りや軸の値を表示するか指定することができます。このセクションにおける記述は全ての棒グラフ、折れ線グラフ、エリアチャートローソク足チャートに対して共通するものです。カテゴリの表示オプションにより、全てのカテゴリ(X-Axis 値) のサブセットが表示されるように指定することができます。すなわち、メモリ、グリッド線、選択されたカテゴリの値が表示されます。表示されたカテゴリのサブセットを以下に入力して作成することができます(i) 表示する最初の値のインデックス、および、(ii) ステップの数量・例えば、1900、1901、1902 ... 1999、2000、から開始する、101 件のカテゴリが存在する場合、最初のインデックスを 1 に、ステップを 10 に設定することにより、1900 年から 2000 年の間の10 年後との期間を表示することができます。



折れ線グラフの場合、値の範囲ならびに表示される目盛りの間隔を定義することもできます。



ラベル
ラベルフィールド内に入力されたテキストは、X-軸の説明文として軸の下に表示されます。

範囲
デフォルトでは範囲グループボックスの自動ラジオボタンが選択されています。チャートの一部分における詳細を表示したい場合、「手動」ラジオボタンを選択して、最小ならびに最大値を対応するフィールドに入力します。X-軸のために使用される列に0 が含まれない場合、「ゼロを含む」チェックボックスを無効化することができます。X-軸は系列内で使用することのできる最小の値と共に開始されます。「軸の反転」オプションによりX-値の値を反転することができます。例えば、 0 から 360 の間の値から実行される場合、このオプションの選択は、X-軸を生成します。これにより、 360 の値が元の値となり、X-軸の値が上がるごとに、0に近づきます。

ライン
チャート内に表示される軸の色は、「ライン」ドロップダウンリストから指定することができます。既に表示されている色を選択するか、「他の色」 ボタンをクリックして標準的な色を選択するか、カスタマイズされた色を作成することができます。カスタムタブにある「選択」 ボタンをクリックすると、スポイトが表示され、画面上に表示されている色を選択することができます。

グリッド線
グリッド線を表示チェックボックスが有効になっている場合、隣にあるドロップダウンボックスから色を選択することができます。

目盛り間隔
デフォルトで表示される目盛り間隔で都合が悪い場合は、目盛り間隔グループボックスにある手動ラジオボタンを有効にして、目盛り間の値を

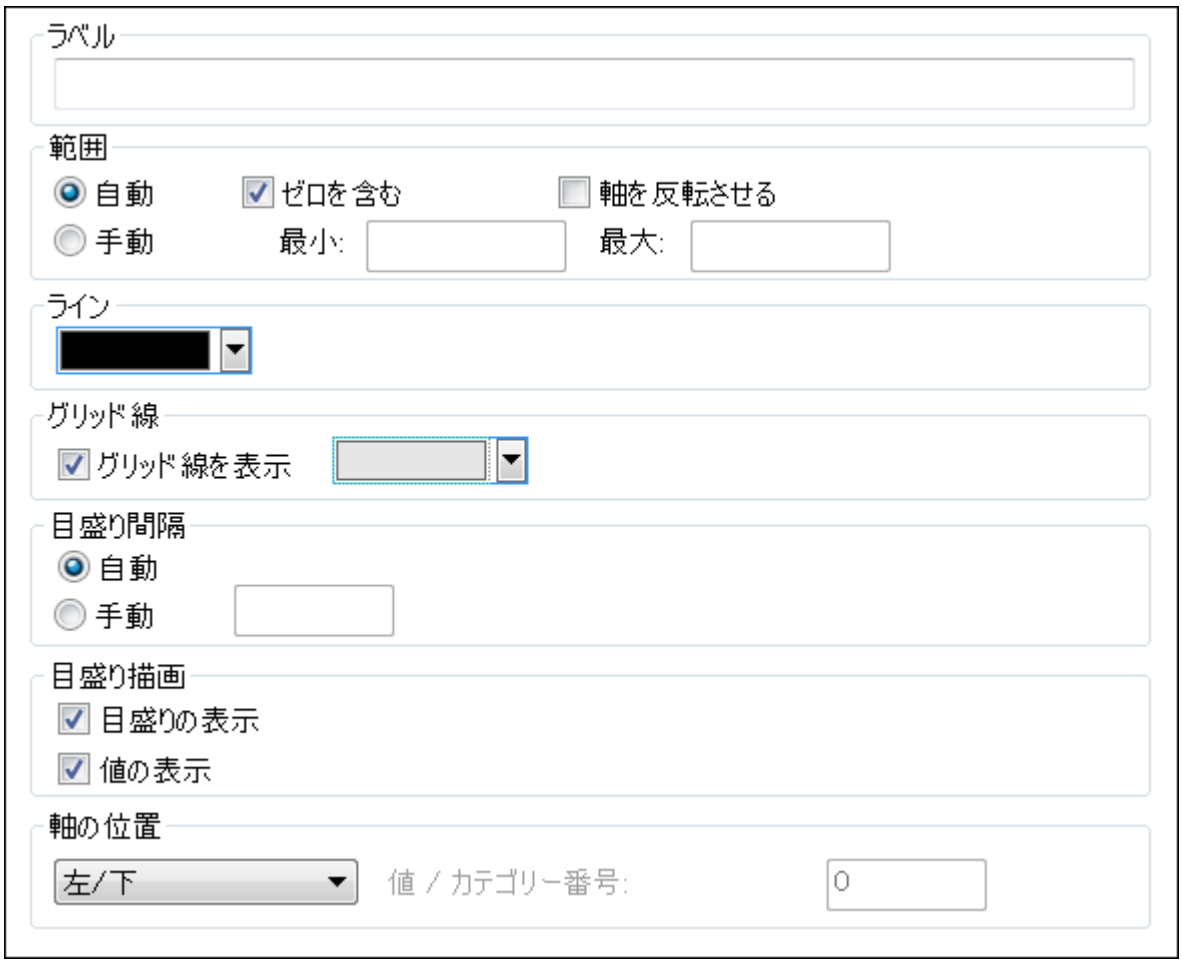
フィールドに入力してください。

目盛り描画
軸にある目盛りや、軸上にある値の表示を切り替えることができます。

軸の位置
軸が表示される位置をドロップダウンリストから選択することができます。「値 / カテゴリー番号」を選択すると、プロット内の自由な位置に軸を配置することができます。

5.11.8.2.5 Y-軸

「外観の変更」ダイアログボックスのY-軸セクションでは、軸のラベルの入力、軸の色や(表示される場合は)グリッド線の定義、値の範囲指定、目盛りの表示やその間隔、そして軸の値を表示するかなどを設定することができます。このセクションにおける記述は全ての棒グラフならびに折れ線グラフに対して共通するものです。



ラベル
ラベルフィールドに入力された値は、Y-軸の説明として軸の左側に表示されます。

範囲
デフォルトでは範囲グループボックスの自動ラジオボタンが選択されています。チャートの一部分における詳細を表示したい場合、「手動」ラジオボタンを選択して、**最小**ならびに**最大値**を対応するフィールドに入力します。Y-軸のために使用される列に0 が含まれない場合、「ゼロを含む」チェックボックスを無効化することができます。Y-軸は系列内で使用することのできる最小の値と共に開始されます。「軸の反転」オプションにより Y-値の値を反転することができます。例えば、 0 から 360 の間の値から実行される場合、このオプションの選択は、Y-軸を生成します。これにより、 360 の値が元の値となり、Y-軸の値が上がるごとに、0に近づきます。

ライン
チャート内に表示される軸の色は、「**ライン**」ドロップダウンリストから指定することができます。既に表示されている色を選択するか、「**他の色**」 ボタンをクリックして標準的な色を選択するか、カスタマイズされた色を作成することができます。カスタムタブにある「**選択**」 ボタンをクリックすると、スポイトが表示され、画面上に表示されている色を選択することができます。

グリッド線
グリッド線を表示チェックボックスが有効になっている場合、隣にあるドロップダウンボックスから色を選択することができます。

目盛り間隔
デフォルトで表示される目盛り間隔で都合が悪い場合は、目盛り間隔グループボックスにある手動ラジオボタンを有効にして、目盛り間の値をフィールドに入力してください。

目盛り描画
軸にある目盛りや、軸上にある値の表示を切り替えることができます。

軸の位置
軸が表示される位置をドロップダウンリストから選択することができます。「値 / カテゴリー番号」を選択すると、プロット内の自由な位置に軸を配置することができます。

5.11.8.2.6 Z-軸

「外観の変更」ダイアログボックスのZ-軸セクションでは、軸のラベルの入力や、軸ならびに色を定義、軸上の目盛りを表示するか指定することができます。このセクションにおける記述は全ての3D 棒グラフ(3D 棒グラフならびにグループ化された3D 棒グラフ) 対して共通するものです。

ラベル
ラベルフィールドに入力された値は、Z-軸の説明として軸の右側に表示されます。

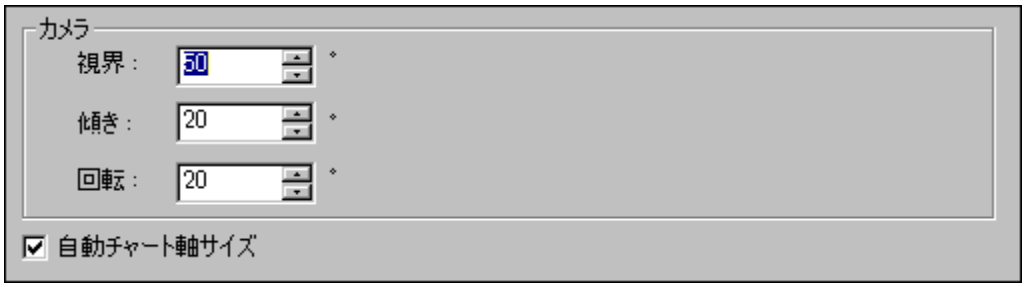
ライン
チャート内に表示される軸の色は、「**ライン**」ドロップダウンリストから指定することができます。既に表示されている色を選択するか、「**他の色**」 ボタンをクリックして標準的な色を選択するか、カスタマイズされた色を作成することができます。カスタムタブにある「**選択**」 ボタンをク

リックすると、スポイトが表示され、画面上に表示されている色を選択することができます。

メモリの描画
目盛りの表示を切り替えることができます。

5.11.8.2.7 3D アングル

「外観の変更」ダイアログボックスの3D アングルセクションでは、チャートの3D 的な外観を3D 棒グラフに対して設定することができます。



視界 オプションにより図を近くまたはお遠くから観察できるように表示することができます。値は1 から120 までの範囲が有効です。高い値により図が遠い場所から観察されているように表示されます。

傾き の値によりX-軸に対する角度が決定され、「回転」の値によりY-軸に対する角度が決定されます。対応するチェックボックスにチェックを入れることで、チャート軸の大きさをチャートのウィンドウに自動的に合わせることができます。

自動チャート軸サイズ チェックボックスが選択されると、現在表示されているチャートの大きさに対して最適なX-軸ならびにY-軸の大きさがXMLSpy により計算されます。チャートの幅と高さは、チャートウィンドウの大きさを変更することで動的に変更されます。

5.11.8.2.8 サイズ

「外観の変更」ダイアログボックスの大きさセクションでは、様々な余白や軸、またはゲージ目盛りの間隔を定義することができます。以下に示す全てのプロパティが全てのチャートに対して使用できるわけでは無いことにご注意してください。

一般
外部の余白 プロットとチャートウィンドウ端の空間
タイトルからプロット チャートタイトルとプロット上部の端との空間
凡例からプロット プロット下部の端と凡例との空間

円グラフ
プロットからラベル 円グラフにて、円グラフの左端ならびに右端とラベル間の空間
グラフの高さ 3D 円グラフの高さ
グラフの影 3D 円グラフにて(有効化されていた場合)表示される影の長さ

X-軸
X-軸から軸ラベル 棒グラフと折れ線グラフにて表示されるX-軸とラベル間の空間
X-軸からプロット 2D 棒グラフと折れ線グラフにて表示されるX-軸とプロット間の空間
X-軸目盛りサイズ 棒グラフと折れ線グラフにて表示されるX-軸上の目盛り長さ

Y-軸

Y-軸から軸ラベル	棒グラフと折れ線グラフにて表示されるY-軸とラベル間の空間
Y-軸からプロット	2D 棒グラフと折れ線グラフにて表示されるY-軸とプロット間の空間
Y-軸目盛りサイズ	棒グラフと折れ線グラフにて表示されるY-軸上の目盛り長さ

Z-軸

Z-軸から軸ラベル	3D 棒グラフにて表示されるZ-軸とラベル間の空間
Z-軸目盛りサイズ	3D 棒グラフにて表示されるZ-軸上の目盛り長さ

ライン描画

接続点のサイズ	折れ線グラフにてチャート内の値をマークする接続点の大きさ
ラインの幅	折れ線グラフにてのラインの幅

3D 軸サイズ

ベースの手動X-軸サイズ	3D 棒グラフにおいてX-軸の長さとチャートウィンドウの大きさの関係を定義します。3D アングルセクションにある**自動チャート軸サイズ**のチェックを外さなければサイズが自動的に計算されることご注意してください
ベースの手動Y-軸サイズ	3D 棒グラフにおいてY-軸の長さとチャートウィンドウの大きさの関係を定義します。3D アングルセクションにある**自動チャート軸サイズ**のチェックを外さなければサイズが自動的に計算されることご注意してください
Z-軸系列余白	3D 棒グラフのZ-軸における各系列間の距離

ゲージ

境界線の幅	円ゲージチャートのゲージ周りに表示される境界線の幅

ゲージ間隔

領域線から目盛りまでの距離	円ゲージチャートにおいて、領域内側の端と値を表示する目盛りの空間
大きな目盛りの長さ	円ゲージチャートに表示される(値を持つた)大きな目盛りの長さ
大きな目盛りの幅	円ゲージチャートに表示される(値を持つた)大きな目盛りの幅
小さな目盛りの長さ	円ゲージチャートに表示される値を持たない目盛りの長さ
小さな目盛りの幅	円ゲージチャートに表示される値を持たない目盛りの幅

ゲージ針

針の長さ	円ゲージチャートにおける針の長さ(パーセントはゲージの直径から計算されるため、50% を超えた値を選択すると、針の先端がゲージの外側に出てしまうことご注意してください！)
ベースにおける針の幅	円ゲージチャートの中心における針の幅(太さ)
針ベースの半径	円ゲージチャート中央部に表示されるベースの半径

ゲージ色の範囲

境界線から色の範囲までの距離	円ゲージチャートにおける境界線内側にある端と、色の範囲 350 との間にある空間
色の範囲幅	円ゲージチャートにおけるカスタマイズ可能な色が表示される範囲の幅(パーセントはゲージの直径から計算されることご注意してください！)

ゲージの値

中央にオフセット	ゲージの値が表示される中央からの距離

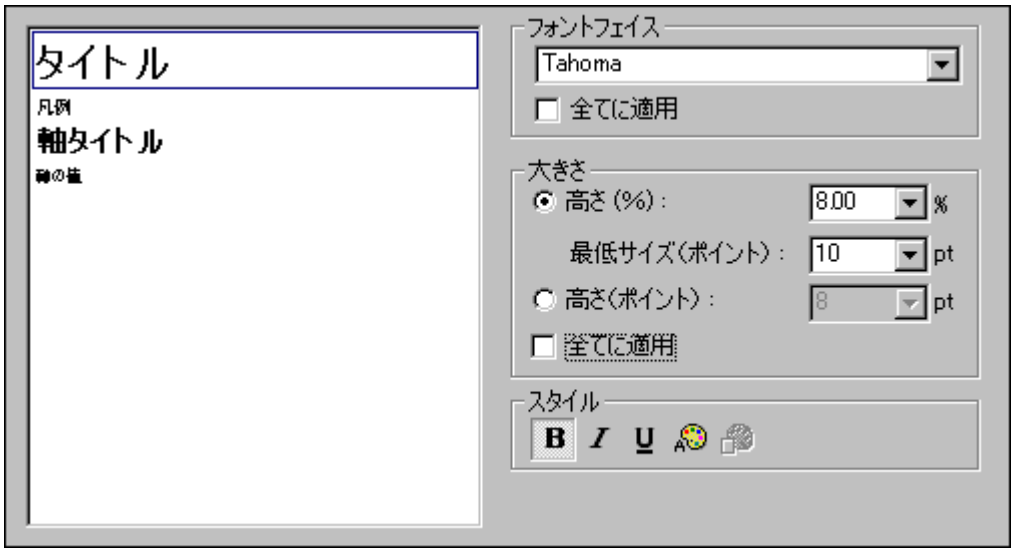
追加の値

(C) 2015-2021 Altova GmbH

中央にオフセット （ゲージチャート設定344内で定義されている）追加ラベルが表示される中央からの距離

5.11.8.2.9 フォント

「外観の変更」ダイアログボックスのフォントセクションではチャートウィンドウ内のオブジェクトに使用されるフォント設定を行うことができます。



フォント設定

フォントセクションではチャートウィンドウ内に表示される各要素のフォントフェイス、大きさ、そしてスタイルを変更することができます。大きさはチャートサイズに対するパーセントとして定義することができるとともに、最低サイズをポイントで指定、または絶対サイズをポイントで指定することができます。同一のフォントや大きさを全てのテキスト要素に適用したい場合、[全てに適用] チェックボックスにチェックを入れます。

リストボックス内にある要素は以下のように定義されます:

- タイトル: チャートの名前
- 凡例: チャート内で使用されている色の説明
- ラベル: 円グラフにおける各パイの説明
- 軸タイトル: 棒グラフおよび折れ線グラフにおけるX-、Y-、Z-軸の名前
- 軸の値: 棒グラフおよび折れ線グラフの軸上に表示される単位
- 目盛りの値: ゲージチャートにて表示される単位

5.11.9 エクスポート

「**エクスポート**」ボタンをクリックすることで、以下のオプションを利用することができます:

- **チャートをファイルに保存** : イメージの形式をPNG、GIF、BMP、JPG から選択することができます。
- **現在のイメージをクリップボードにコピー** または **サイズ変更されたイメージをクリップボードにコピー** : チャートのイメージをクリップボードにコピーして、他のアプリケーションで使用することができます。
- **このチャートを印刷** : イメージをネットワーク上のプリンターに送信します。イメージの高さと幅は、それぞれページサイズとの比率（パーセンテージ）で指定することができます。

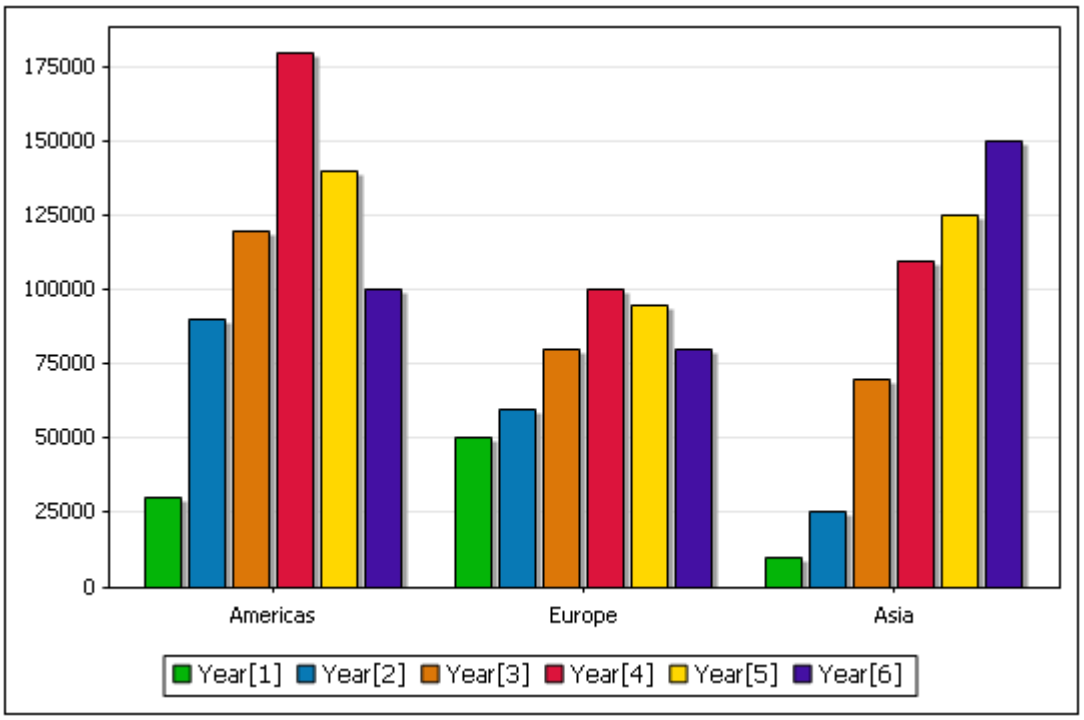
- **XSLT または XQuery コードをクリップボードにコピー**：XSLT または XQuery フラグメントを作成します。両者とも、実質は Altova 拡張関数の `CreateChart` となります。この関数は、他の Altova 拡張関数にて使用することができ、XMLSpy または RaptorXML エンジンにて処理することでチャートの生成を行うことができます。各フラグメントには、`CreateChart` 拡張関数の使用を助けるための例がコメントアウトされた形で作成されます。

5.11.10 チャートの例：基本

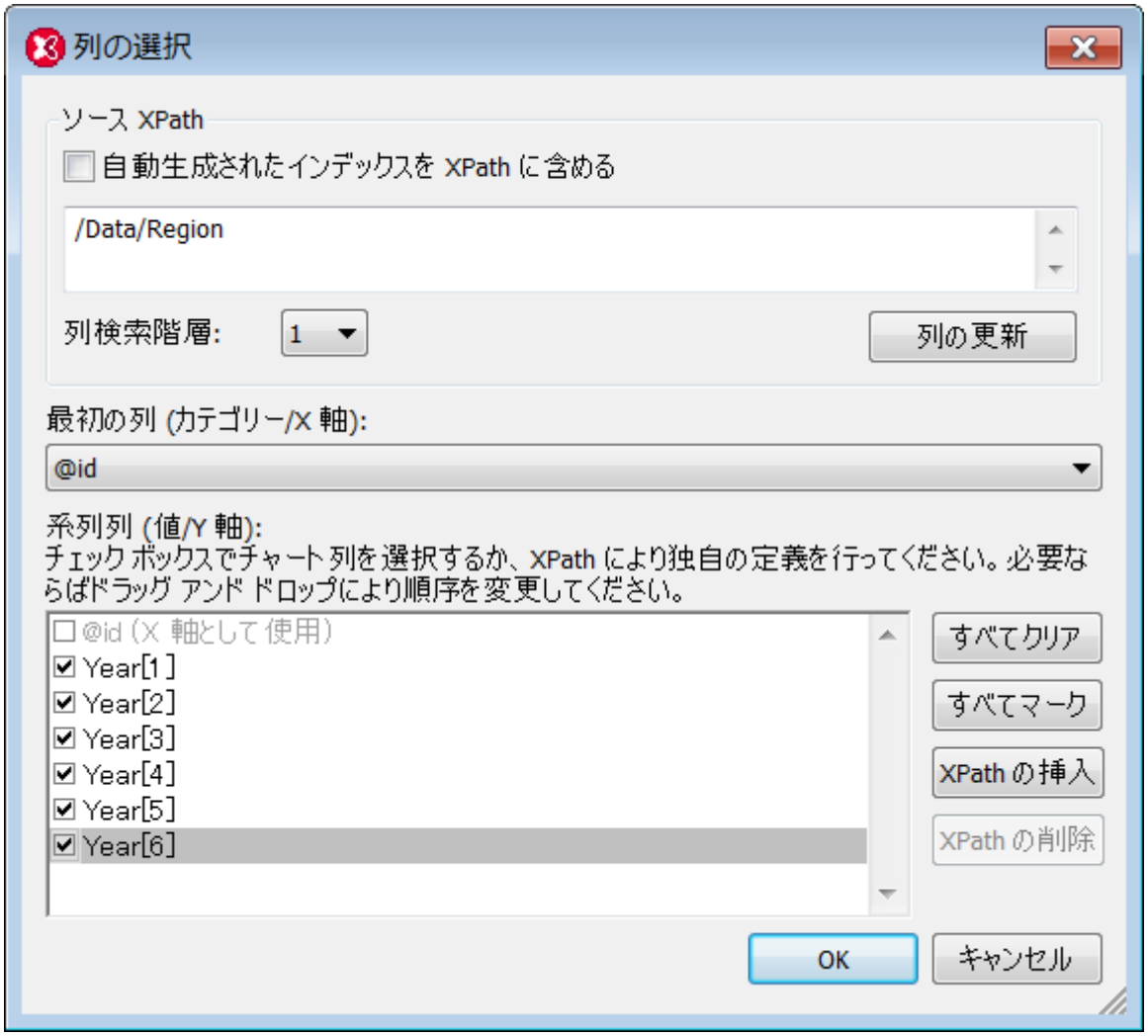
以下の XML ドキュメントを考えてみましょう。このファイルは `YearlySales.xml` というファイル名で `C:\Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples¥Tutorial` に収められています。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
       <Region id="Americas">
              <Year id="2005">30000</Year>
              <Year id="2006">90000</Year>
              <Year id="2007">120000</Year>
              <Year id="2008">180000</Year>
              <Year id="2009">140000</Year>
              <Year id="2010">100000</Year>
       </Region>
       <Region id="Europe">
              <Year id="2005">50000</Year>
              <Year id="2006">60000</Year>
              <Year id="2007">80000</Year>
              <Year id="2008">100000</Year>
              <Year id="2009">95000</Year>
              <Year id="2010">80000</Year>
       </Region>
       <Region id="Asia">
              <Year id="2005">10000</Year>
              <Year id="2006">25000</Year>
              <Year id="2007">70000</Year>
              <Year id="2008">110000</Year>
              <Year id="2009">125000</Year>
              <Year id="2010">150000</Year>
       </Region>
</Data>
```

ここで X-軸に3つの地域が表示され、各地域の年間売上がプロットされるようなグラフの作成を行ないます。グラフは以下の棒グラフにあるようなものになります。



`Region` 要素をソース XPath として選択するため、このチャートの作成は簡単に行うことができます。ソース XPath 条件式からは 3 つのアイテム(3 つの `Region` 要素)を含むシーケンスが取得されます。各 `Region` 要素は X-軸ならびに Y-軸のデータ選択におけるコンテキストノードとなります。



系列に関しては、各地域にある `Year` 要素を使用するので、列の検索階層は1で十分となります。`Region` 要素の `id` 属性をX-軸に使用します。従って、`id` 属性の値がX-軸の要素のラベルとして使用されます。チャートには存在する全ての `Year` 要素を表示したいので、`Year` 系列は全てチェックします。

「OK」ボタンをクリックすることで、目的のチャートが表示されます。より高度なチャートに関する情報は、チャートの例: 詳細 360 を参照ください。

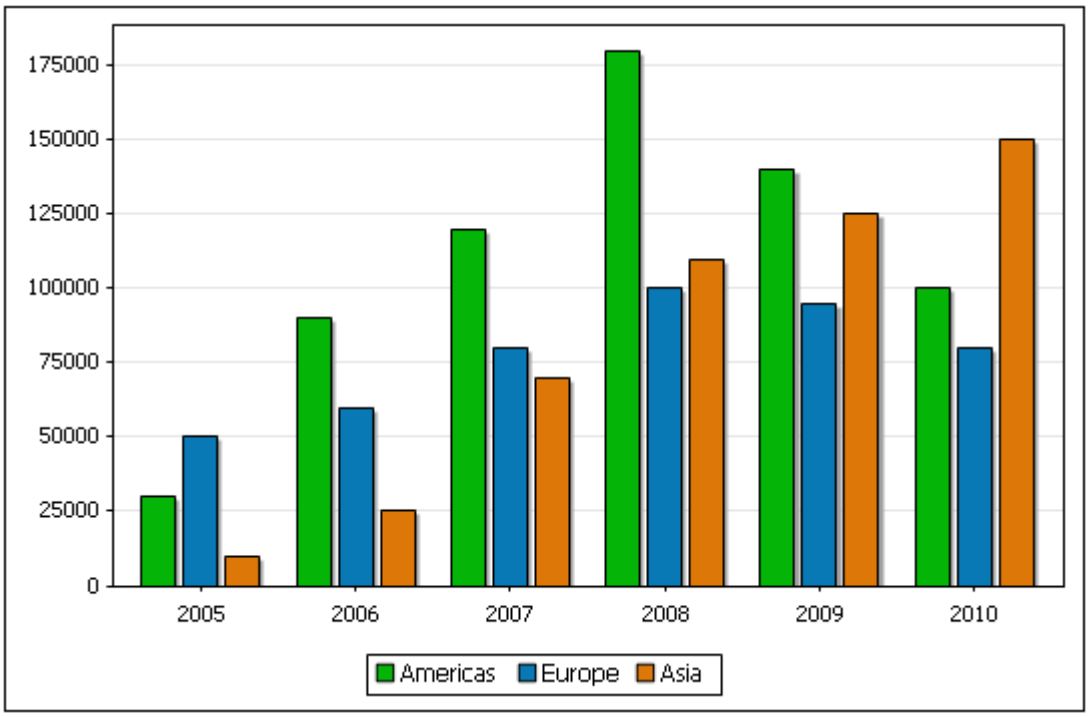
5.11.11 チャートの例: 詳細

以下のXMLドキュメントを考えてみましょう。このファイルは `YearlySales.xml` というファイル名で `C:\Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples¥Tutorial` に収められています。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
        <Region id="Americas">
                <Year id="2005">30000</Year>
```

```
            <Year id="2006">90000</Year>
            <Year id="2007">120000</Year>
            <Year id="2008">180000</Year>
            <Year id="2009">140000</Year>
            <Year id="2010">100000</Year>
        </Region>
        <Region id="Europe">
            <Year id="2005">50000</Year>
            <Year id="2006">60000</Year>
            <Year id="2007">80000</Year>
            <Year id="2008">100000</Year>
            <Year id="2009">95000</Year>
            <Year id="2010">80000</Year>
        </Region>
        <Region id="Asia">
            <Year id="2005">10000</Year>
            <Year id="2006">25000</Year>
            <Year id="2007">70000</Year>
            <Year id="2008">110000</Year>
            <Year id="2009">125000</Year>
            <Year id="2010">150000</Year>
        </Region>
</Data>
```

ここで、X-軸に年が表示され、毎年ごとに地域間の年間売上比較を行えるようなグラフを作成します。グラフは以下の棒グラフにあるようなものになります。

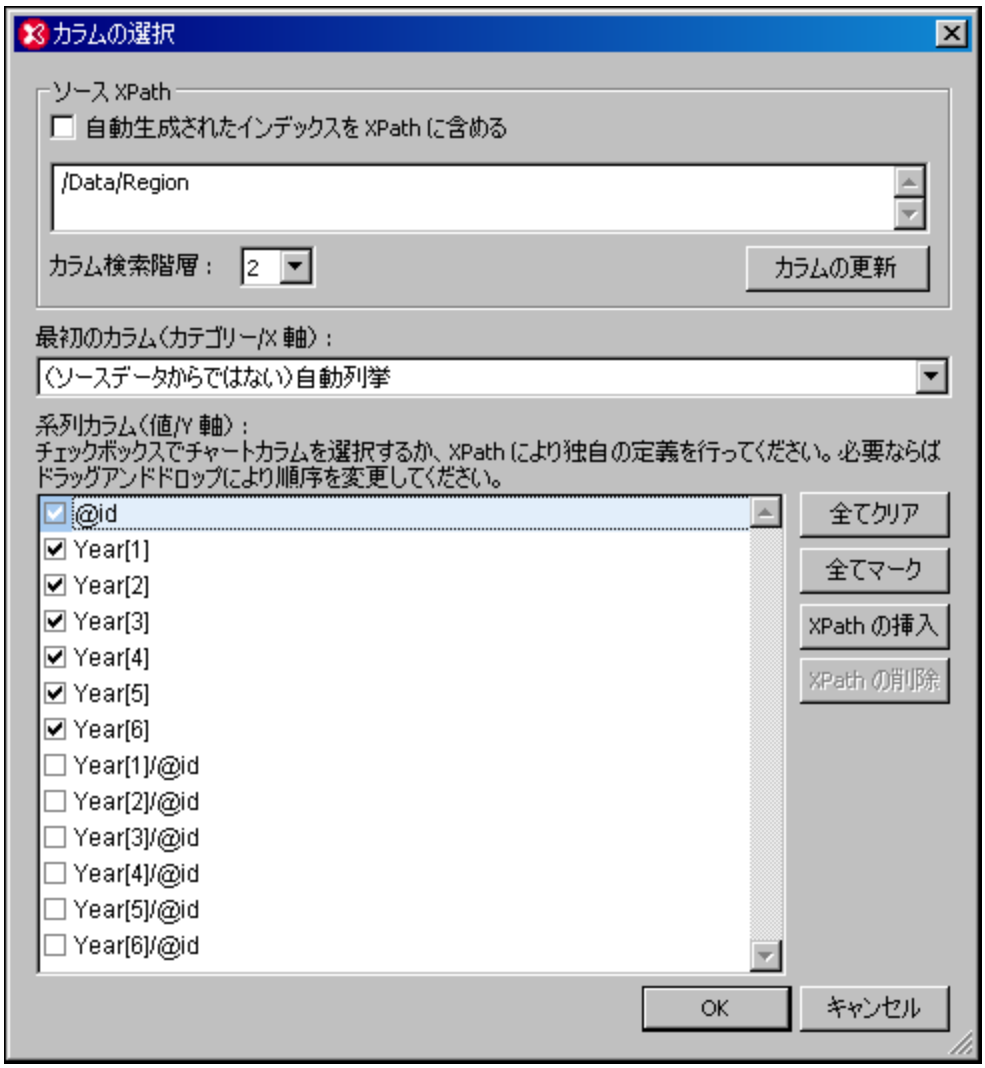


このグラフを作成する2つの方法を説明します。これら2つの説明を通して、異なるデータ選択パラメーターを組み合わせることで目的のグラフを作成する方法を学習します。

方法その1：軸の値を変更

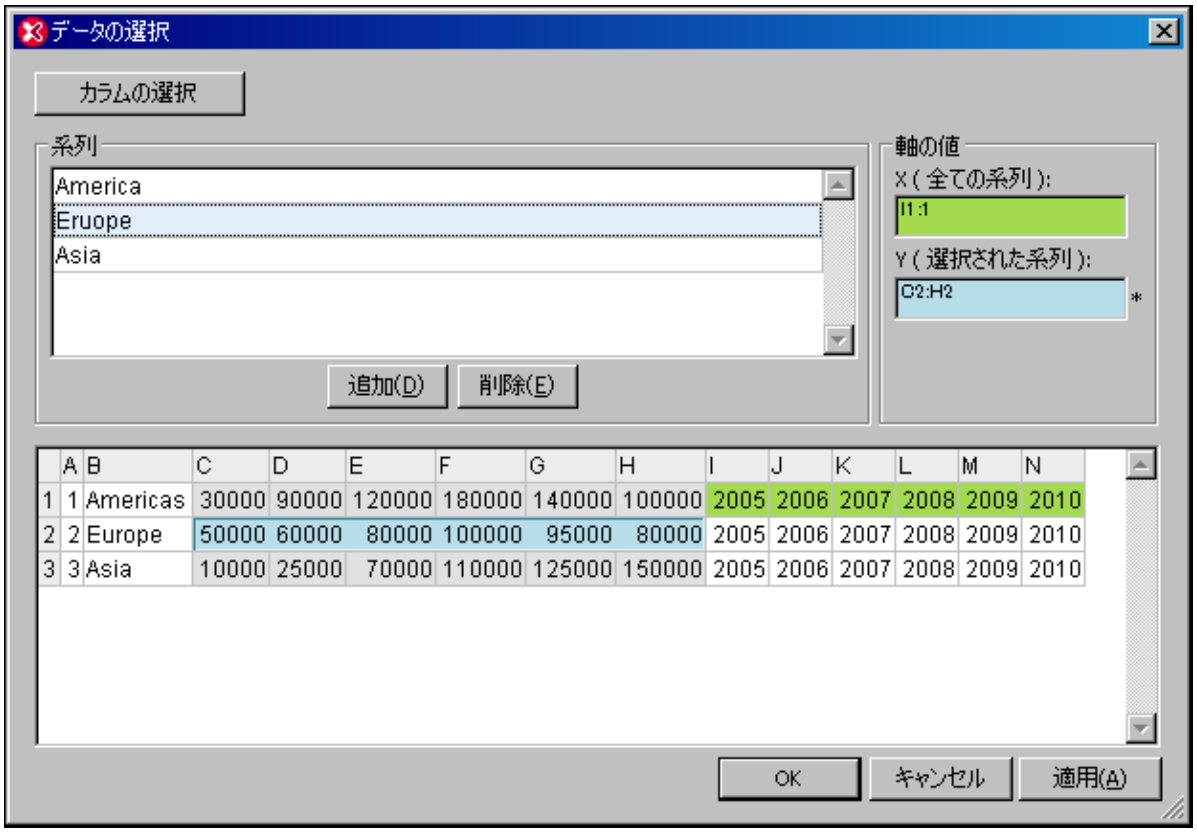
最初の方法では、列の選択ダイアログにて選択された軸をデータの選択ダイアログにて修正します。ここで注意しなければならない唯一のことは、必要なデータは全てデータの選択ダイアログのチャートデータテーブルにて使用可能になっているということです。

1. 列の選択ダイアログにて、必要な全てのノードがX-軸、Y-軸として選択できる状態であることを確認してください。以下のスクリーンショットにて、列検索階層が2にセットされていることにご注目してください。これで `Year/@id` 属性を選択することができるようになります。



2. （以下のスクリーンショットにある）データの選択ダイアログにて、チャートデータテーブルには以下の列が含まれています：最初の列がX-軸の選択（自動列挙した選択）で、その他の列が、`Region/@id`、`Year` 要素のコンテンツ、そして `Year/@id` 属性からなる、系列（Y-軸）の列。以下の2 つの点に注意してください：（i）テーブルには3 つの行しか存在せず、X-軸の要素数は3 つになります（しかし我々がX-軸に欲しいのは6 年分の（つまり6 つの）要素です）。（ii）系列には13 の列があります。

(C) 2015-2021 Altova GmbH



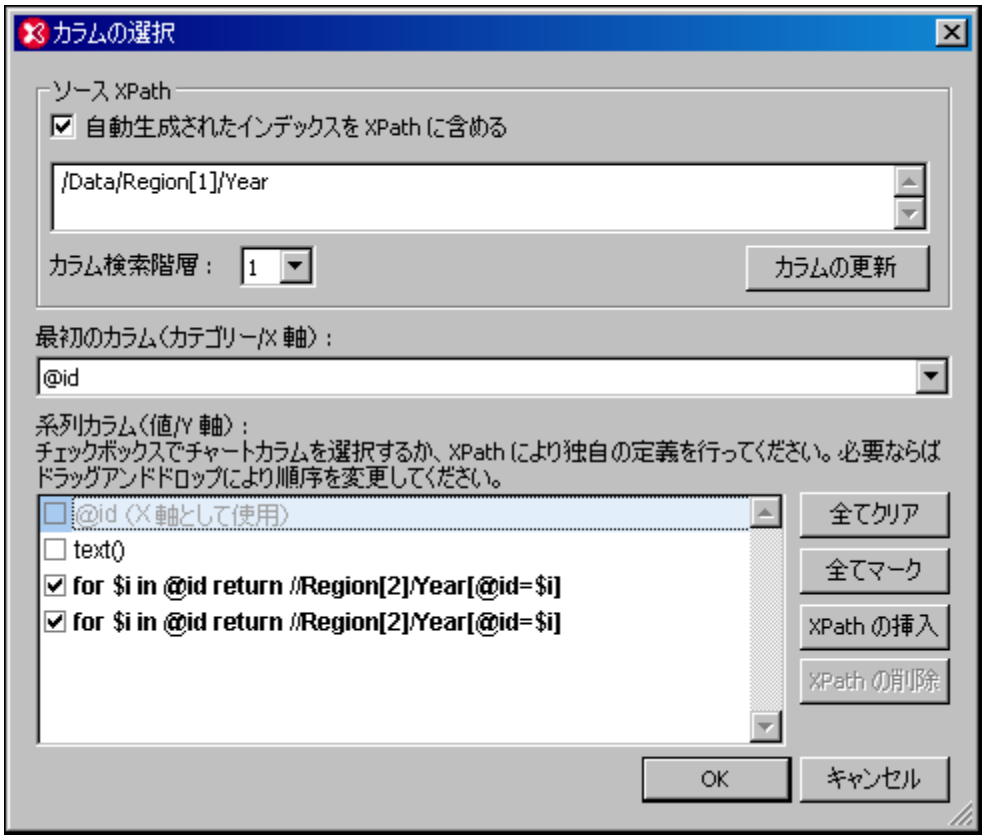
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	1	Americas	30000	90000	120000	180000	140000	100000	2005	2006	2007	2008	2009	2010
2	2	Europe	50000	60000	80000	100000	95000	80000	2005	2006	2007	2008	2009	2010
3	3	Asia	10000	25000	70000	110000	125000	150000	2005	2006	2007	2008	2009	2010

3. 系列ペインでは、13 あるうちの系列行のうち、任意の10 個を削除し、残った3 つの名前を、スクリーンショットにあるように America、Europe、Asia に変更します。ここで選択された順序がX-軸の要素にて表示されるラベルの順序になります。
4. 系列ペインにて、America の系列を選択します。軸の値ペインにてX-軸のテキストボックスをクリックして、修正を開始します。チャートデータテーブルの I1 をクリックして、そのまま N1 までドラッグします。Y-軸テキストボックスにて C1:H1 と入力するか、ドラッグすることでチャートデータテーブルの C1 から H1 までを選択します。
5. Europe とAsia の系列に対しては、C2:H2 ならびに C3:H3 をそれぞれY-軸に対して選択します。X-軸の選択は America の場合と同様になります。
6. 「OK」をクリックすると、目的のグラフが生成されます。

メモ: X-軸のラベル数が6になったため、、(チャートデータテーブルにある行の数からデフォルトで定義された)X-軸の要素も3 から6 に増加します。

方法その2:XPath 条件式により系列を生成する

2つめの方法では、系列の生成のためにXPath 条件式を挿入します。これは、目的の系列に対応するノードを、(以下のスクリーンショットにあるように)ソースXPath が子孫として持たないために必要な操作となります。それでもソースXPath は6つのX-軸要素を(最初の Region 要素の Year 要素を選択することで)生成します。かつこの [1] により、最初の Region 要素を選択するには、インデックスに含めるチェックボックスを有効にして、列の更新ボタンをクリックしてください。

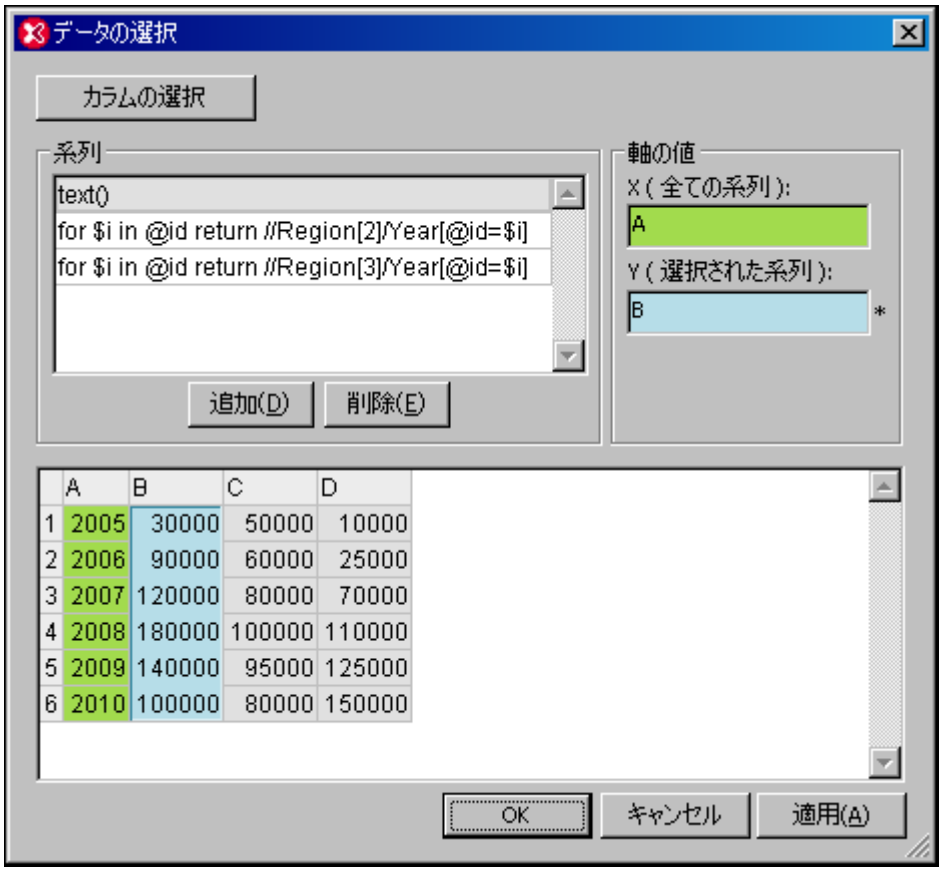


列の選択ダイアログにて、Region[1]/Year 要素は、`@id` と `text()` という2つの子孫しか持っていないことが確認できます。@ id 属性をX-軸に選択し、6つのX-軸要素に対して、正しいX-軸のラベルを生成するようにします。チャートデータテーブルは以下のように評価されます:

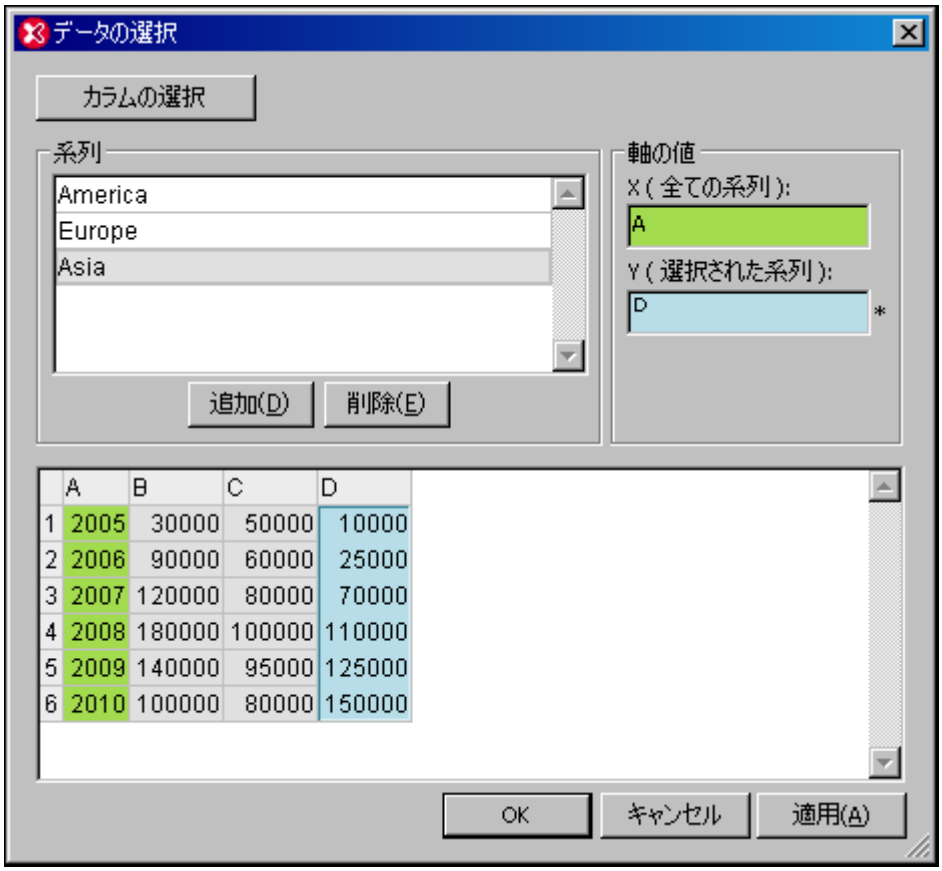
ソース XPath	X-軸	Y-軸（系列列）		
Region[1]/Year[1]	@id	text()	XPath-1	XPath-2
Region[1]/Year[2]	@id	text()	XPath-1	XPath-2
Region[1]/Year[3]	@id	text()	XPath-1	XPath-2
Region[1]/Year[4]	@id	text()	XPath-1	XPath-2
Region[1]/Year[5]	@id	text()	XPath-1	XPath-2
Region[1]/Year[6]	@id	text()	XPath-1	XPath-2

コンテキストノードが、6つある `Region[1]/Year` 要素の各ノードになっていることに注目してください。（スクリーンショットに表示されている）最初のXPath 条件式は、現在の `Year/@id` 属性値を探し、現在の `Region[1]/Year` の `Year@id` と同じ `Year/@id` 値を持つ `Region[2]/Year` 要素を返します。2つめのXPath 条件式は同じことを `Region[3]/Year` 要素に対して行ないます。この方法を使えば、6つある各年に対して、Y-軸の系列は、各 `Region` 要素の `Year` 子要素となります。text() ノードを使うことで、`Region[1]/Year` 要素のコンテンツが返されます。

データの選択ダイアログのチャートデータテーブルは以下のようになります。



データの選択ダイアログの系列の表示を XPath 条件式から、(以下のスクリーンショットにあるような)意味の分かる凡例に変更します。軸の値ペインを使うことで、正しいデータ列を各系列に対して割り当てることができます(Y-軸のテキスト・ボックスをクリックして、チャート・データテーブル内にある目的の列を選択します)。



上で紹介した2つの方法により、全く同じチャートが作成されます。ここでは2つの異なるアプローチを紹介することで、データ選択パラメーターの使用方法について説明しました。

5.11.12 チャートの例 : ローソク足チャート

ローソク足チャートは、通常株式市場における株価の動きを表すために使用されます。ローソク足チャートには2 つの種類があります:

- 4 つの系列を持つローソク足チャート : 1 日の始値、高値、低値、終値が表示されます。
- 3 つの系列を持つローソク足チャート : 1 日の高値、低値、そして終値が表示されます。

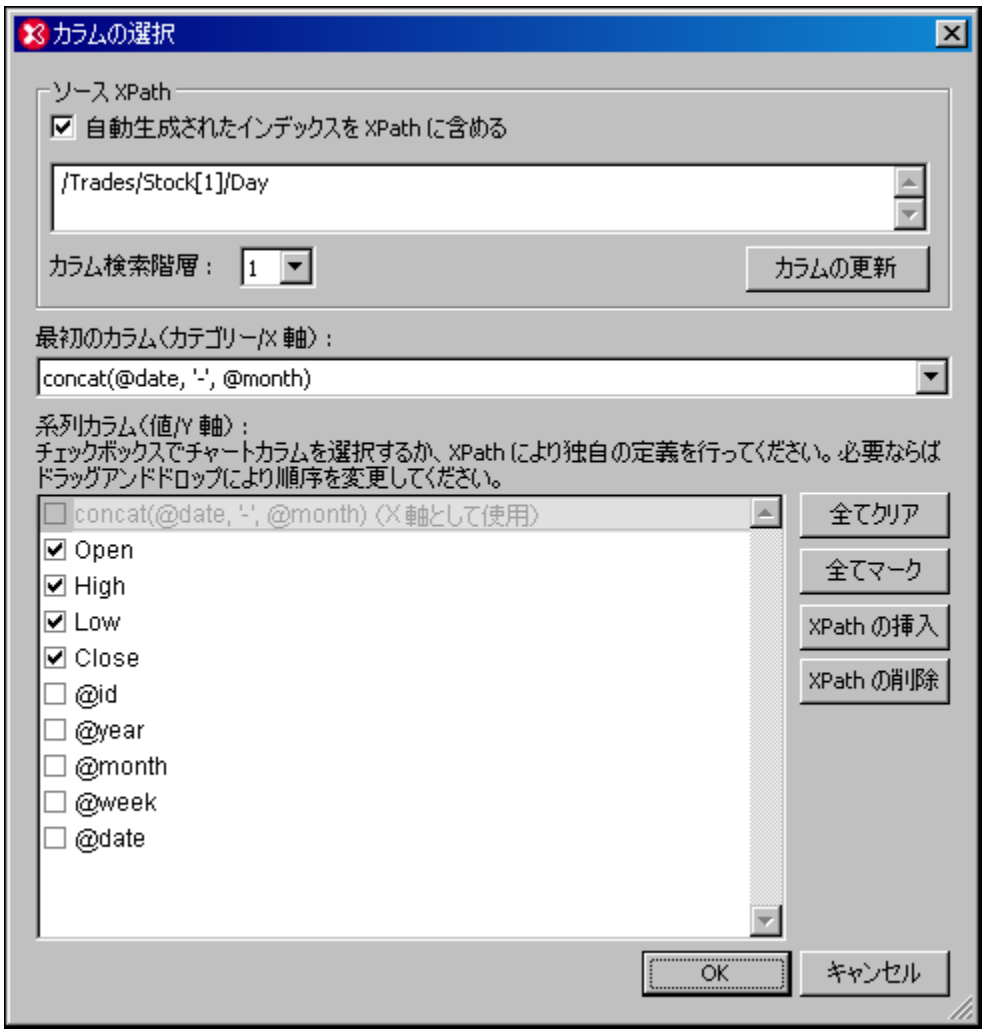
フォルダー C: ¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples¥Tutorial 内にある Candlestick.xml という名前のサンプル XML ドキュメントを使用して、ローソク足チャートを実際に作成してみましょう。以下がファイルの基本的な構造になります。

```
<Trades>
  <Stock name="MyStock">
    <Day id="20110103" year="2011" month="Jan" week="01" date="03">
      <Open>90</Open>
      <High>110</High>
      <Low>88</Low>
      <Close>105</Close>
    </Day>

    .....

  </Stock>
```

```
</Trades>
```

以下の操作を行うことで、このファイルからローソク足チャートを作成することができます:

1. カーソルを Day 要素内部に移動し、チャートウィンドウ[310]の「新規チャート」ボタンを押下します。列の選択ダイアログが表示されます(以下のスクリーンショットを参照)。インデックスを含めるというチェックボックスが無効になっている場合、そのチェックボックスを有効にして、「列の更新」をクリックします。

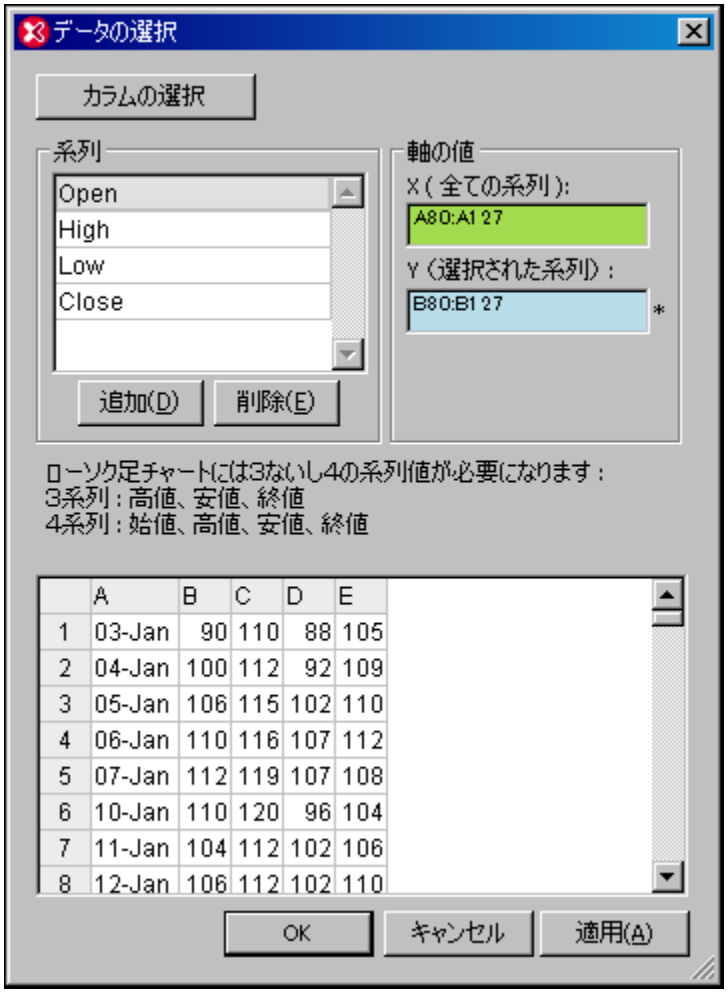


2. XPath の挿入ボタンをクリックして、以下の XPath 条件式を入力します: `concat(@daet, '-', @month)`
3. 最初の列コンボボックスのドロップダウンリストから、先ほど入力した XPath 条件式を選択します。これで各 `Day` 要素の日付と月が X-軸要素のラベルとして使用されます。
4. Y-軸の系列に対して、系列列ペインから Open、High、Low、Close を選択します。
5. 「OK」をクリックします。
6. チャートウィンドウにある「種類の変更」ボタンをクリックし、ローソク足チャートを選択します。
7. 「OK」をクリックします。これで以下のスクリーンショットにあるようなローソク足チャートが作成されます。

データサブセットの選択

列の選択ダイアログにて選択したデータのサブセットのみを使用したい場合（例えば、選択したチャートデータの中から数日分のデータだけをチャートに表示したい場合）、チャートウインドウにある「**データの選択**」ボタンをクリックして、データの選択ダイアログを開きます（以下のスクリーンショットを参照）。

(C) 2015-2021 Altova GmbH



	A	B	C	D	E
1	03-Jan	90	110	88	105
2	04-Jan	100	112	92	109
3	05-Jan	106	115	102	110
4	06-Jan	110	116	107	112
5	07-Jan	112	119	107	108
6	10-Jan	110	120	96	104
7	11-Jan	104	112	102	106
8	12-Jan	106	112	102	110

軸の値ペインにて、X-軸の範囲を入力します。例えば、上のスクリーンショットにあるように、`A80:A127` と入力します。その他の系列では、Y-軸のテキストボックスをクリックした後、系列ペインから目的の系列を選択し、その系列の範囲を入力します。4つある系列に対して同様の操作を行います。例えば、上のスクリーンショットでは、Open 系列における Y-軸の範囲が選択されています。データの選択ダイアログに関するより詳細な情報については、チャートのデータ[329] のセクションを参照ください。

5.12 XML 署名

XML ファイルにはデジタル署名を施して、署名の検証を行うことができます。署名が行われた後にファイルが変更された場合、検証に失敗することになります。XMLSpy ではXML 署名の作成ならびに検証がサポートされます。

XMLSpy ビューにおける XML 署名

XML 署名はXML スキーマやW SDL、XBRL を含む全種類のXML ファイルに対して作成することができます。「XML | XML 署名の作成[372]」ならびに「XML | XML 署名の検証[375]」コマンドはXMLSpy のテキストビュー[130]、グリッドビュー[149]、スキーマビュー[159]、W SDL ビュー[231]、そしてXBRL ビュー[243]にて利用することができます。

XML 署名の動作原理

署名の作成から検証まで、以下のような流れで行われます:

1. 証明書の秘密鍵またはパスワードによりXML ファイルが署名されます。XMLSpy では、「XML | XML 署名の作成[372]」コマンドにより署名を作成することができます。署名は、(i) XML ドキュメントを処理して、(ii) 証明書の秘密鍵またはパスワードを処理することで作成されます。
2. 署名はXML ファイル内部に保管できるほか、独立したファイルとしても保管することができます。
3. XML ファイルの署名は、証明書の公開鍵またはパスワードにより行われます。検証は、(i) XML ドキュメントを処理、(ii) 署名に使用された証明書の公開鍵またはパスワードを処理して、その結果を署名と比較することで行われます。署名が行われた後にXML ファイルが変更されると、検証に失敗することになります。XMLSpy では「XML | XML 署名の検証[375]」コマンドにより署名の検証を行うことができます。

XMLSpy にて署名の作成ならびに検証を行う方法については、このセクション以下にあるサブセクションを参照ください:

- XML 署名の作成[372]
- XML 署名の検証[375]

XML 署名にて証明書を使用する方法

証明書をXML 署名に使用するには、証明書に対応した秘密鍵ならびに公開鍵が必要になります。秘密鍵はXML 署名の作成時に使用され、XML 署名の検証を行う際には公開鍵が必要になります。

通常であれば、XML ドキュメントの送信者が証明書の秘密鍵に対するアクセス権限を持っており、XML 署名が作成されます。ドキュメントの受信者は証明書の公開鍵へアクセスすることになります。公開鍵へアクセスするには2つの方法があります: (i) 送信された署名内にある公開鍵情報へアクセス、(ii) 送信者が使用した証明書の公開鍵へアクセスする。

証明書に関する詳細については、証明書を使用する[377]セクションを参照ください。

メモ: XMLSpy のXML 署名機能では、RSA-SHA1 ならびにDSA-SHA1 およびSHA-256 の証明書のみがサポートされます。パスワードをベースにする署名は、HMAC-SHA256 アルゴリズムを使用します(XMLSpy 2018 以降に関しては、*XML 署名の作成*[372] を参照してください)。

XML ドキュメントの確認

XML ドキュメント内にXML 署名が埋めこまれている場合、`http://www.w3.org/2000/09/xmlsig#` 名前空間にある `Signature` 要素がXML ドキュメントに追加されます。スキーマに従う形でドキュメントの妥当性を保つためには、適切な要素宣言をスキーマに加える必要があります。XMLSpy では2 つの方法で署名を埋め込むことができます:

- *Enveloped* : ルート要素(またはドキュメント要素)最後の子要素として、`Signature` 要素が作成されます。

- *Enveloping* :Signature 要素がルート要素(またはドキュメント要素)として作成され、オリジナルのXMLドキュメント要素が、Object という名前でSignature 要素の子要素として埋めこまれます。

XMLドキュメントのスキーマを修正しない場合、XML 署名を外部ファイルとして作成することができます。詳細については、XML 署名の作成 372 セクションにある配置に関する記述を参照ください。

Enveloped された署名のSignature 要素が許されるようなXML スキーマの抜粋を以下に示します。以下にあるサンプルをガイドとして使用しながら、スキーマの修正を行ってください。

以下にあるうちの最初の例では、XML 署名スキーマがユーザーのスキーマにインポートされます。XML 署名スキーマは http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd というW eb アドレスに配置されています。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:xsig="http://www.w3.org/2000/09/xmldsig#"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
              schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
schema.xsd"/>
   <xs:element name="Root">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="FirstChildOfRoot"/>
            <xs:element ref="SecondChildOfRoot" minOccurs="0"/>
            <xs:element ref="ThirdChildOfRoot" minOccurs="0"/>
            <xs:element ref="xsig:Signature" minOccurs="0"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   ...
</xs:schema>
```

以下にある2つ目のサンプルでは、他の名前空間にある全ての要素にマッチする一般的なワイルドカード要素を追加します。processContents 要素をlax にセットすることで、バリデーターの検証処理でその要素の検証がスキップされます(マッチする要素宣言が見つからないため)。結果として、ユーザーはXML 署名スキーマを参照する必要がなくなります。しかしこの方法を使用すると、(Signature 要素に限らない)全ての要素をXMLドキュメントの特定位置に加えることが可能になってしまいます。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="Root">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="selection"/>
            <xs:element ref="newsitems" minOccurs="0"/>
            <xs:element ref="team" minOccurs="0"/>
            <xs:any namespace="##other" minOccurs="0" processContents="lax"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   ...
</xs:schema>
```

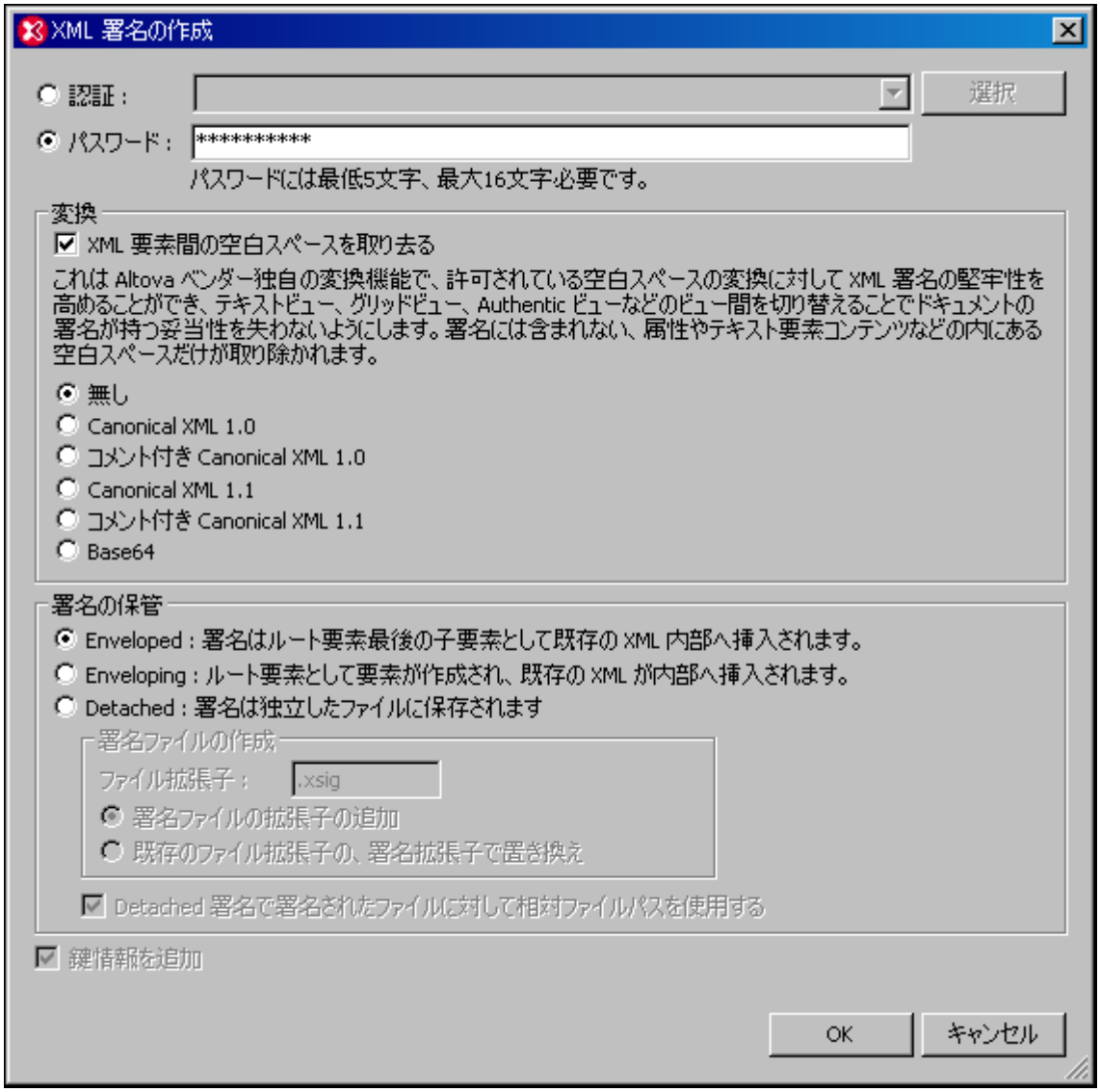
W3C 仕様

XML 署名に関する詳細については、W3C のXML 署名に関する仕様を http://www.w3.org/TR/xmldsig-core/ から参照ください。

5.12.1 XML 署名の作成

XML ドキュメントに対して XML 署名を作成するには、署名を行いたい XML ドキュメントを開きます。その後「XML | XML 署名の作成」コマンドをクリックします。この操作により XML 署名の作成ダイアログが表示されます（以下のスクリーンショットを参照）。

(C) 2015-2021 Altova GmbH



確認方法：認証またはパスワード

署名は証明書またはパスワードを使うことで行うことができます。使用するラジオボタンを選択してください。

- *認証*：選択ボタンをクリックして、目的の証明書を選択してください。選択した証明書には秘密鍵が含まれている必要があります。署名はこの証明書の秘密鍵により生成されます。署名の検証を行うには、その証明書（または、その公開鍵）へアクセスする必要があります。証明書の公開鍵が署名の検証に使用されます。証明書に関する詳細は、証明書を使用する 377 セクションを参照ください。
- *パスワード*：5 文字から 16 文字の長さのパスワードを入力してください。このパスワードは、署名の検証を行う際に必要になります。

メモ XMLSpy 2018 から、パスワードをベースにした署名ドキュメントを使用するアルゴリズムはHMAC-SHA1 からHMAC-SHA256 にアップデートされています。XMLSpy の前のバージョンは、XMLSpy 2018 により生成されるパスワードを使用して検証することができませんでした。

変換

XML データ変換ならびに変換の結果が署名の作成に使用されます。署名の計算を行う前に、正規化アルゴリズムを XML データ (`SignedInfo` コンテンツ) のファイルへ適用することができます。以下にアルゴリズムの重要な特徴を記します:

- *(コメント付き) Canonical XML* : 署名の計算にコメントが含まれる場合、XML 内にあるコメントに対して変更が加えられることで検証が失敗するようになります。署名の計算にコメントを含めない場合、XML ドキュメント内にコメントを加えたり、既にあるコメントを修正しても、署名の検証結果に影響は与えません。
- *Base64* : XML ドキュメントのルート要素 (または ドキュメント要素) が Base64 にエンコードされているものとして、バイナリからの読み込みを試みます。ルート要素が Base64 でエンコードされていない場合、エラーが返されるか、要素が空として認識されます。
- *無し* : 変換を行うことはなく、ディスクに保存されているバイナリファイルから得られた XML データが署名の作成に直接使用されます。署名作成後にデータの編集が行われると、署名の検証が失敗するようになります。*空白スペースを取り去る* チェックボックスが有効になっている場合、全ての空白スペースが取り去られるため、空白スペースに対する変更は無視されます。*無し* オプションと *Canonical* オプションの違いは、正規化により XML データストリームが生成され、属性の順序などが正規化されるというものです。結果として正規化による変換を行う際には属性の順序といった変更が正規化され (つまり検証には成功し)、変換を伴わない場合、それらの変更が反映されることになります (つまり検証は失敗します)。署名が (Enveloped または Enveloping により) 埋め込まれている場合、デフォルトで正規化が行われます。署名が Detached で作成された場合、*無し* が選択された場合、または *空白スペースを取り去る* チェックボックスにチェックが入っていない場合、XML データはそのままのかたちで (つまり変換無しで) 使用されます。

証明の保管

署名は XML ファイル内部に配置することができるほか、独立したファイルとしても作成することができます。以下のオプションが利用可能です:

- *Enveloped* : 署名要素がルート要素 (ドキュメント要素) 最後の子要素として作成されます。
- *Enveloping* : 署名要素がルート要素 (ドキュメント要素) として作成され、XML ドキュメントが子要素として挿入されます。
- *Detached* : XML 署名が独立したファイルとして作成されます。この場合、署名ファイルのファイル拡張子を指定することができるほか、(i) XML ファイルに拡張子を追加するかたちで (例: `test.xml.xsig`)、または (ii) XML ファイルの XML 拡張子を置き換えることで (例: `test.xsig`) ファイル拡張子の形式を指定することができます。署名ファイルにおける XML ファイルへの *リファレンス* を相対または絶対パスで行うか指定することもできます。

メモ: XML スキーマ (`.xsd`) ファイルに対しては、外部署名ファイルとしてしか XML 署名を作成することはできません (埋め込むことはできません)。XBRL ファイルに対する XML 署名は、外部署名として XBRL ビューから作成することができます。WSDL ファイルの場合、署名ファイルを外部ファイルとして作成するか、Enveloped として WSDL ファイル内に含めることができます。

メモ: XML 署名が Detached (独立したファイル) として作成された場合、署名ファイルにある *リファレンス* により XML ファイルと署名ファイルの関連付けが行われます。従って署名の検証を行う際には、署名ファイルがアクティブになっている状態で行う必要があります (XML ファイルがアクティブになっている状態で行うことはできません)。

鍵情報の追加

鍵情報を追加 オプションは、証明書をベースに署名を作成する際に有効になるオプションで、パスワードによる署名では無効になっています。

このオプションが有効になっていると、公開鍵情報が署名内部に配置されます。オプションが選択されていない場合、鍵情報が署名に加えられることはありません。鍵情報を追加することで、検証を行う際に、証明書そのもの (証明書内部にある公開鍵情報) が必要なくなります (公開鍵情報が既に署名に含まれているため)。

5.12.2 XML 署名の検証

XML 署名の作成後にXML ファイルが変更されなかった場合に限りXML 署名の検証は成功し、そうでない場合、検証は失敗します。XMLSpy では以下に記される条件が満たされる場合、XML 署名の検証を行うことができます:

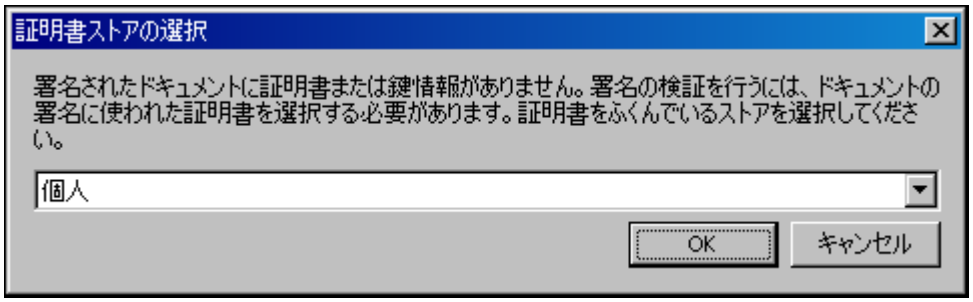
- XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれている375
- XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれていない375
- 証明書をベースにした署名は外部ファイルとして保存されており、鍵情報が署名に含まれている376
- 証明書をベースにした署名は外部ファイルとして保存されており、鍵情報が署名に含まれていない376
- パスワードをベースにした署名がXML ファイルに含まれている376
- パスワードをベースにした署名が独立したファイルに存在する377

XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれている

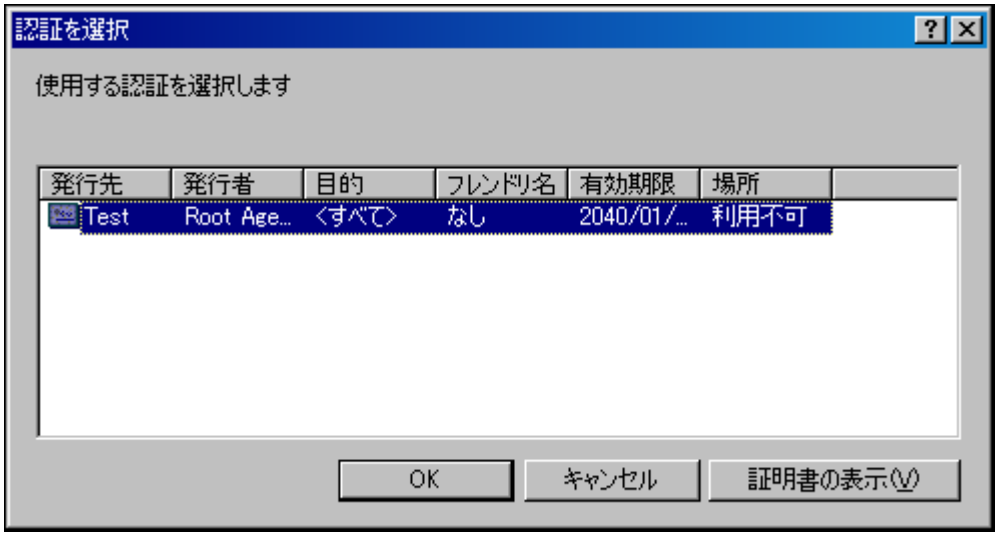
XML 署名の検証を行うには、XMLSpy にてXML ファイルをアクティブにします。「XML | XML 署名の検証」コマンドをクリックすると、検証処理が実行され、(検証が成功または失敗したという)結果がメッセージウィンドウに表示されます。

XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれていない

証明書をベースにした署名に鍵情報が含まれていない場合、XMLSpy により検証に使用する公開鍵が収められている証明書を指定するよう促されます。XMLSpy にてXML ファイルがアクティブになっている状態で検証が実行されます。「XML | XML 署名の検証」コマンドをクリックすると、証明書が収められている証明書ストア377を選択するよう促されます(以下のスクリーンショットを参照)



証明書ストア377を選択し、「OK」ボタンをクリックすると、そのストア内にある証明書を表示するダイアログが表示されます(以下のスクリーンショットを参照)。検証に必要な証明書を選択し、「OK」をクリックします。



検証処理が実行され、結果がメッセージウィンドウに表示されます。

証明書をベースにした署名は外部ファイルとして保存されており、鍵情報が署名に含まれている

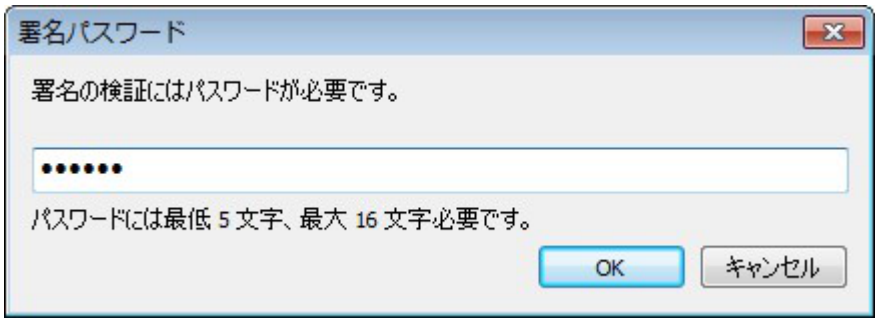
証明書をベースにしたXML 署名が独立したファイルとして存在する場合、署名ファイルがXMLSpy にてアクティブになっている状態で署名の検証が行われます。「XML | XML 署名の検証」コマンドをクリックすると、検証処理が実行され、その結果(検証が成功したか失敗したか)がメッセージウィンドウにて表示されます。

証明書をベースにした署名は外部ファイルとして保存されており、鍵情報が署名に含まれていない

証明書をベースにしたXML 署名が独立したファイルとして存在する場合、署名ファイルがXMLSpy にてアクティブになっている状態で署名の検証が行われます。「XML | XML 署名の検証」コマンドをクリックすると、検証に必要な公開鍵情報が得られる証明書を指定するようXMLSpy により促されます。XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれていない[375]の記述に従い、証明書を選択してください。

パスワードをベースにした署名がXML ファイルに含まれている

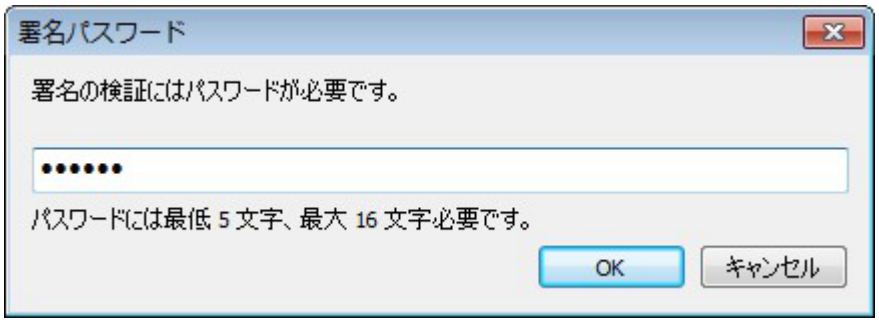
XML ファイル内にパスワードをベースにしたXML 署名が含まれている場合、XMLSpy ではそのXML ファイルがアクティブになっている状態で検証を行うことができます。「XML | XML 署名の検証」コマンドをクリックすると、パスワードを入力するためのダイアログが表示されます(以下のスクリーンショットを参照)。



5文字から16文字のパスワードを入力し、「OK」をクリックします。検証処理が実行され、その結果(検証が成功したか失敗したか)がメッセージウィンドウにて表示されます。

パスワードをベースにした署名が独立したファイルに存在する

独立したファイルにパスワードをベースにした署名が収められている場合、XMLSpy では署名ファイルがアクティブになっている状態で検証を行うことができます。「XML | XML 署名の検証」コマンドをクリックすると、パスワードを入力するためのダイアログが表示されます(以下のスクリーンショットを参照)。



5文字から16文字のパスワードを入力し、「OK」をクリックします。検証処理が実行され、その結果(検証が成功したか失敗したか)がメッセージウィンドウにて表示されます。

5.12.3 証明書を使用する

証明書はXML 署名の作成や検証に広く使用されています。このセクションでは、証明書の入手、インポート、エクスポート方法について記述します。以下のような構成となります:

- 秘密鍵-公開鍵のペアとともに証明書を入手する 377
- 秘密鍵-公開鍵証明書をインポートする 377
- Windows マシン上の証明書ストア 378
- 公開鍵証明書をエクスポートする 379

秘密鍵-公開鍵のペアとともに証明書を入手する

以下の方法により証明書を入手することができます:

- **認証局(CA)**:認証局により証明書の所有者のアイデンティティーが確認されます。この方法により入手した証明書は、誰にでも作成することができる独自の証明書と異なるものです。
- **独自の証明書を作成**:認証局により確認されていない証明書ですが、それなりの安全性も提供されます。Microsoft Visual Studio といったツールを使用することで、証明書を作成することができます。

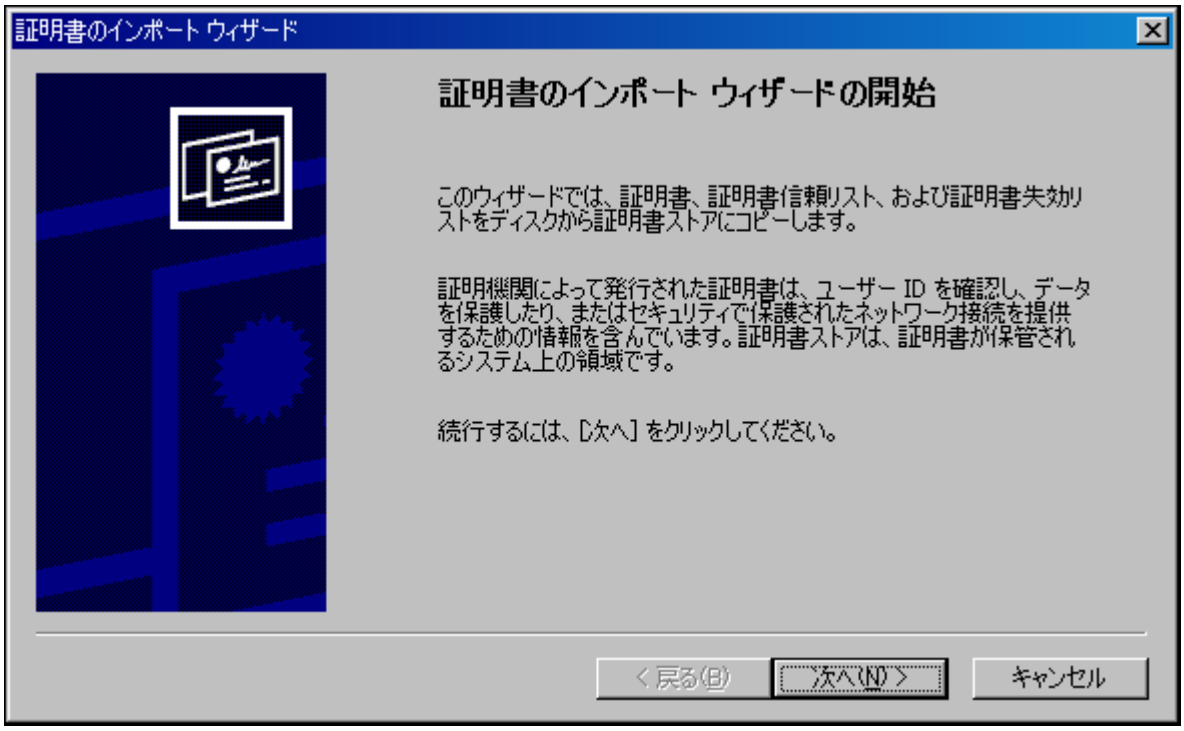
XML 署名を行うには、秘密鍵と公開鍵ペアを含む証明書を作成する必要があります。

メモ: XMLSpy のXML 署名機能では、RSA-SHA1、DSA-SHA1 および SHA-256 の証明書がサポートされます。

秘密鍵-公開鍵証明書をインポートする

秘密鍵と公開鍵を含む証明書を入手できたら、Windows 証明書ストアにインポートする必要があります。以下の操作を行ってください:

1. 証明書ファイルをダブルクリックして、証明書のインポートウィザードを起動します(以下のスクリーンショットを参照)。「次へ」をクリックします。



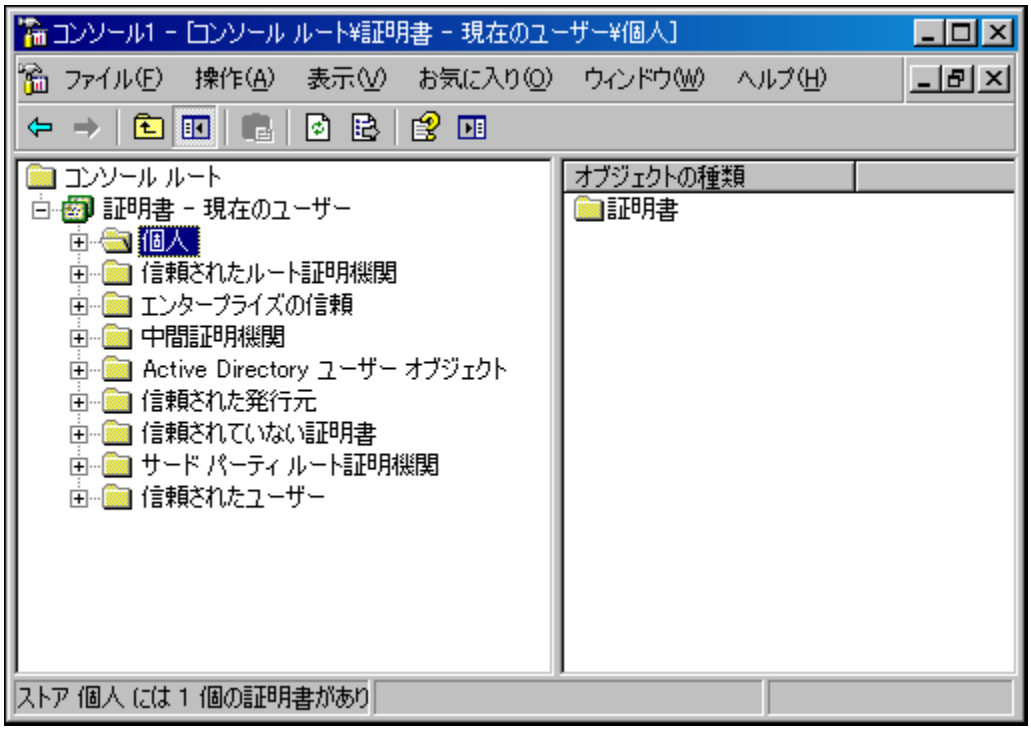
2. インポートする証明書ファイルにて目的の証明書ファイルが選択されていることを確認して、「次へ」をクリックします。
3. 秘密鍵のパスワードを入力します。XML 署名の作成に秘密鍵を使用するには、パスワードを知っている必要があります。秘密鍵のパスワードは、証明書を作成した際に入力したものを使用してください。パスワードを入力した後に、「次へ」をクリックしてください。
4. 証明書の種類により、証明書が配置される証明書ストアを自動的に選択させることができるほか、ストアを自分で選択することもできます(証明書が配置されているストアが分かっているという点で、自分でストアを選択したほうが良いかも知れません)。完了後に「次へ」をクリックしてください。
5. 「完了」をクリックして、処理を完了します。

Windows マシン上の証明書ストア

以下の操作によりWindows XP マシン上にある証明書ストアへアクセスすることができます:

1. スタートメニューから「ファイル名を指定して実行」を選択します。
2. mmc と入力して、「OK」をクリックします。コンソールウインドウが表示されます(以下のスクリーンショットを参照)

(C) 2015-2021 Altova GmbH



3. コンソールウィンドウにて、「**ファイル | スナップインの追加と削除**」コマンドを選択します。
4. スナップインの追加と削除ダイアログのスタンドアロンタブにて、「**追加**」ボタンをクリックします。
5. スタンドアロンスナップインの追加ダイアログが表示されるので、証明書を選択して「**追加**」をクリックします。
6. スタンドアロンスナップインの追加ダイアログを閉じます。
7. スナップインの追加と削除ダイアログにて「OK」をクリックします。
8. コンソールウィンドウのコンソールルートに証明書アイテムが追加されます(上のスクリーンショットを参照)。証明書アイテムの中にはお使いのコンピューター内にある証明書ストアが含まれます。
9. コンソールウィンドウの「**ファイル | 保存**」コマンドにより、Microsoft Management Console ファイル(.msc ファイル)が保存されます。このMSC ファイルを使用することで、(コンソールウィンドウの「**ファイル | 開く**」コマンドから)お使いのコンピューターにある証明書ストアへアクセスすることができます。

公開鍵証明書をエクスポートする

秘密鍵付きの証明書を使用している場合、公開鍵だけを持つ証明書をエクスポートしたほうが良い場合があります。作成された公開鍵付きの証明書をXML 署名受信者へ渡すことで、秘密鍵の証明書により作成された署名の検証に使用することができます。

以下の操作を行うことにより秘密鍵と公開鍵の証明書から公開鍵の証明書をエクスポートすることができます。

1. コンソールウィンドウから証明書ストアを開きます。(i) **スタート**メニューの「**ファイル名を指定して実行**」コマンドにて mmc と入力し、(ii) 表示されるコンソールウィンドウにて「**ファイル | 開く**」を選択し、証明書ストアが保存されたMSC ファイルを選択します(このセクション直前のセクションを参照ください)。
2. 公開鍵証明書としてエクスポートする証明書を選択し、右クリックします。
3. コンテキストメニューから、「**すべてのタスク | エクスポート**」を選択します。これで証明書のエクスポートウィザードが表示されます(以下のスクリーンショットを参照)。

4. 「次へ」を選択します。
5. 秘密鍵のエクスポートウィンドウにて、*秘密鍵をエクスポートしない*を選択し、「次へ」をクリックします。
6. エクスポートファイルの形式ウィンドウにて、目的のフォーマットを選択し、「次へ」をクリックします(分からない場合は、デフォルトの DER フォーマットのままにしてください)。
7. エクスポートするファイルウィンドウにて、ファイルを保存する場所とファイルの名前を指定します(ファイルの拡張子は、自動的に追加されます)。「次へ」をクリックしてください。
8. 「完了」をクリックしてエクスポートを完了します。

公開鍵を含む証明書が指定された場所にて作成されます。対応する秘密鍵の証明書により署名された XML ファイルを受け取った相手に、公開鍵の証明書を送ることができます。証明書を受信した相手は、公開鍵の証明書を証明書ストアにインポートすることで、証明書の公開鍵を検証に使用することができます。

5.13 その他の機能

今までご紹介した以外にも、XML ファイルを使用するうえで以下の機能を利用することができます。

- エンコーディング 381
- DTD やスキーマの生成 381
- 検索と置換 381
- XPath の評価 382
- テキストのインポートとエクスポート 382

エンコーディング

XML ファイル(そしてその他のドキュメント)のエンコーディングはメニューコマンド「ファイル | エンコード」1137 により設定することができます。XML ならびに非 XML ファイルのデフォルトエンコーディングは「オプション | エンコード」1460 セクションにより指定することができます。

DTD やXML スキーマの生成

「DTD/スキーマ | DTD/スキーマの生成」1245 メニューコマンドにより、XML ドキュメントの構造を記述するスキーマを作成することができます。表示される DTD スキーマの生成ダイアログでは、生成するスキーマの種類(DTD もしくは XML スキーマ)や、XML ドキュメントにある値から列挙を生成するなどの XML スキーマオプションを指定することができます。

検索と置換

「編集」メニューからアクセスすることができる検索 1159 ならびに置き換え 1162 機能により、パワフルな検索機能を手に入れることができます。検索の対象となる単語には、大文字や小文字の区別をする、単語の完全マッチ、更には正規表現を使用することもできます。検索の対象はドキュメント内の選択や(以下のスクリーンショットに示されるように)特定のノード型に限定することができます。

上のスクリーンショットはグリッドビュー内に検索ダイアログを表示しています。。ダイアログとその機能は、アクティブになっているビューにより変化します。テキストビューの検索機能については、ここ[140]を参照してください。

XPath の評価

XPath ウィンドウで入力した XPath 条件式をアクティブな XML ドキュメントに対して評価することができます。評価結果は XPath ウィンドウ内に表示され、その中のノードをクリックすることで、メインウィンドウ内の対応するノードがハイライトされます。XPath ウィンドウは「XML | XPath を評価」コマンドによりアクティブにすることができます。

テキストのインポートとエクスポート

テキストデータは他のアプリケーションから(そして他のアプリケーションへ)インポート(ならびにエクスポート)することができます。これらの機能のコマンドは変換[1331]メニューから利用することができます。

(C) 2015-2021 Altova GmbH

6 DTD と XML スキーマ

Altova Web サイト: XML スキーマエディター

このセクションではDTD[384] ならびにXML スキーマ[386]に関する概要を説明します。SchemaAgent[397] ならびにパワフルなスキーマの内を検索[415]する機能についても解説されます。編集機能の他に、XMLSpy ではDTD/スキーマに関する以下の機能が提供されます:

カタログメカニズム

OASIS カタログ機能[409]によりURI からローカルアドレスのリダイレクトが可能になり、複数のワークステーションにおける利用が促進されます。

スキーマルール

ユーザー定義の追加制約をXML スキーマに対して割り当てることができます。XMLSpy にはスキーマルールエディターが搭載されており、XML スキーマに対して、スキーマルールセットを作成、編集することができます。

スキーマサブセット

スキーマビューでは、巨大なスキーマのコンポーネントを、別のファイルとして作成することができます。これら(より小さな)スキーマサブセットを、大きなスキーマにインクルードすることもできます。また、スキーマのフラット化という名前の機能を使うことで、逆の操作を行い、インクルードされているファイル内にあるコンポーネントを、直接巨大なスキーマに取り込むことができます。スキーマのサブセットと、スキーマのフラット化に関する詳細な情報は、スキーマサブセット[387]のセクションを参照ください。

DTD とXML スキーマ間の変換

「DTD/スキーマ[1243]」メニューを使用することで、DTD とXML スキーマ間の変換を行うことができます。使用方法などDTD がフラット化されている場合、含まれているインポートされたモジュールは親ファイルに直接保存され、使用されていないコンポーネントは削除されます。

サンプルXML ファイルの生成

「DTD/スキーマ | サンプルXML ファイルの生成」[1251] メニューコマンドにより、アクティブなDTD やXML スキーマファイルをベースにしたスケルトンXML ドキュメントの生成を行うことができます。スキーマをベースにしたXML ファイルを作成するにはとても便利な機能です。

定義へ移動

カーソルがXML ドキュメントのノード内に位置している場合、「DTD/スキーマ | 定義へ移動[1245]」メニューコマンドにより、スキーマファイルが開かれ、選択されているXML ノードの定義を行っている箇所がハイライトされます。

6.1 DTD

テキストビューとグリッドビューでDTD ドキュメントを編集することが可能です。デフォルトのビューはオプションダイアログのファイル型セクション[1458] で設定することができます。

テキストビュー

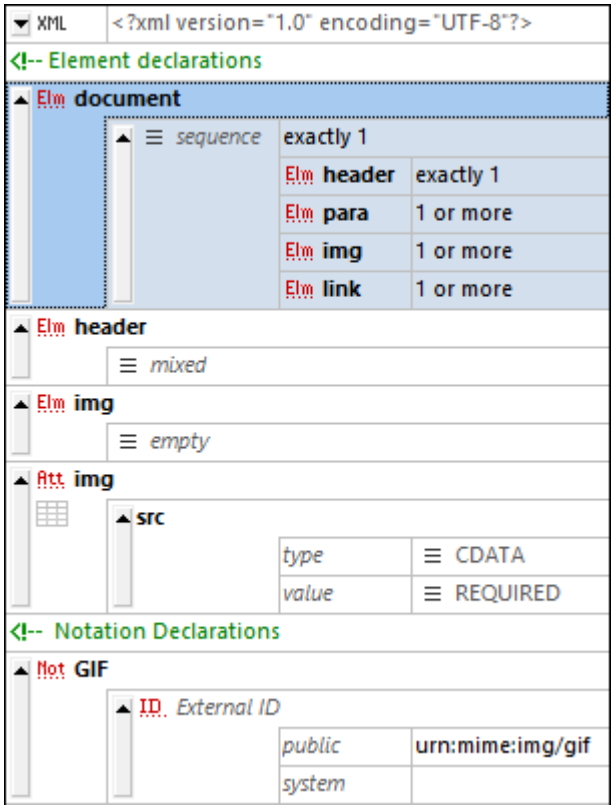
テキストビューでは、ドキュメントは構文により色分けされています。以下はDTD の断片サンプルです。:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Element declarations-->
<!ELEMENT document (header, para+, img+, link+)>
<!ELEMENT header (#PCDATA)>
<!ELEMENT img EMPTY>
  <!ATTLIST img
    src CDATA #REQUIRED
  >

<!-- Notation Declarations -->
<!NOTATION GIF PUBLIC "urn:mime:img/gif">
```

インデントはインデントガイドにより表され、タブ キーにより入力することができます。インデントの大きさはテキストビュー設定ダイアログ[132] にてセットすることができます。

グリッドビュー

グリッドビューにおいて、DTD ドキュメントはテーブルとして表示されます。以下のスクリーンショットでは、上にあるDTD をグリッドビューで表示したものです。



DTD 構造の編集

- コンテキストメニューまたは **XML** メニューを使用してカーソルがセル内にある場合、ノードを挿入または追加、または子ノードを追加することができます。
- セルの左上のノードの型アイコンをクリックしてノードの型を変更します。
- (*シーケンス*、*複合*、*空* などの) コンテンツモデルとノードの発生修飾子 (*厳密に 1*、*1 またはそれ以上* など) を対応するアイコンとクリックして、オプションを選択して変更することができます。
- ドラッグアンドドロップを使用してノードをドキュメント内の新規の場所に移動、および新規の場所にコピーして貼り付けることができます。

DTD 値の編集

- 要素、属性名、およびコメントなどの値を編集する場合セルをダブルクリックして編集します。

グリッドビューツールバー
グリッドビューツールバーではビューの設定ダイアログへのアクセスを提供され、また、倍率の設定やワードラップなどのコマンドが含まれています。

XMLSpy 内の DTD 機能

XMLSpy には以下の非常に便利な機能が含まれています::

- *DTD を XML スキーマへ変換*: 「DTD/スキーマ | DTD/スキーマの変換」[1247] コマンドにより、DTD を XML スキーマへ変換することができます。
- *DTD からサンプル XML ファイルを生成*「DTD/スキーマ | サンプル XML ファイルの生成」[1251] コマンドにより、アクティブな DTD をベースにした XML ドキュメントを生成することができます。

6.2 XML スキーマ

XML スキーマドキュメントの編集はテキストビュー、グリッドビュー、そしてスキーマビューで行うことができます。XML スキーマドキュメントが開かれるデフォルトのビューは、オプションダイアログのファイルの型タブ[1458]にて設定することができます。ドキュメントの編集中でも、その時に一番使いやすいビューに切り替えて作業を行うことができます。XML スキーマドキュメントは、大抵の場合 .xsd または .xs のファイル拡張子で保存されます。

テキストビューでの編集

テキストビューの XML スキーマ編集は XML ドキュメントと同じように行われ、XML ドキュメントで使用可能な編集機能[266]を XML スキーマでも利用することができます。XML ドキュメントと同様に、参照しているスキーマを特定し、アクセスすることができる場合、カーソルの位置に追加することができるアイテムがテキストビュー入力ヘルパー[145]に表示されます。

グリッドビューでの編集

グリッドビューの XML スキーマ編集は XML ドキュメントと同じように行われ、XML ドキュメントで使用可能な編集機能[149]を XML スキーマでも利用することができます。グリッドビューにてアイテムが選択されている場合、カーソルの位置に追加することのできるアイテムがグリッドビュー入力ヘルパー[155]に表示されます。

スキーマビューでの編集

スキーマビューはスキーマのデザインをグラフィカルに行うためのビューです。スキーマビューにてスキーマの作成、編集を行うと、対応するテキストドキュメントが XMLSpy により生成されます。スキーマビューにて XML スキーマドキュメントを作成、編集する詳細な方法については編集ビュー | スキーマビュー[159]を参照ください。

XMLSpy での XML スキーマ機能

XMLSpy には更に、以下のような便利な機能が含まれます:

- *XML スキーマを DTD へ変換* : 「DTD/スキーマ | DTD/スキーマの変換」[1249] コマンドにより、XML スキーマを DTD へ変換することができます。
- *スキーマからサンプル XML ファイルを生成* : 「DTD/スキーマ | サンプル XML ファイルの生成[1251]」コマンドにより、アクティブな XML スキーマをベースにした XML ドキュメントを生成することができます。生成されるサンプル XML の要素ならびに属性のサンプル値も指定することができます。
- スキーマビューにて表示される XML スキーマ (.xsd) に対する XML 署名[370]は、独立した署名ファイルとして作成することができます。署名の使用方法については XML 署名[370]のセクションを参照ください。

6.3 スキーマサブセット

XML スキーマのコンポーネントは、スキーマサブセットと呼ばれる、(より小さな)外部スキーマファイルとして作成することができます。小さなスキーマサブセットの集合を(インクルードと一緒に)使用することで、巨大なスキーマを使用した際に比べて、単一の巨大なスキーマよりもスキーマの管理を行ないやすくなります。

スキーマビューにて以下のような操作を行うことで、スキーマサブセット機能が持つ機能を理解することができるはずです:

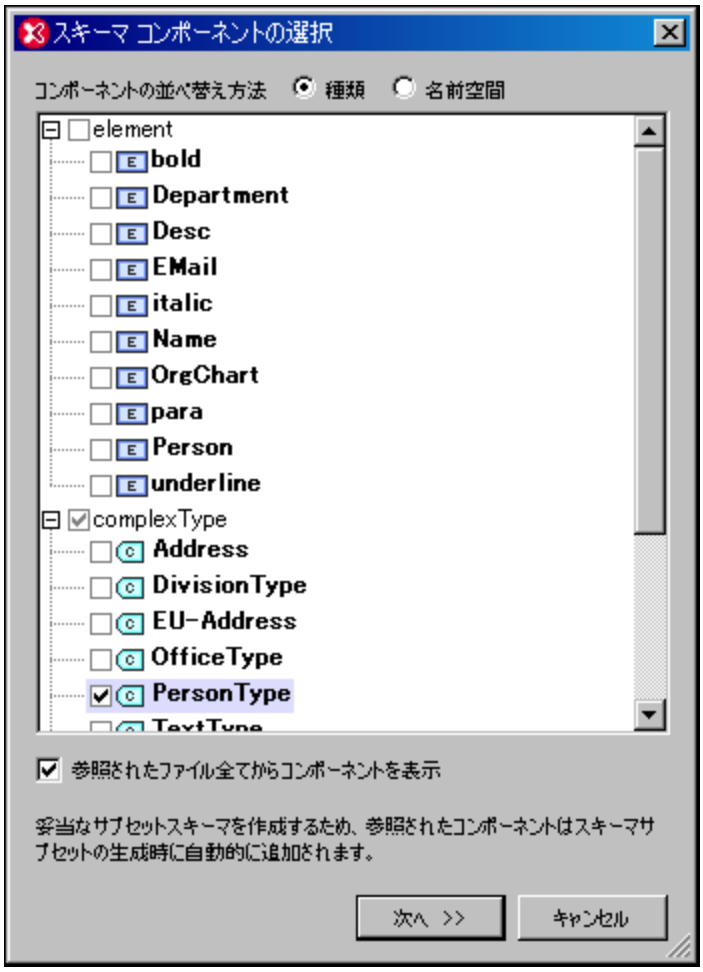
1. アクティブなスキーマのコンポーネントを含むスキーマサブセットを作成します。具体的な操作の方法については以下を参照ください 387。
2. 必要に応じて別のスキーマサブセットを作成します。
3. 新たに作成されたスキーマサブセットをインクルードして、大きなスキーマを形成します。スキーマ概要ウィンドウ 164にて、インクルードコンポーネントを追加または挿入し、新たに作成したスキーマサブセットファイルを選択する操作を、各スキーマサブセットに対して行ないます。
4. オリジナルの巨大なスキーマに元々存在し、インクルードされたサブセットにも存在するコンポーネントを削除します。

スキーマビューでは、上の操作とお逆に、インクルードされたファイルをフラット化することができます。つまり、(i) スキーマサブセットに含まれているコンポーネントがメインスキーマに直接追加し、(ii) インクルードされたスキーマサブセットをメインスキーマから削除します。スキーマをフラット化する方法については以下に記述されます 389。

スキーマサブセットの作成

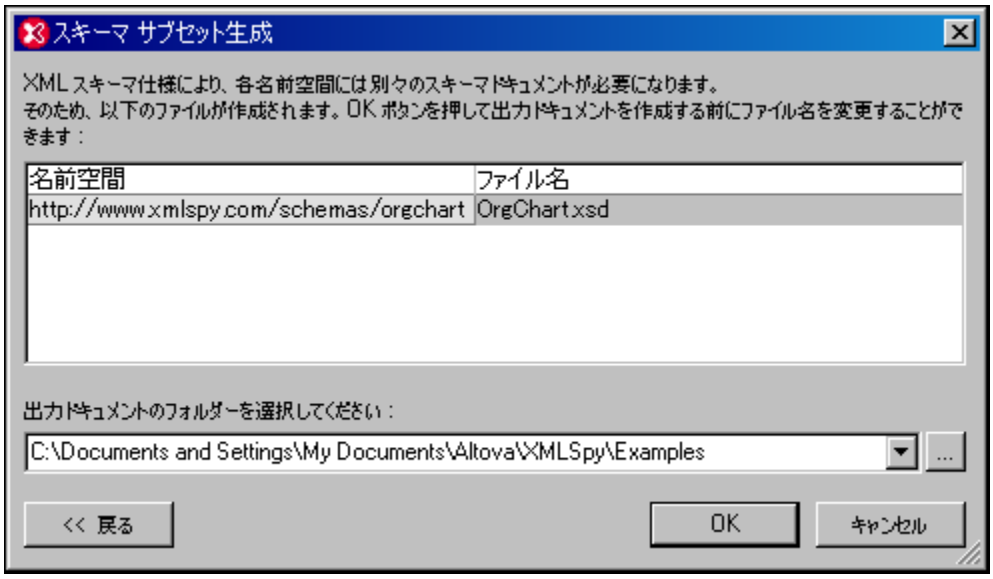
スキーマサブセットを作成するには:

1. 目的のXML スキーマがスキーマビューにてアクティブな状態で、メニューコマンドの「**スキーマ設計 | スキーマサブセットの作成**」を選択します。この操作により、スキーマコンポーネントの選択ダイアログが表示されます(以下のスクリーンショットを参照)。
2. ダイアログにて、単一のスキーマサブセットとして作成したいコンポーネントにチェックを入れ、「**次へ**」をクリックします。ペイン以下のチェックボックスにより、参照されている全てのファイルにあるコンポーネントを表示することができます。



3. （以下のスクリーンショットにある）スキーマサブセット生成ダイアログにて、スキーマサブセットパッケージの名前を入力します。同時に、新たなスキーマサブセットファイルが保存されるフォルダーも指定する必要があります。オリジナルのスキーマにてインポートされていたコンポーネントが、作成されるコンポーネントに含まれている場合、スキーマサブセットパッケージには複数のファイルが含まれることになります。スキーマサブセットでは、各名前空間に対して、異なるファイルが作成されます。デフォルトでダイアログに表示されるファイル名はオリジナルのファイル名となりますが、オリジナルのファイルを上書きすることは許されないため、オリジナルのファイルがあるフォルダーにサブセットファイルを保存したい場合は、名前の変更を行う必要があります。

(C) 2015-2021 Altova GmbH

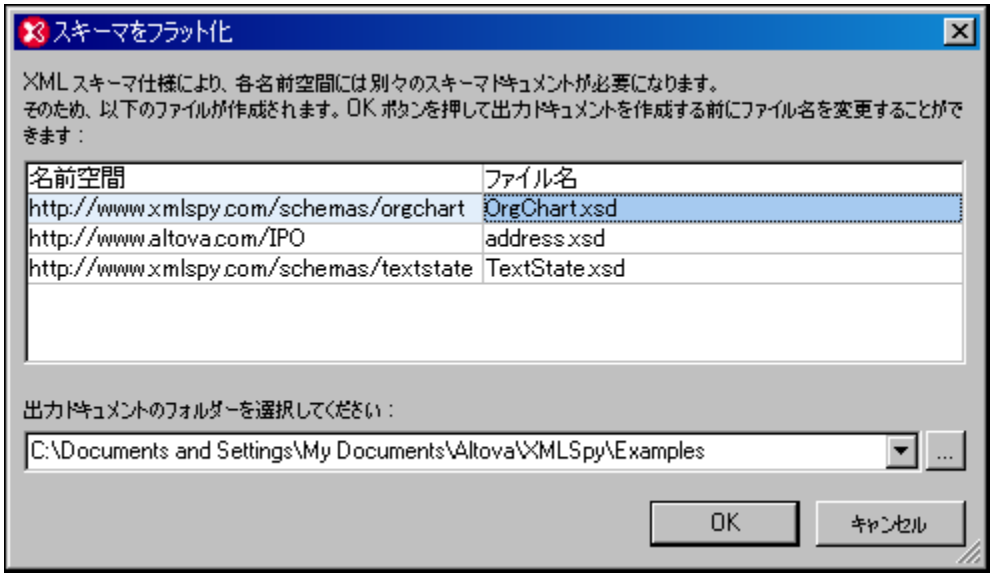


4. 「OK」をクリックすることで、アクティブなファイルの名前空間を持つスキーマサブセットファイルが、スキーマビューで開かれます。パッケージ内にあるその他のファイルは、作成されますが、スキーマビューでは開かれません。

スキーマのフラット化

スキーマビューにてアクティブなスキーマをフラット化すると、(i) 各スキーマのコンポーネントがアクティブなスキーマのグローバルコンポーネントとして追加され、(ii) インクルードされたスキーマが削除されます。

アクティブなスキーマをフラット化するには、メニューコマンドから「**スキーマ設計 | スキーマをフラット化**」を選択します。この操作により、(以下のスクリーンショットにある) スキーマをフラット化ダイアログが表示され、各ファイルの名前が、フラット化されたスキーマで使用される名前空間と共に表示されます。デフォルトのファイル名はオリジナルのファイル名のままになっていますが、オリジナルのファイルの場所と同じフォルダーに保存しようとする場合には、ファイル名を変更する必要があります。フラット化されたスキーマと、関連するファイルを保存するフォルダーを指定することもできます。



「OK」をクリックすると、フラット化されたスキーマが、スキーマビューで開かれます。

(C) 2015-2021 Altova GmbH

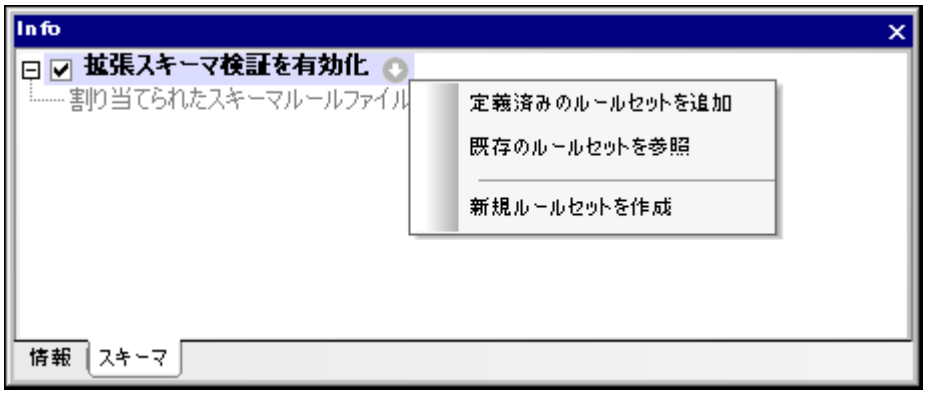
6.4 スキーマルール

スキーマルールセットとはXML スキーマの検証を行う際に使用することのできるルールセットのことです。例えばルールを使うことで、(XML スキーマで定義されている様に) 属性名を小文字で始まる様に指定し、特定の型からしか複合型を作成できないように指定することができます。

スキーマルールのセットはXML (.xml) ファイルであるルールセットファイル内に保存されます。XMLSpy にはスキーマルールエディター[392]が搭載されており、グラフィカルにスキーマルールを編集することができます。XMLSpy では複数のルールセットファイルを XML スキーマに割り当てる[391]ことができます。XML スキーマビューにて「XML | XML 検証(F8)」コマンドを使用すると、割り当てられたルールセットに対して XML スキーマが検証されます。

6.4.1 ルールセットの管理

アクティブな XML スキーマ(.xsd ファイル)には1つ以上のスキーマルールセットファイル(.xml ファイル)を割り当てることができます。この操作は情報ウィンドウのスキーマタブから行うことができます(以下のスクリーンショットを参照)。



ルールセットによる拡張検証

スキーマルールセットを追加するにはコンテキストメニューボタン をクリックします。(上のスクリーンショットで表示される)ポップアップメニューからルールセットファイルを XML スキーマに追加することができます。以下のオプションが利用可能です:

- **定義済みのルールセットを追加** : XMLSpy と共に提供される定義済みのスキーマルールセットのリストから選択を行うことができます。これらのルールセットファイルはXMLSpy アプリケーションフォルダー以下の Extended Schema Validation フォルダーに保存されます。このフォルダーに新たなルールセットファイルを追加することで、そのファイルをルールセットの追加時に定義済みルールセットダイアログから選択することができるようになります。
- **既存のルールセットを参照** : 定義済みリストにはないスキーマルールセットファイルを参照することができます。
- **新規ルールセットを作成** : スキーマルールエディターを起動して、スキーマルールセットファイル内のスキーマルールを編集することができます。スキーマルールエディターの使用方法については、ルールセットの定義[392]を参照ください。

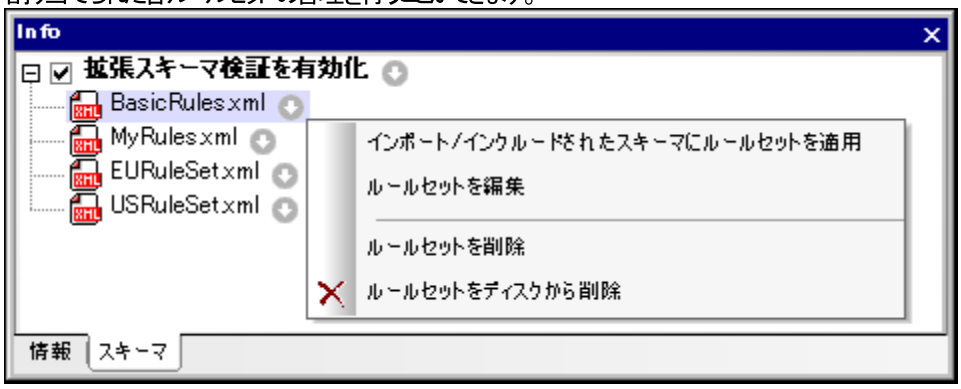
(上のスクリーンショットにあるように) 追加できるスキーマルールセットの数に上限はありません。2つ以上のスキーマルールセットがXML スキーマに割り当てられている場合、全てのスキーマルールセットから得られたルールがXML スキーマの検証時(「XML | XML 検証」)に使用されます。

拡張スキーマ検証の有効/無効化

拡張スキーマ検証を有効化チェックボックスをクリックすることで、拡張スキーマ検証を有効化/無効化することができます。

ルールセットの編集と削除

(以下のスクリーンショットにあるように) コンテキストメニューボタン をクリックすることで表示されるコンテキストメニューから、XML スキーマに割り当てられた各ルールセットの管理を行うことができます。



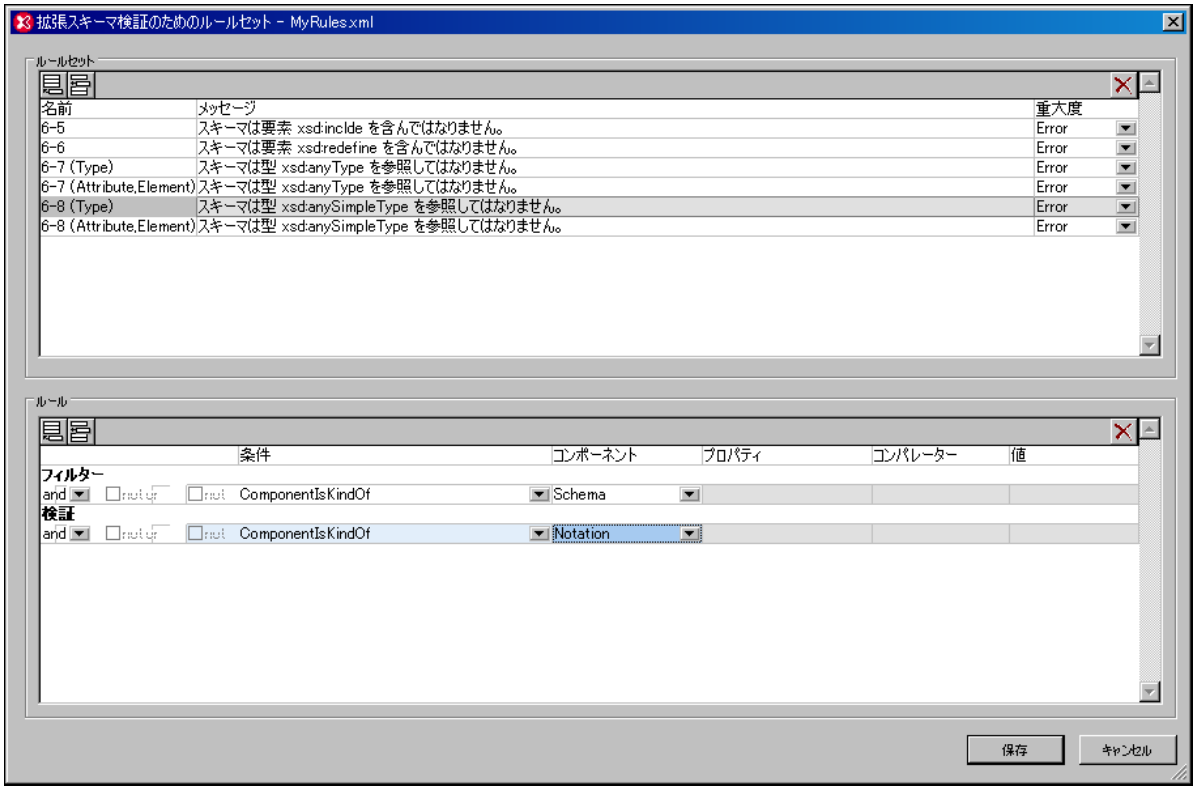
以下のオプションが利用可能です:

- **インポート/インクルードされたスキーマにルールセットを適用** : メインスキーマがインポートまたはインクルードする全てのスキーマに対してルールセットが適用されます。
- **ルールセットを編集** : スキーマルールセットをスキーマルールエディターで開きます。
- **ルールセットを削除** : ルールセットのリストからルールセットを削除します。
- **ルールセットをディスクから削除** : このコマンドは定義済みのリストにはないルールセットに対して有効になります。ルールセットのリストからルールセットを削除するだけでなく、ルールセットのファイルそのものを削除します。

6.4.2 ルールセットの定義

スキーマルールセットは(以下のスクリーンショットにあるように) スキーマルールエディターにて開き、編集することができます。スキーマルールセットファイル内にてスキーマルールを作成、編集、削除することができます。スキーマルールエディターでルールセットを開くには以下の操作を行ってください:

1. 情報ウィンドウにてルールセットをリストから選択します。
2. そのルールセットのコンテキストメニューボタン をクリックします。
3. 表示されるコンテキストメニューから、ルールセットを編集を選択します。



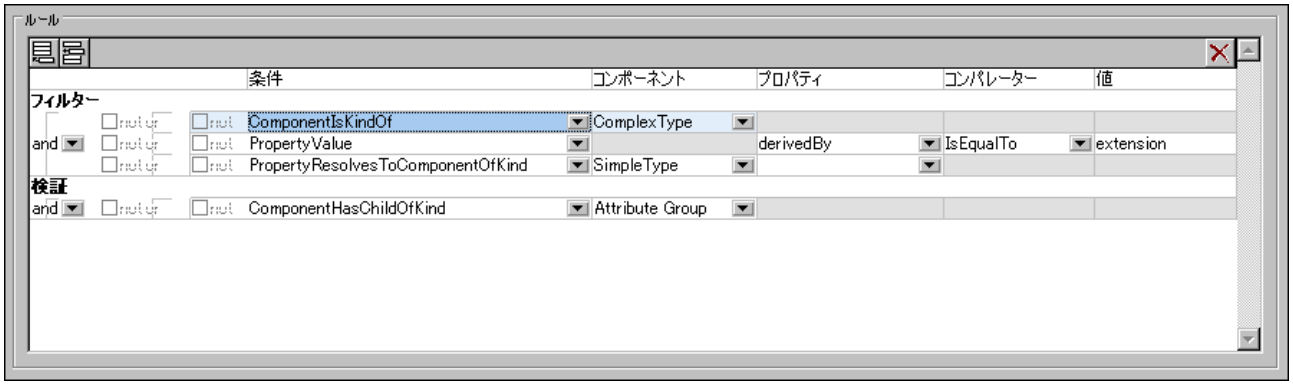
スキーマルールエディターには2つのペインがあります:

- （エディターの上部にある）ルールセットペインでは、ルールの追加や削除を行うことができます。対応する（「**追加**」または「**挿入**」）ボタンをクリックすることで、ルールを表す空の行が追加または挿入されます。ルールの削除を行うには、目的の行を選択した後ペインの右上にある「**削除**」ボタンをクリックします。このペイン内の各ルールには名前と、メッセージテキスト、そして（ルールの違反を発見した際にバリデーターがエラーまたは警告を返すかを決定する）重大度を入力するフィールドが存在します。
- （エディターの下部にある）ルールペインには、上部のルールセットペインで選択されているルールの詳細が表示され、ルールの詳細を変更することができます。ルールの定義に関する詳細については、以下のルールの定義 393を参照ください。

ルールセットにあるルールの編集が完了したら、「**保存**」をクリックしてルールセットファイルのルールを保存します。

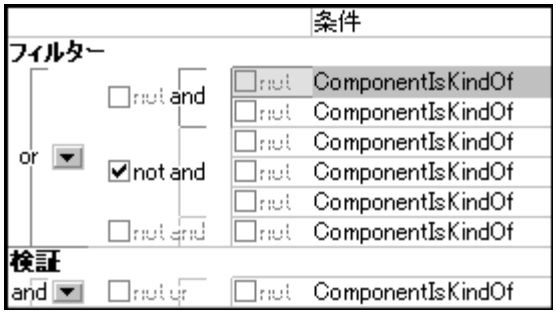
ルールの定義

ルールの定義や編集を行うには、上部のルールセットペインからルールを選択します。選択されたルールの定義がルールペインに表示され、編集を行うことができるようになります。以下のスクリーンショットに示されるルールは以下のように定義されています: *複合型が単純型の拡張であれば、Attribute Group の子を持たなければなりません。*

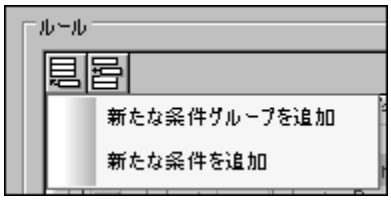


検証条件セットとフィルター条件セット:

- 各ルールには検証条件ならびにフィルター条件のセットがそれぞれ1つずつあります(上のスクリーンショットの第1列を参照)。
- 検証条件が評価されるには、フィルター条件のセットが真(true)とならなければなりません。
- 条件の各セット(検証ならびにフィルター)は1 つ以上の条件グループにより構成され、各条件グループには1 つ以上の条件が含まれます。上のスクリーンショットの例では、1 つの条件が含まれる条件グループが検証セットにあり、各自 1 つの条件を持つ3 つの条件グループがフィルターセットにはあります。以下のスクリーンショットでは、フィルターセットに3 つの条件グループが存在します: 最初のグループには2 つの条件、2 つ目のグループには3 つ、そして最後のグループには1 つの条件が含まれます。



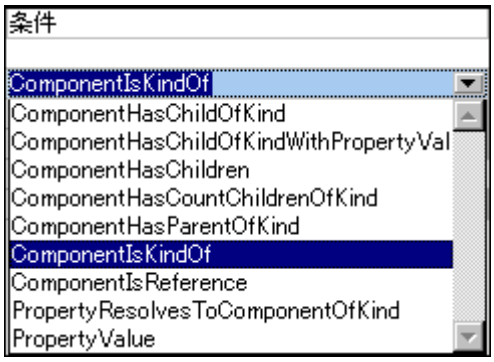
- (条件の左にある) *not* チェックボックスをクリックすることで各条件の否定を行うことができます。
- 条件グループにある論理演算子 **and** を使えば、グループ内の条件すべてが真となることで条件グループ全体が真として評価され、**or** では条件グループ内の条件が1 つでも真と評価されれば条件グループ全体が真として評価されます。
- (条件グループの論理演算子の左にある) *not* チェックボックスをチェックすることで各条件グループの否定を行うことができます。
- 外側にある **and** 論理演算子を使えば(検証またはフィルター以下の) 条件グループ全てが真となることで(検証またはフィルター) セット全体が真になり、**or** 論理演算子を使えば条件グループのうち1つが真であればセット全体が真となります。
- 外側にある(条件グループのための) コンボボックスの内容を変更することで論理演算子を変更することができます。内側の(条件グループ内の) 論理演算子の値は外側の論理演算子とは逆のものに変更されます。
- 選択された条件を起点として条件グループまたは条件を追加、挿入することができます。条件を選択している状態で、(ペインの左上にある)「**追加**」または「**挿入**」ボタンをクリックして、メニューから目的のアイテム(条件ループまたは条件)を選択します(以下のスクリーンショットを参照)。



条件の種類

条件は以下のうちのどれかに分類されます（以下のスクリーンショットも参照）：

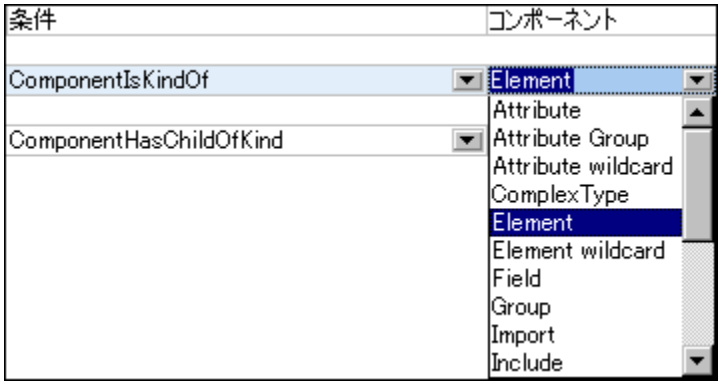
- コンポーネント（ドロップダウンリストの中で *コンポーネント* から始まるもの: 以下のスクリーンショットを参照）
- プロパティ（*Property　値*）
- コンポーネントとプロパティの混合（名前に *Property* と *Component* が入っているもの）



（上のスクリーンショットにあるように）*条件*列のドロップダウンリストから条件の種類が選択されます。3つある条件の種類については以下の記述を参照ください。

コンポーネント条件

コンポーネントの条件（*Component* で始まる条件）では、（以下のスクリーンショットにあるように）コンポーネント列にてコンポーネントを指定する必要があります。コンポーネントフィールドにあるドロップダウンリストからコンポーネントを選択します。コンポーネント条件ではその他のフィールド（プロパティやコンパレーター）の値を指定する必要が無いので、他のフィールドはグレーで表示されます。



上のスクリーンショットの例では、コンポーネントの種類が *Element* となっているものに対してフィルター条件が指定されています。例えば、これに加えて検証条件にコンポーネントは Notation という子を持たなければならないと指定されていれば、「*Element* コンポーネントは

Notation という種類の子を持たなければならない」という条件が完成します。検証条件の `not` オプションがチェックされていれば、「*Element* コンポーネントは *Notation* という種類の子を持ってはならない」という条件になります。

プロパティ条件

プロパティの条件は(以下のスクリーンショットにあるように) *プロパティ値* で決定されます。このような条件ではプロパティの値そのものが指定されるため、プロパティならびにコンパレーター両方の列(そして必要な場合は値列)で入力を行う必要があります。*コンポーネント*列への入力を行う必要は無いのでグレーで表示されます。プロパティ列のドロップダウンリストに表示されるアイテムには(`default` や `maxOccurs` といった) XML 属性の他にも(`derivedBy` といった)コンポーネントの論理プロパティが含まれます。

上のスクリーンショットでは *model* プロパティの値が *all* に等しいというルールが(`not` チェックボックスにより)否定されています。*Model group* コンポーネントに対するフィルター条件と組み合わせることで、スキーマは `xsd:all` 要素を含んでいけないという意味になります。

注意: 以下の点に注意してください:

- `IsQNameEqualTo` コンパレーターを使用するときは、対応する値を `{URI}localName` の形式で記述する必要があります。例えば、以下のような値になります: `{http://www.w3.org/2001/XMLSchema}NOTATION`
- `default` プロパティは空の値と共に記述(`<element name default=""/>`)、または除外することができます(`<element name/>`)。

コンポーネントとプロパティを混合する条件

コンポーネントとプロパティの混合した条件は以下になります:

- `ComponentHasChildOfKindWithPropertyValue` : 子要素のコンポーネントの種類と、プロパティの値を指定します。
- `PropertyResolvesToComponentOfKind` : コンポーネントの種類を解決するプロパティが指定されます。*コンパレーター*と*値*の列は空になります。

条件の否定

条件の左にある(内側の) *not* チェックボックスをチェックすることで、その条件は否定されます。条件グループの論理演算子の左(外側にある) *not* チェックボックスをチェックすることで、条件グループを否定することができます。

6.5 SchemaAgent との連携

XMLSpy は Altova の SchemaAgent テクノロジーと連動するように設定することができます。

SchemaAgent テクノロジー

SchemaAgent テクノロジーにより、複数のスキーマ間のリレーションシップを構築することができます。以下により構成されています:

- (フォルダーやネットワークの)検索パス上にある、複数のスキーマ間のリレーションシップに関する情報を保持、提供する SchemaAgent Server。
- A SchemaAgent クライアント、(i) スキーマ間のリレーションシップを作成し、(ii) これらのスキーマを管理するために接続されている (スキーマの名前の変更、移動、削除する)SchemaAgentサーバーからの情報を使用する Altova の SchemaAgent 製品。

SchemaAgent サーバーの2つの種類を使用することができます:

- ネットワーク上にインストールし、利用することのできる Altova SchemaAgent Server。
- SchemaAgent クライアント製品である Altova SchemaAgent 。より軽量なサーバーである LocalServer が搭載されており、SchemaAgent がインストールされているマシンからのみアクセスすることができます。

XMLSpy から SchemaAgent テクノロジーを利用することで、SchemaAgent サーバーから得られるスキーマ情報を使用して、スキーマをスキーマビューにて直接編集することができます。ここでは、XMLSpy が SchemaAgent サーバーに接続し、SchemaAgent Client とのやり取りを行いながら SchemaAgent Server へリクエストを送信します。XMLSpy が SchemaAgent と連動するよう設定された場合、スキーマビューの入力ヘルパーには、現在のスキーマビューでアクティブなスキーマから得られたコンポーネントだけでなく、接続している SchemaAgent サーバーの検索パスにあるその他のスキーマから得られたコンポーネントも表示されます。この機能を使うことで、これらのコンポーネントに直接アクセスすることができます。他のスキーマに属しているコンポーネントのコンテンツモデルを、スキーマビューで確認し、これらコンポーネントを再利用することができます。スキーマ間のリレーションシップを構築して、XMLSpy から複雑なスキーマの管理を直接行うこともできます。

SchemaAgent と SchemaAgent Server のインストール

SchemaAgent ならびに SchemaAgent Server のインストール、そしてサーバー上の検索パスの設定については、SchemaAgent ユーザーマニュアルを参照ください。

XMLSpy を SchemaAgent クライアントとしてセットアップする

XMLSpy が SchemaAgent クライアントと連動して使用するには以下の操作を行う必要があります:

- Altova ウェブサイトから SchemaAgent をダウンロードします。SchemaAgent の LocalServer を使用することでスキーマを提供することができるようになります。LocalServer にて検索パスを設定する方法については SchemaAgent のユーザーマニュアルを参照ください。LocalServer 上の検索パスの構成に関する詳細は SchemaAgent ユーザーマニュアルを確認してください。
  注意: 無料トライアル期間が終了した後に SchemaAgent を使用するには有効なライセンスを購入する必要があります。Altova MissionKit 製品パッケージ Enterprise Edition には SchemaAgent 製品とライセンスキーが含まれています。(SchemaAgent Server アプリケーションは Altova MissionKit パッケージには含まれていません。)
- 上記に加えて、必要ならば Altova ウェブサイトからネットワークベースの SchemaAgent Server をダウンロードしてください。
- SchemaAgent(そして SchemaAgent Server) の検索パスを定義します。検索パスの詳細な定義方法については SchemaAgent ユーザーマニュアルを参照ください。(検索パスとはリレーションシップのマッピングに使われる、スキーマが含まれるフォルダーへのパスのことです)。
- XMLSpy 内から SchemaAgent への接続を開始します。

重要点: Altova から提供される SchemaAgent ならびに SchemaAgent に関連する製品(XMLSpy を含む)のバージョン 2005 リリース3 以降の製品は、それ以前のバージョンの SchemaAgent または SchemaAgent に関連する製品との互換性を持ちません。

XMLSpy 内の SchemaAgent コマンド

XMLSpy 内の SchemaAgent 機能はスキーマビューからのみ利用することができ、スキーマ設計メニューのメニューコマンド（*スクリーンショットを参照*）ならびにスキーマビューの入力ヘルパーからアクセスすることができます。

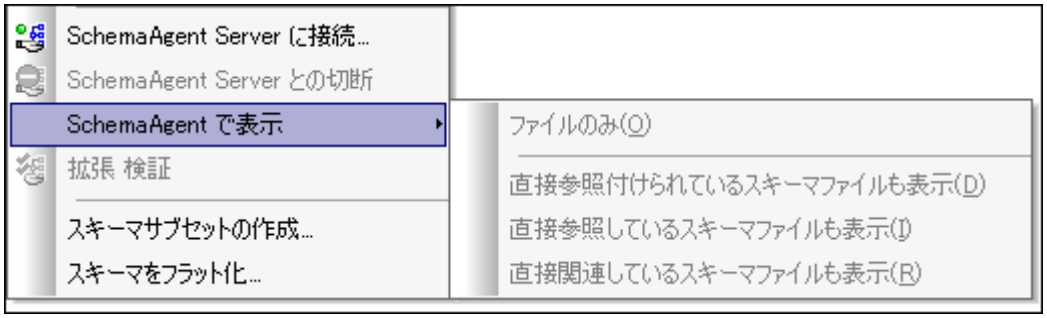
メニューコマンドでは一般的な管理機能が提供されます。入力ヘルパー（ならびにドラッグアンドドロップといった GUI を使った機能）によりスキーマの編集を実際に行うことができます。

このセクションではスキーマビューで利用可能な SchemaAgent の機能の使用方法について説明します。

6.5.1 SchemaAgent Server へ接続

メモ 接続を確立するには SchemaAgent Client がインストールされている必要があります。

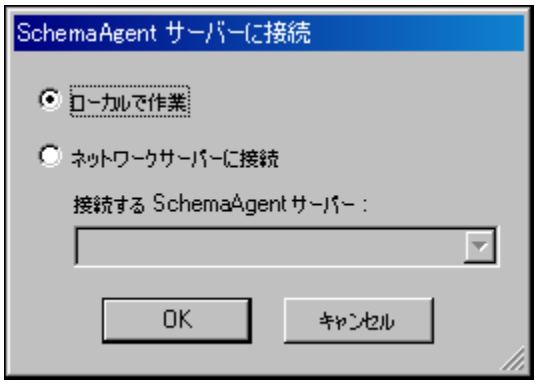
SchemaAgent Server に接続する前に only the **SchemaAgent Server に接続** コマンドのみが **スキーマ設計** メニュー内で使用することができます。**スキーマ設計** メニュー内の他の SchemaAgent コマンドは無効化されています（*スクリーンショット参照*）。SchemaAgent Server への接続が確立されれば、その他のメニューアイテムも選択可能になります。



接続方法

SchemaAgent サーバーへ接続するには、以下の操作を行ってください:

1. Click the Connect to SchemaAgent サーバーに接続アイコン (「**スキーマ設計 | SchemaAgent Server**に接続」)をクリックします SchemaAgent Server への接続ダイアログが開かれます（*下のスクリーンショット*）:

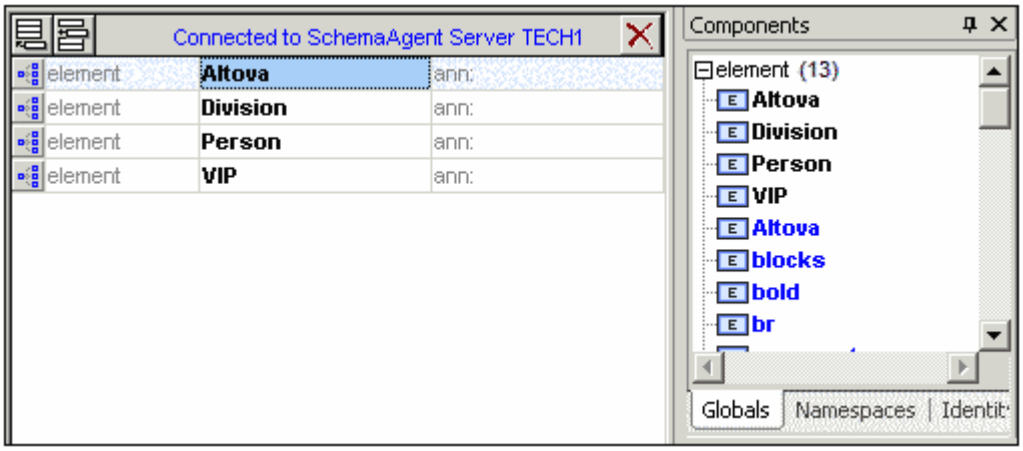


2. ローカルサーバー(Altova SchemaAgent に同封されている SchemaAgent サーバー)またはネットワークサーバー(無料で利用することのできる Altova SchemaAgent Server 製品)を使用することができます。ローカルで作業を選択し、**OK**を押すと、SchemaAgent のローカルサーバーが起動され 接続が確立されます。ネットワークサーバーへ接続を選択する場合、選択された SchemaAgent Server が起動していなければなりません。

> **Windows XP SP2 上で動作しているサーバーについて**
> SchemaAgent Server 名が SchemaAgent Server への接続ダイアログにリストされていても接続できない場合、サーバーがネットワークの名前解決プロセスの一部ではない場合があります。Windows XP SP2 ではファイヤーウォールにより名前解決をデフォルトでブロックしています。
>
> このようなサーバーに接続するには、以下のうちどれかを行ってください:
>
> - サーバーの設定を変更して、名前解決を有効にする
> - サーバーの IP アドレスを 接続ダイアログボックスの編集フィールドに入力します。
>
> SchemaAgent Client には最後に成功した接続の接続文字列が保存されるため、この操作は1度だけ行う必要があります。

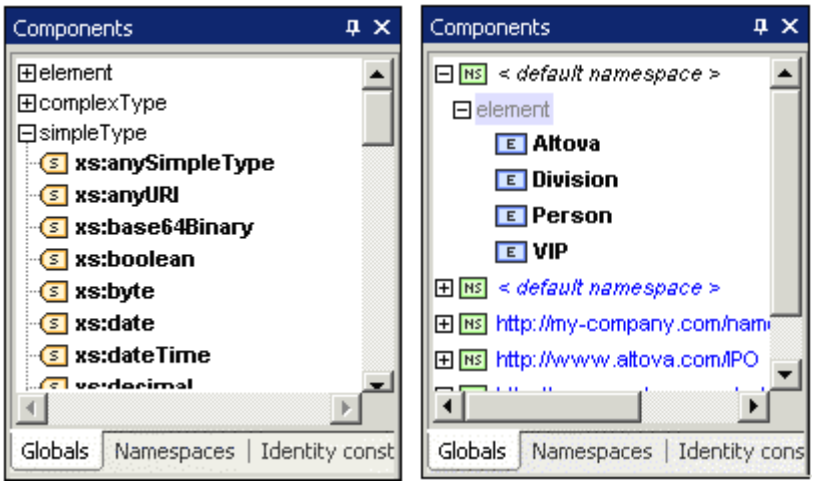
SchemaSchemaAgent サーバーに接続後のスキーマビュー

SchemaAgent サーバーへの接続が確立された後は、スキーマビューの表示は以下のようになります:



メモ

- グローバルビューの上部には「SchemaAgent Server と接続」という文字列が表示され、接続を確立したサーバー名が表示されます。
- サーバー検索パス上にある全てのスキーマやスキーマコンストラクトへアクセスすることができるようになります。SchemaAgent スキーマはグローバル要素のような複合型を構成し、(**黒字で表示される**) アクティブなスキーマの下で単純型は **太字の青色のテキスト** で表示されます。.

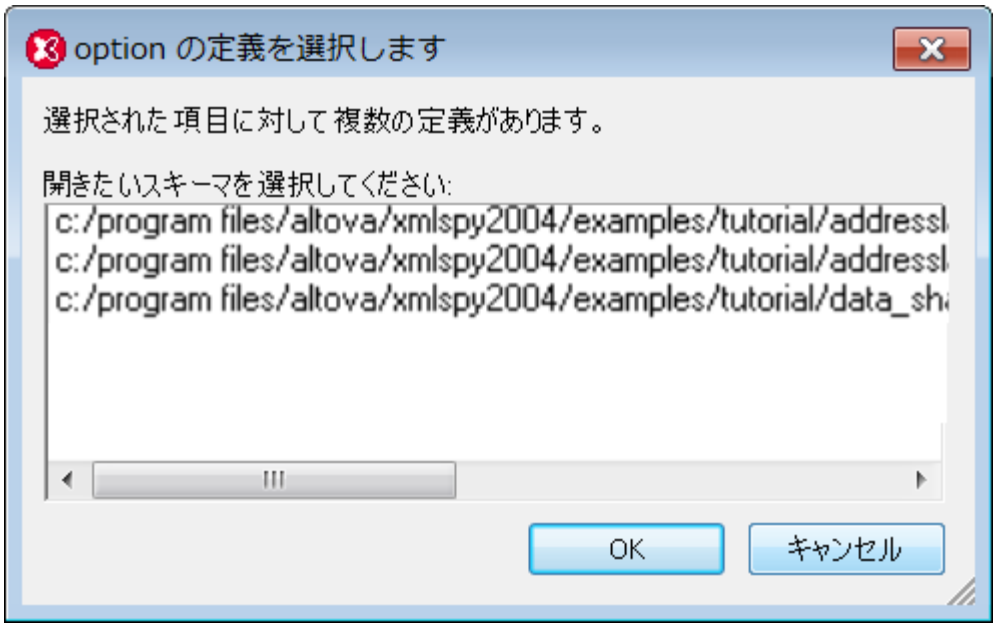


スキーマコンストラクトは、コンポーネント入力ヘルパーの対応するタブである型(グローバル)、名前空間、同一性制約から参照することができます。

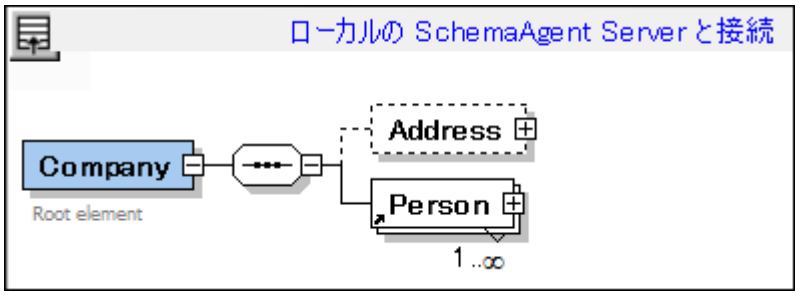
6.5.2 検索パス上のスキーマを開く

この例では、SchemaAgent Server で定義された検索パス上にあるスキーマを開く方法を説明します。ここでは `..\Tutorial` フォルダー以下にある `Db2schema.xsd` ファイルを、アクティブなスキーマとして使用します。コンポーネント入力ヘルパーのグローバルタブをアクティブにします。

1. コンポーネント入力ヘルパーにある青色で記された Company アイテムまで移動し、アイテムをダブルクリックします。定義へ移動ダイアログが開かれます。



2. Click the `Addresslast.xsd` エントリを選択して「OK」を押して確認します。これにより `MFCompany.xsd` スキーマが開かれ、`Company` 要素のコンテンツモデルが表示されます。



メモ 要素や複合型、または単純型といったSchemaAgent スキーマコンストラクトをダブルクリックすることで、関連するスキーマ(そしてインクルードされた他のスキーマ)がXMLSpy で開かれます。

6.5.3 IIR の使用

XML スキーマでは、Import、Include、そして Redefine (IIR) ステートメントを使用することでスキーマをモジュール化することができます。各メソッドでは異なる名前空間が必要になります。IIR を行う際に必要となる項目はIIR を作成する際にSchemaAgent Client と XMLSpy により自動的にチェックされます。

インポート(Import)、インクルード(Include)、そして再定義(Redefine) (IIR)

スキーマコンストラクトを「挿入」するにはいくつか方法があります:

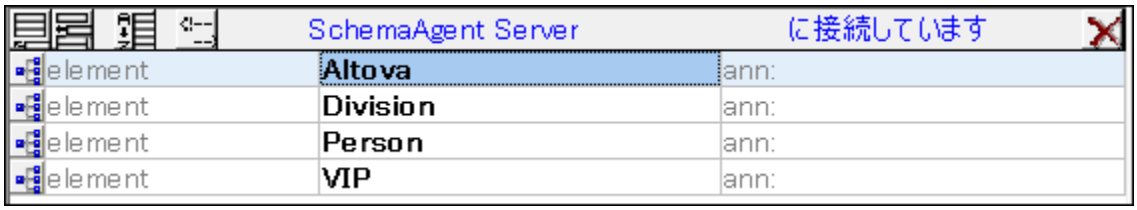
- グローバル要素はコンポーネント入力ヘルパーから、(スキーマビューの) スキーマコンポーネントのコンテンツモデルに直接ドラッグすることができます。
- 新規要素や属性などを定義する際に自動的に開かれるリストボックスから、複合型や単純型のようなコンポーネントを選択することができます。
- 複合型の様なコンポーネントを作成、更新する際に、詳細入力ヘルパーからコンストラクトを選択することができます。

スキーマコンポーネントの組み込み

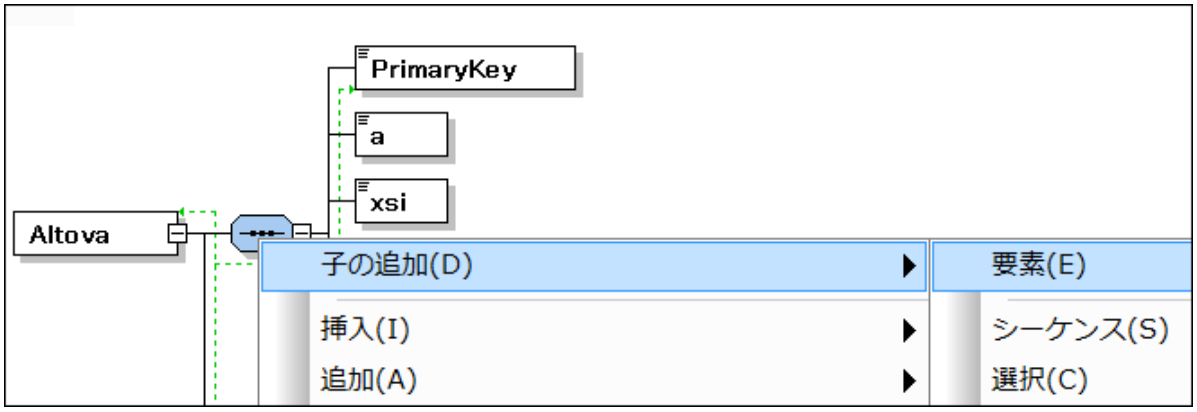
この例ではアクティブなスキーマとして..\Tutorial フォルダー内のDb2schema.xsd ファイルを使用します。コンポーネント入力ヘルパーグローバルが有効化されています。

SchemaAgent Server スキーマにあるスキーマコンストラクトを使用するには:

1. SchemaAgent サーバーに接続していることを確認します(SchemaAgent サーバーへ接続[398]を参4.Altova 以下にあるシーケンスコンポジッターを右クリックして、メニューオプションから「子を追加 | 要素」を選択します。照)。
2. DB2Schema.xsd ファイルを、この例のために(例えば) Altova-officeという名前に変更します。

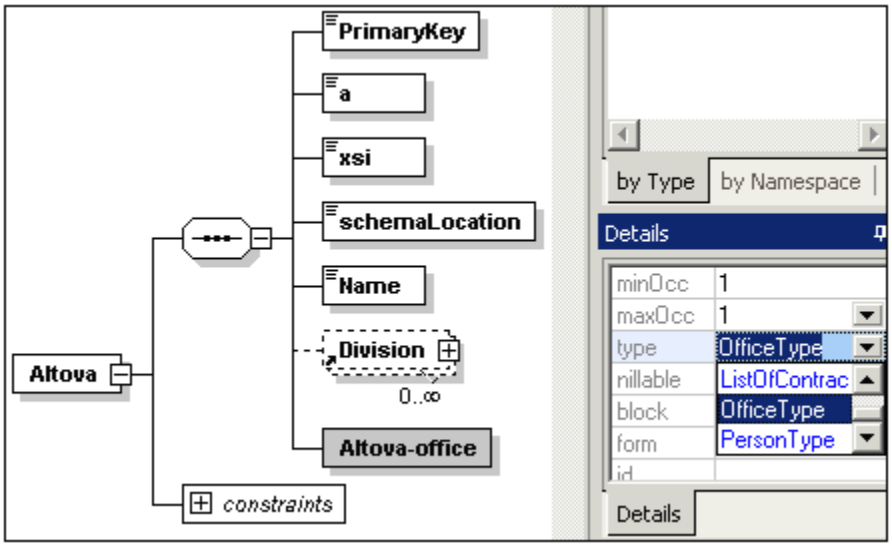


3. スキーマ概要に表示されている Altova 要素のアイコンをクリックして、コンテンツモデルを表示します。
4. Altova シーケンスコンポジッターを右クリックして、メニューオプション「子を追加 | 要素」を選択します。サーバープロジェクト内のすべてのグローバル要素を含むリストボックスはこの時点で自動的に開かれます。1つ選択するとその要素は組み込まれます。

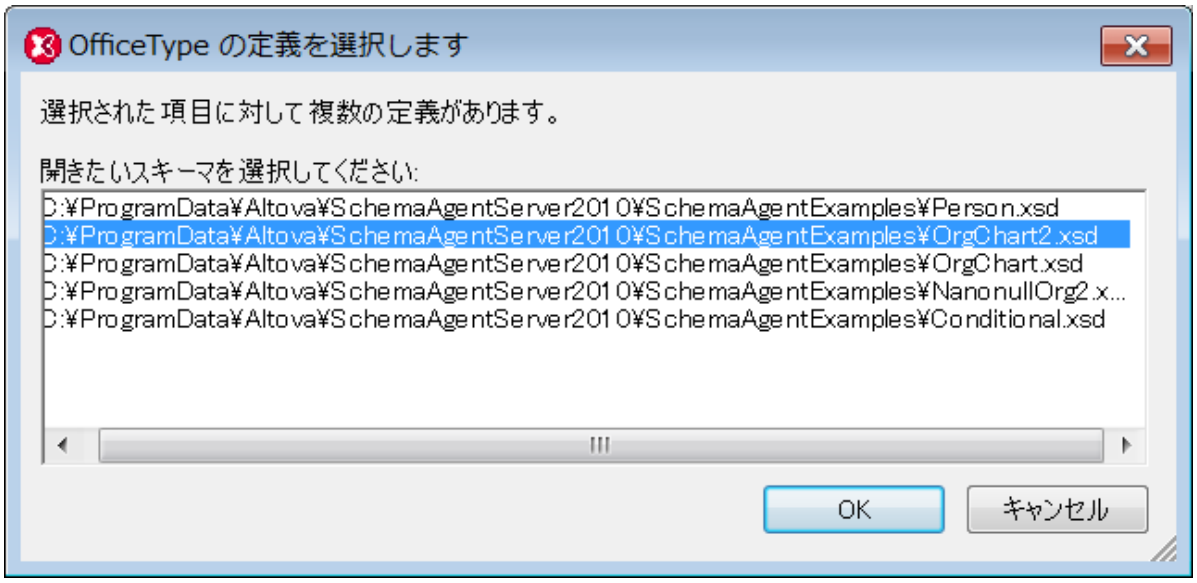


5. 新たに作成した要素の名前をAltova-office として、「Enter」で確定します。
6. 詳細入力ヘルパーにあるtype コンボボックスをクリックして、OfficeType を選択します。

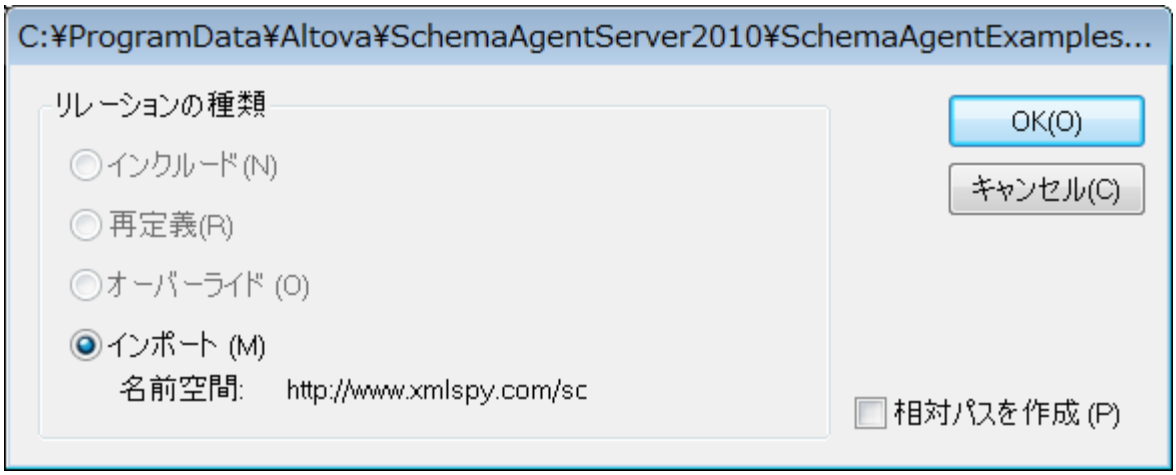
(C) 2015-2021 Altova GmbH



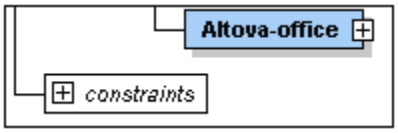
OfficeType の定義を選択するダイアログボックスが表れます。



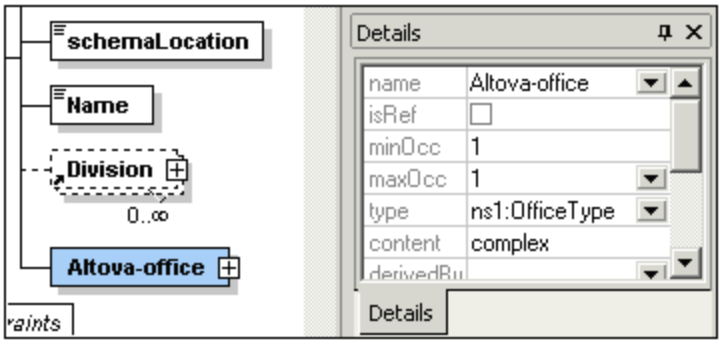
7. `Orgchart.xsd`を選択して「OK」をクリックして確認します。



8. 「OK」をクリックします。インポートコマンドが自動的に選択されます。`Altova-office` 要素に展開アイコンが表示されます。

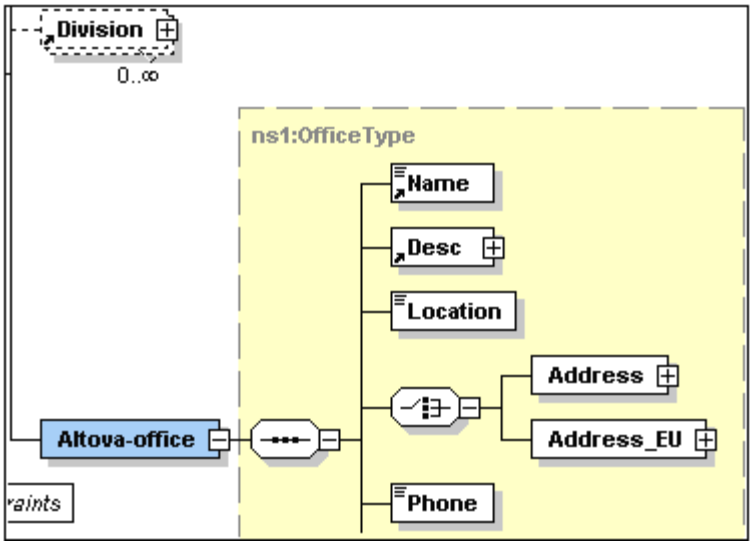


メモ `Orgchart.xsd` スキーマファイルがインポートされると、各スキーマのターゲット名前空間は異なるものと見なす必要があるため詳細入力ヘルパーの type エントリは `ns1:OfficeType` に変更されます。インポートコマンドもスキーマに追加されます。



9. 展開ボタンをクリックして、`OfficeType` コンテンツモデルを確認します。

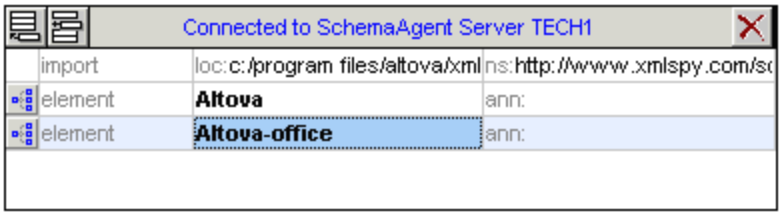
(C) 2015-2021 Altova GmbH



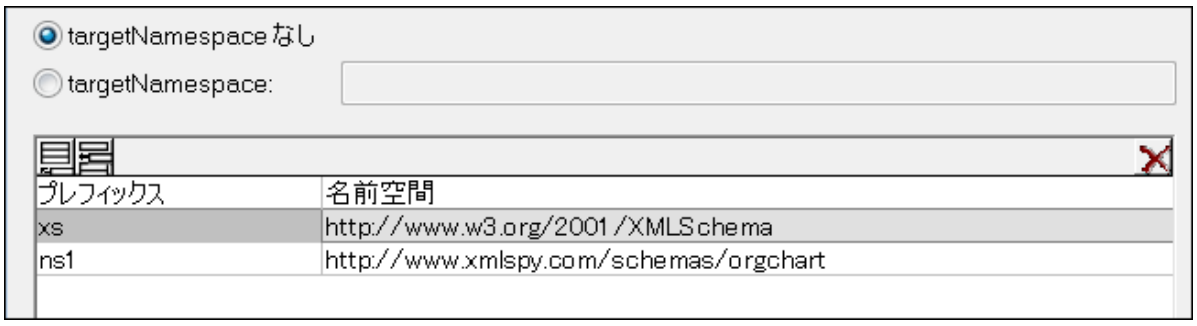
10. **F8** キーを押してスキーマを検証します。「スキーマは妥当です」というメッセージが表示されるはずです。

スキーマの整理:

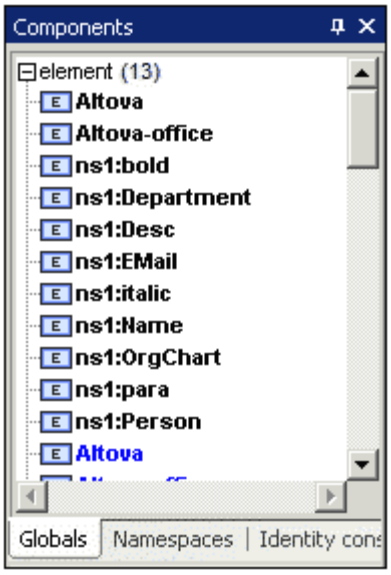
1. コンテンツモデルの `Division` 要素を削除します。
2. 概要へ戻るアイコン をクリックして、スキーマ概要へ画面を切り替えます。
3. 以下のグローバル要素を削除します: `Division`、`Person`、`VIP`。



4. メニューオプションから「**スキーマ設計 | スキーマ設定**」を選択し、名前空間の設定が変更されていることを確認します。



`ns1` プレフィックスが `www.xmlspy.com/schemas/orgchart` 名前空間に自動的に追加されます。コンポーネント (*スクリーンショット参照*) と詳細入力ヘルパーは `ns1`: namespace prefix.名前空間プレフィックスからインポートされた全てのコンストラクトを表示します。



メモ

- XMLSpyを使ってSchemaAgent Server コントロールを変更することで、SchemaAgent Server のパス上にある（変更されたスキーマを参照している）他のスキーマも自動的に更新されます。
- 作業中のスキーマがSchemaAgent Server パス上にある場合、入力ヘルパーで複製されたコンストラクト要素や単純型を（黒色と青色の違いにより）確認することも可能です。

6.5.4 SchemaAgent でスキーマを閲覧

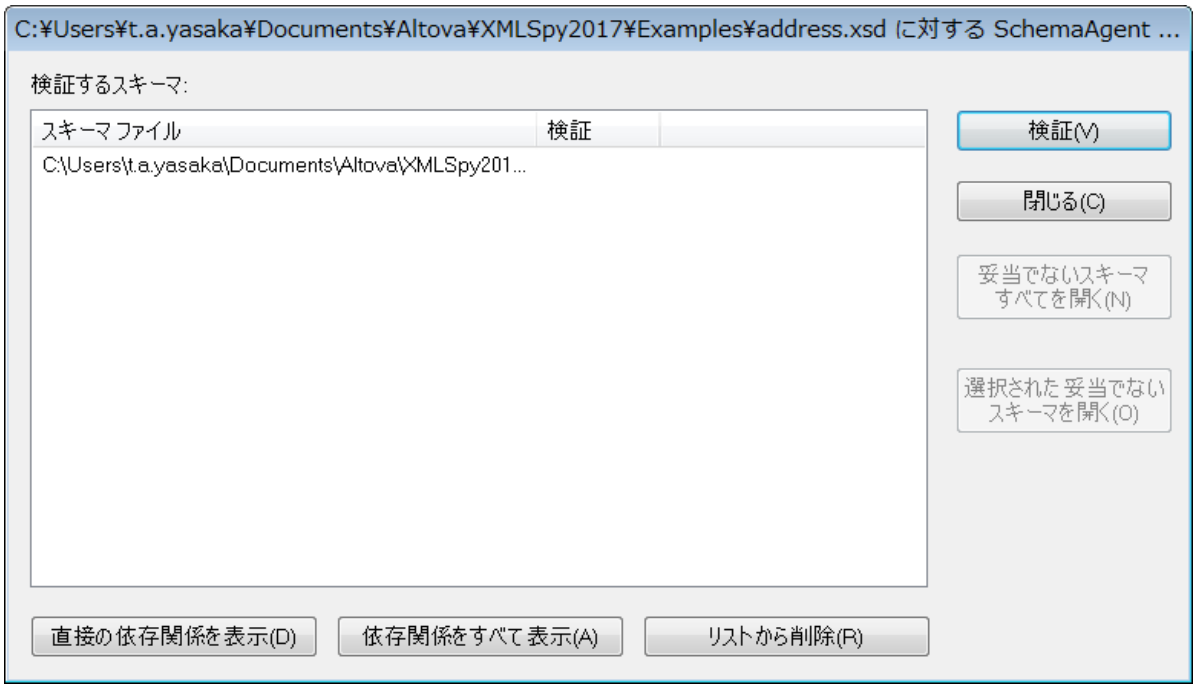
アクティブならびに関連したスキーマを SchemaAgent を使って確認、編集するには、メニューオプションから「**スキーマ設定 | SchemaAgent で表示 | *スキーマまたは関連するスキーマ***」を選択します（以下のスクリーンショットを参照）。
SchemaAgent を使って、アクティブなスキーマのみ（「**ファイルのみ**」コマンド）、またはアクティブなスキーマと、(i) 直接参照されている全てのスキーマか、(ii) 直接参照している全てのスキーマか、(iii) 直接関連する全てのスキーマを開くことができます。

6.5.5 SchemaAgent 検証

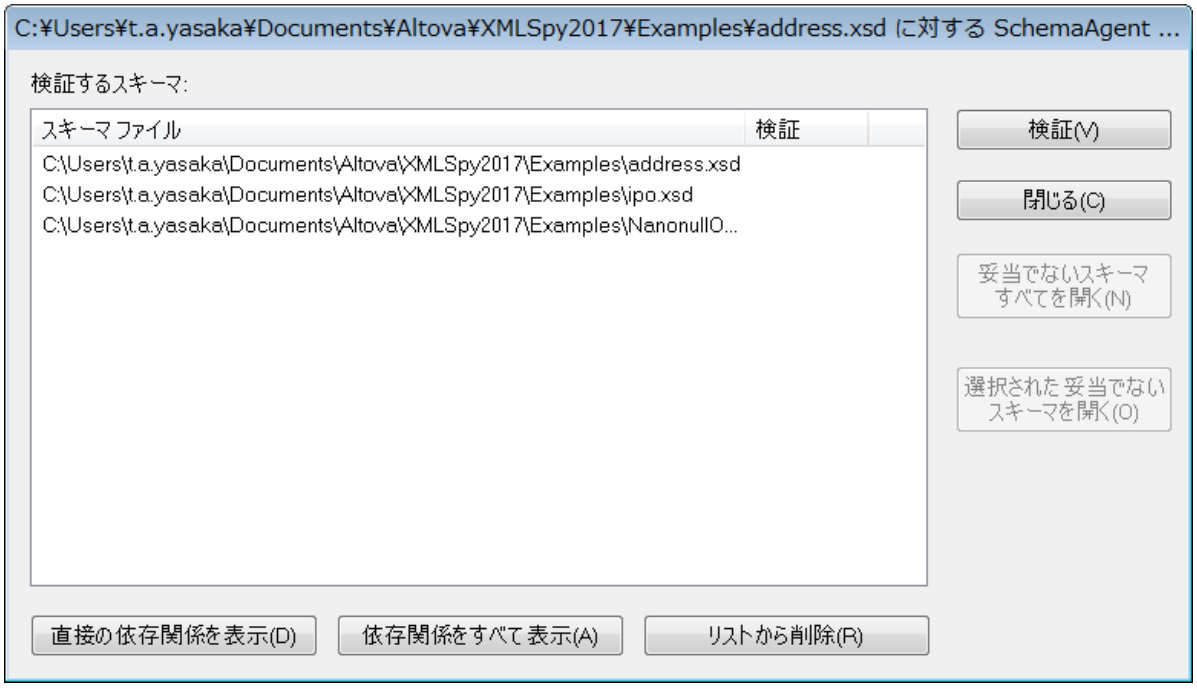
SchemaAgent と組み合わさったXMLSpy により、現在アクティブなスキーマのみならず、それに関連したスキーマの検証も行うことができます。このようなスキーマの検証は、SchemaAgent 検証と呼ばれます。関連するスキーマには (i) 直接の依存関係を持つスキーマ（直接参照している、または直接参照されているスキーマ）と、(ii) 全ての依存関係を持つスキーマ（直接の依存関係に加え、中間的なスキーマを介することで、他のスキーマに関連したスキーマセットの間接的な依存関係も含まれます）の2 種類があります。

SchemaAgent 検証がどのように行われるかは、以下の例により示されます。この例では `address.xsd` というファイルがXMLSpy のスキーマビューのアクティブなスキーマであることを前提にしています。SchemaAgent 検証のコマンドを使用にするには、SchemaAgent Server の検索パスにアクティブなファイルと依存関係を持つファイルが含まれることを確認してください。以下の操作を行ってください：

1. ツールバーにある「**SchemaAgent 検証**」アイコン をクリックするか、メニューアイテムの「**スキーマ設計 | SchemaAgent 検証**」を選択します。この操作によりSchemaAgent 検証ダイアログが開かれ、アクティブなスキーマのみ、または関連するスキーマまでも検証を行うか選択することができます（以下のスクリーンショットを参照）。

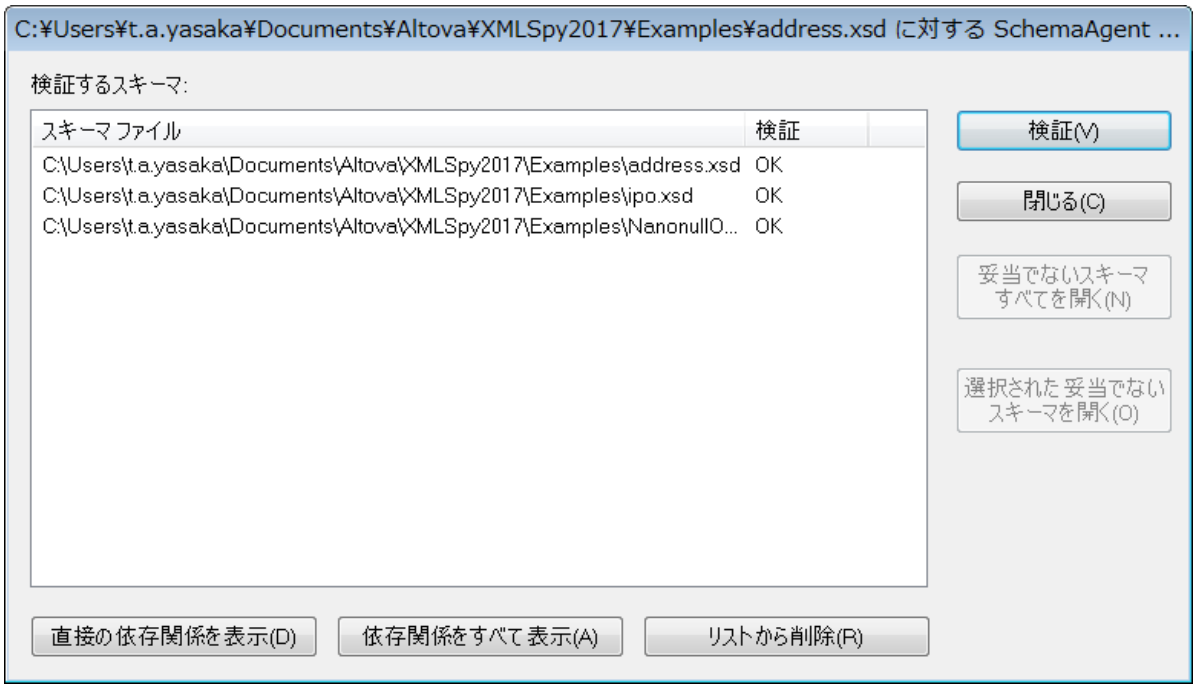


2. スキーマをリストに追加するには、「**直接の依存関係を表示**」または「**依存関係を全て表示**」ボタンをクリックします。この例では、「**依存関係を全て表示**」ボタンをクリックすることで、直接参照されているファイル、または間接的に参照されているファイルがリストに表示されます。



この時点で、必要に応じて(「**リストから削除**」により)スキーマをリストから取り除くことができます。

3. 「**検証**」ボタンをクリックして、リストボックスに表示されているスキーマ全ての検証を行ないます。



検証列に検証結果が表示されます。

妥当でないスキーマ全て、または選択されうち妥当でないスキーマセットを XMLSpy で開くことができます。

6.6 XMLSpy 内のカタログ

XMLSpy は OASIS XML カタログメカニズムのサブセットをサポートします。XMLSpy のカタログ機能により、広く使われているスキーマ（または スタイルシートなどその他のファイル）をローカルユーザーフォルダーから取得することができます。これにより全体的な処理スピードが向上し、（ネットワークに接続していない状態でも）ユーザーはオフラインで作業することができるようになり、（URI の変更はカタログファイル内部に限定されるので）ドキュメントのポータビリティーが向上します。

XMLSpy のカタログ機能は以下のように動作します。

- カタログの仕組み 409
- XMLSpy 内のカタログ構造 410
- カタログのカスタム化 411
- 環境変数 413

カタログの詳細に関しては XML カタログ仕様 を参照してください。

6.6.1 カタログのしくみ

カタログを DTD と XML スキーマをリダイレクトするために使用することができます。両方のケースのメカニズムの後ろにあるコンセプトは同じですが、違いの詳細は以下に説明されている通りです。

DTD

カタログは DTD への呼び出しをローカルの URI にリダイレクトするために使用されます。これはマッピング、カタログファイル、ローカル URI により必要とされる パブリックまたはシステム識別子により達成されます。XML ファイル内の DOCTYPE 宣言が読み取れれると、パブリックまたはシステム識別子が必要とされるローカルリソースをカタログファイルマッピングによりロケートします。

人気のあるスキーマに関しては PUBLIC 識別子は通常定義済みです。このためカタログファイル内の URI のみが正しいローカルコピーに対して PUBLIC 識別子 をマップします。XML ドキュメントが解析されると内部の PUBLIC 識別子が読み取られます。この識別子がカタログファイル内で見つかると、カタログファイル内の対応する URL がルックアップされ、スキーマがこのロケーションから読み取られます。例えば、次の SVG ファイルが XMLSpy 内で開かれるとします:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="20" height="20" xml:space="preserve">
  <g style="fill:red; stroke:#000000">
     <rect x="0" y="0" width="15" height="15"/>
     <rect x="5" y="5" width="15" height="15"/>
  </g>
</svg>
```

カタログはこの SVG ファイルの PUBLIC 識別子を検索します。カタログファイルは以下のエントリを含んでいます:

```
<catalog>
  ...
    <public publicId="-//W3C//DTD SVG 1.1//EN" uri="schemas/svg/svg11.dtd"/>
  ...
</catalog>
```

この場合 PUBLIC 識別子のために一致するものがあります。この結果 SVG DTD のためのルックアップがカタログファイルに相対的な URL schemas/svg/svg11.dtd にリダイレクトされます。これは SVG ファイルのために DTD として使用されるローカルファイルです。カタログ内で Public ID のためのマッピングが存在しない場合、XML ドキュメント内の URL が使用されます（上の SVG ファイルのサンプルの場合、これはインターネット URL です: http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd）。

XML スキーマ

XMLSpy 内では XML スキーマをリダイレクトするためにカタログを使用することができます。XML インスタンスファイル内で、スキーマへのレファレンスが XML ドキュメントのトップレベルの要素の xsi:schemaLocation 属性内で発生します。例えば、

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

xsi:schemaLocation 属性の値には2つの部分があります: （上で緑色で表示されている）名前空間のパートとハイライトされている）URI パート。名前空間の部分は代替リソースをマップするためにカタログ内で使用されています。例えば、以下のカタログエントリは上のスキーマレファレンスを代替のロケーションのスキーマにリダイレクトします。

```
<uri name="http://www.xmlspy.com/schemas/orgchart" uri="C:\MySchemas\OrgChart.xsd"/>
```

通常 xsi:schemaLocation 属性の値の URI 部分は実際のスキーマのロケーションのパスです。しかしながら、スキーマがカタログを介して参照される場合、URI の部分は実際の XML スキーマをポイントする必要はなく、xsi:schemaLocation 属性の文法の妥当性が保持されるために存在する必要があります。Foo の値は、例えば、属性の値の部分のが妥当であるために十分です。

6.6.2 XMLSpy 内のカタログ構造

XMLSpy が開始されると、（以下のリスティングで示されるような構造を持った）RootCatalog.xml というファイルがロードされ、ファイル内部に記述されているカタログファイルが検索されます。このファイルを修正して、検索するカタログファイルを増やすこともできます（各カタログファイルは nextCatalog 要素にて記述されます）。各カタログファイル内に記述されている URI が、ファイル内で指定されたマッピングに従って解決されます。

Listing of RootCatalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
   xmlns:spy="http://www.altova.com/catalog_ext"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:entity:xmlns:xml:catalog Catalog.xsd">
  <nextCatalog catalog="%PersonalFolder%/Altova/%AppAndVersionName%/CustomCatalog.xml"/>
  <!-- Include all catalogs under common schemas folder on the first directory level -->
  <nextCatalog spy:recurseFrom="%CommonSchemasFolder%" catalog="catalog.xml"
spy:depth="1"/>
  <nextCatalog spy:recurseFrom="%ApplicationWritableDataFolder%/pkgs/.cache"
catalog="remapping.xml" spy:depth="0"/>
  <nextCatalog catalog="CoreCatalog.xml"/>
</catalog>
```

上のリスティングはカスタムカタログ（named CustomCatalog.xml）と（W3C XML スキーマと SVG スキーマなどの一般的に使用されるスキーマをロケートするカタログのセットを参照します

- （%PersonalFolder% を介してロケートされる）CustomCatalog.xml は個人用フォルダーにあります自身のマッピングを作成するスケルトンファイルです。共通のスキーマフォルダー内のカタログファイルにより指されていない必要とするスキーマのためにマッピングを CustomCatalog.xml に追加することができます。OASIS カタログメカニズムのサポートされる要素を使用してこれを行います（*下を参照*）。

- （共通に使用されるスキーマのセットを含む変数 **%CommonSchemasFolder%** を使用してロケートされる）共通のスキーマフォルダー—これらのスキーマフォルダの内部は保存された対応するスキーマのコピーをポイントする URI に対するパブリックおよび/またはシステム識別子をマップする **catalog.xml** ファイルです。
- XBRL と多種の XBRL タクソノミに関連するスキーマは大きく XBRL タクソノミ マネジャー[722] 助けを借りてローカルにてオンデマンドでインストールされます。（変数 **%ApplicationWritableDatFolder%** によりロケートされる）個別のスキーマとタクソノミはプログラムデータフォルダーの **pkgs/.cache** サブフォルダー内にあるカタログ **remapping.xml** 内でマップされます。
- **CoreCatalog.xml** は XMLSpy アプリケーションフォルダー内にあり、XML ドキュメントの Altova の Authentic ビューを生成するために使用される StyleVision パワースタイルシートなど XMLSpy 固有のプロセスにより使用されるスキーマとスタイルシートをロケートするために使用されます。

ロケーション変数
RootCatalog.xml 内で使用されている変数は以下の値を持っています（*上のリスティング*）:

%PersonalFolder%	現在のユーザーの個人的なフォルダー、例えば C:\Users\<name>\Documents
%CommonSchemasFolder%	C:\ProgramData\Altova\Common2021\Schemas
%ApplicationWritableDataFolder%	C:\ProgramData\Altova

カタログファイルとスキーマのロケーション
多種のカタログファイルの場所にご注意してください。

- **RootCatalog.xml** と **CoreCatalog.xml** は XMLSpy アプリケーションフォルダー内にあります。
- **CustomCatalog.xml** は **MyDocuments\Altova\XMLSpy** フォルダー内にあります。
- **Catalog.xml** 入るはそれぞれ特定のスキーマフォルダー内にあり、これらのスキーマフォルダーは共通のスキーマフォルダー内にあります。

6.6.3 カタログのカスタム化

CustomCatalog.xml（または XMLSpy により読み取られる他のカタログファイル）内でエントリを作成する場合 OASIS カタログ仕様の以下の要素のみを使用してください。下の各要素は属性の値の説明と共にリストされています。詳細に関しては（XML カタログ仕様）を参照してください。各要素はその要素のベース URI を指定するために使用される **xml:base** 属性を取ることができます。

- `<public publicId="PublicID of Resource" uri="URL of local file"/>`
- `<system systemId="SystemID of Resource" uri="URL of local file"/>`
- `<uri name="filename" uri="URL of file identified by filename"/>`
- `<rewriteURI uriStartString="StartString of URI to rewrite" rewritePrefix="String to replace StartString"/>`
- `<rewriteSystem systemIdStartString="StartString of SystemID" rewritePrefix="Replacement string to locate resource locally"/>`

以下の点にご注意してください：

- 多くのスタイルシート同様パブリック識別子が存在しない場合、システム識別子を **system** 要素を介して直接 URL にマップすることができます
- uri 要素を使用して URI を他の URI にマップすることができます。
- rewriteURI と rewriteSystem 要素により URI またはシステム識別子の開始の部分をそれぞれ書き換えることが可能になります。これによりファイルパスの開始部分が置き換えられこの結果他のディレクトリをターゲットにすることができます。これらの要素の詳細に関しては XML カタログ仕様 を参照してください。

リリース2014 以降XMLSpy はXML Catalogs specification (OASIS Standard V1.1, 7 October 2005) 仕様を厳密に順守しています。この仕様は厳密にパブリックID またはシステムID）外部識別子ルックアップを（パブリックID またはシステムID を持たないURI ）URI ルックアップと区別しています。名前空間 URI は単純にパブリックID またはシステムID ではなくURI と考えられ外部識別子ルックアップではなくURI ルックアップとして使用されるべきです。バージョン2014 以前のXMLSpy バージョンではスキーマ名前空間URI は<public> マッピングを使用して訳されてきました。2014 以降から<uri> マッピングが使用されています。

v2014 以前：
```
<public publicID="http://www.MyMapping.com/ref"
   uri="file:///C:/MyDocs/Catalog/test.xsd"/>
```
V-2014 以降:
```
<uri name="http://www.MyMapping.com/ref"
   uri="file:///C:/MyDocs/Catalog/test.xsd"/>
```

XMLSpy が参照されるスキーマを検索する方法

xsi:scemaLocation 属性を介してXMLドキュメント内で参照されるスキーマ（*以下に表示されています*）。
xsi:schemaLocation 属性の値には2 つの部分があります:（緑色で表示されている）名前空間のパートとハイライトされている）URI パート。

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

以下のステップはXMLSpy がシーケンスにより参照されたスキーマを検索するステップを示しています。スキーマは最初のステップでロードされます。

1. **xsi:schemaLocation** 値のURI 部分のためのカタログをルックアップします。**rewriteURI** マッピング内を含みマッピングが検索されるとスキーマのロードのために結果のURI を使用します。
2. **xsi:schemaLocation** 値の名前空間部分のためのカタログをルックアップします。**rewriteURI** マッピング内を含みマッピングが検索されるとスキーマのロードのために結果のURI を使用します。
3. スキーマのロードのために**xsi:schemaLocation** 値のURI 部分を使用します。

スキーマ別のファイル拡張子とインテリジェントな編集

カタログファイルを介して特定のファイル拡張子を持つドキュメントがXMLSpy のインテリジェントな編集機能を指定するスキーマ内のルールに順守するように適用することができます。例えば、HTML DTD に従い妥当な（HTML）ファイルのためにカスタムファイル拡張子 `.myhtml` を作成する場合、**CustomCatalog.xml** を**<catalog>** 要素の子と追加してこれらの拡張子を持つファイルのためにインテリジェントな編集を有効化することができます。

```
<catalog>
  ...
   <spy:fileExtHelper ext="myhtml" uri="schemas/xhtml/xhtml1-transitional.dtd"/>
  ...
</catalog>
```

これはXMLSpy 内の **.myhtml** ファイルのインテリジェントな編集（自動補完、エントリヘルパーなど）をXHTML 1.0 Transitional DTD に従い有効化します。類似したエントリを含む**%AltovaCommonSchemasFolder%\Schemas\xhtml** フォルダー内の**catalog.xml** ファイルを参照してください。

XML スキーマ仕様

XML スキーマ仕様情報はXMLSpy にビルトされておりXML スキーマ（`.xsd`）ドキュメントの妥当性はこの内部情報に対してチェックされます。XML スキーマドキュメント内では、このためレファレンスはXML スキーマ仕様を定義するスキーマになってはなりません。

`%AltovaCommonSchemasFolder%\Schemas\schema` フォルダー内の**catalog.xml** ファイルは古いXML スキーマ仕様を実装するDTD へのレファレンスを含んできます。XML スキーマドキュメントをこれらのスキーマに対して検証してはなりません。参照されるファイルはドキュメントをこれらの古い奨励に従い作成する場合に編集の目的でXMLSpy に入力ヘルパー情報を与えるために含まれています。

6.6.4 環境変数

nextCatalog 要素ではシェル環境変数が使用され、システムに関係する場所へのパスが参照されます(*上の RootCatalog.xml の内容を参照*)。以下のシェル環境変数がサポートされます:

%PersonalFolder%	現在のユーザーの個人的なフォルダーへのフルパス、例えば C:\Users\<name>\Documents
%CommonSchemasFolder%	C:\ProgramData\Altova\Common2021\Schemas
%ApplicationWritableDataFolder%	C:\ProgramData\Altova
%AltovaCommonFolder%	C:\Program Files\Altova\Common2021
%DesktopFolder%	現在のユーザーのデスクトップフォルダーへのフルパス。
%ProgramMenuFolder%	現在のユーザーのプログラムメニューフォルダーへのフルパス。
%StartMenuFolder%	現在のユーザーの開始メニューフォルダーへのフルパス。
%StartUpFolder%	現在のユーザーのスタートフォルダーへのフルパス。
%TemplateFolder%	現在のユーザーのテンプレートフォルダーへのフルパス。
%AdminToolsFolder%	現在のユーザーの管理ツールを保管するファイルシステムディレクトリへのフルパス。
%AppDataFolder%	現在のユーザーのアプリケーションデータフォルダーへのフルパス。
%CommonAppDataFolder%	全てのユーザーのアプリケーションデータを含むファイルディレクトリへのフルパス。
%FavoritesFolder%	現在のユーザーのお気に入りフォルダへフルパス。
%PersonalFolder%	現在のユーザーの個人フォルダーへのフルパス。
%SendToFolder%	現在のユーザーの送信フォルダーへのフルパス。
%FontsFolder%	システムフォントフォルダーへのフルパス。
%ProgramFilesFolder%	現在のユーザーのプログラムファイルフォルダーへのフルパス。
%CommonFilesFolder%	現在のユーザーの共通ファイルフォルダーへのフルパス。
%WindowsFolder%	現在のユーザーの Windows フォルダーへのフルパス。

%SystemFolder%	現在のユーザーのシステムフォルダーへのフルパス。
%LocalAppDataFolder%	ローカル（非ローミング）アプリケーションのためのデータレポジトリとしての役割を果たすファイルシステムディレクトリへのフルパス。
%MyPicturesFolder%	ピクチャフォルダーへのフルパス。

6.7 スキーマの中を検索

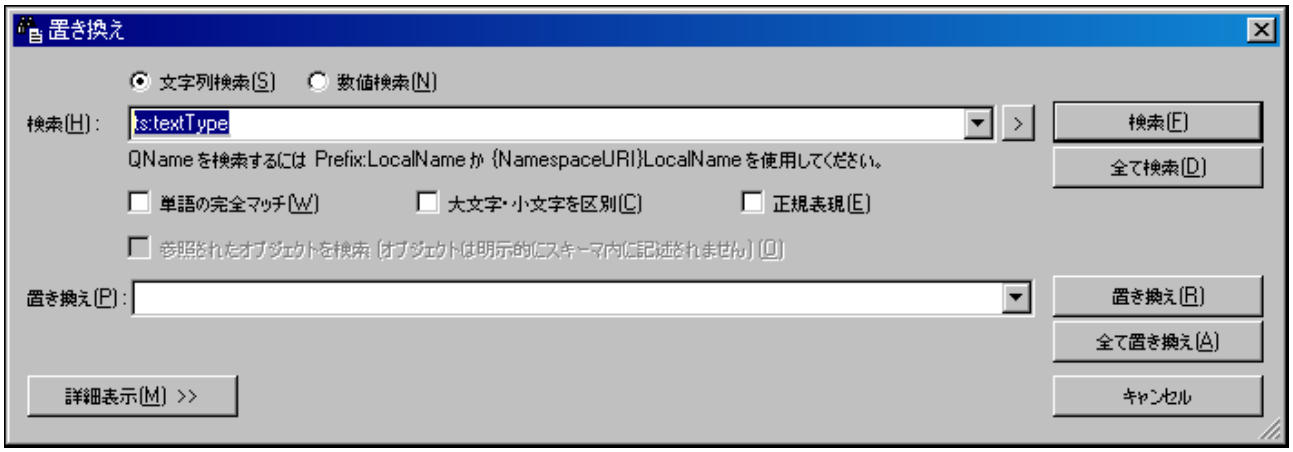
XMLSpy の検索と置き換え機能をスキーマビューで使うことで、インテリジェントな検索を行うことができます。

スキーマビューの検索と置き換え機能は、スキーマファイルがスキーマビューにてアクティブになっている時に使用することができます。以下の2つの方法でこの機能にアクセスすることができます:

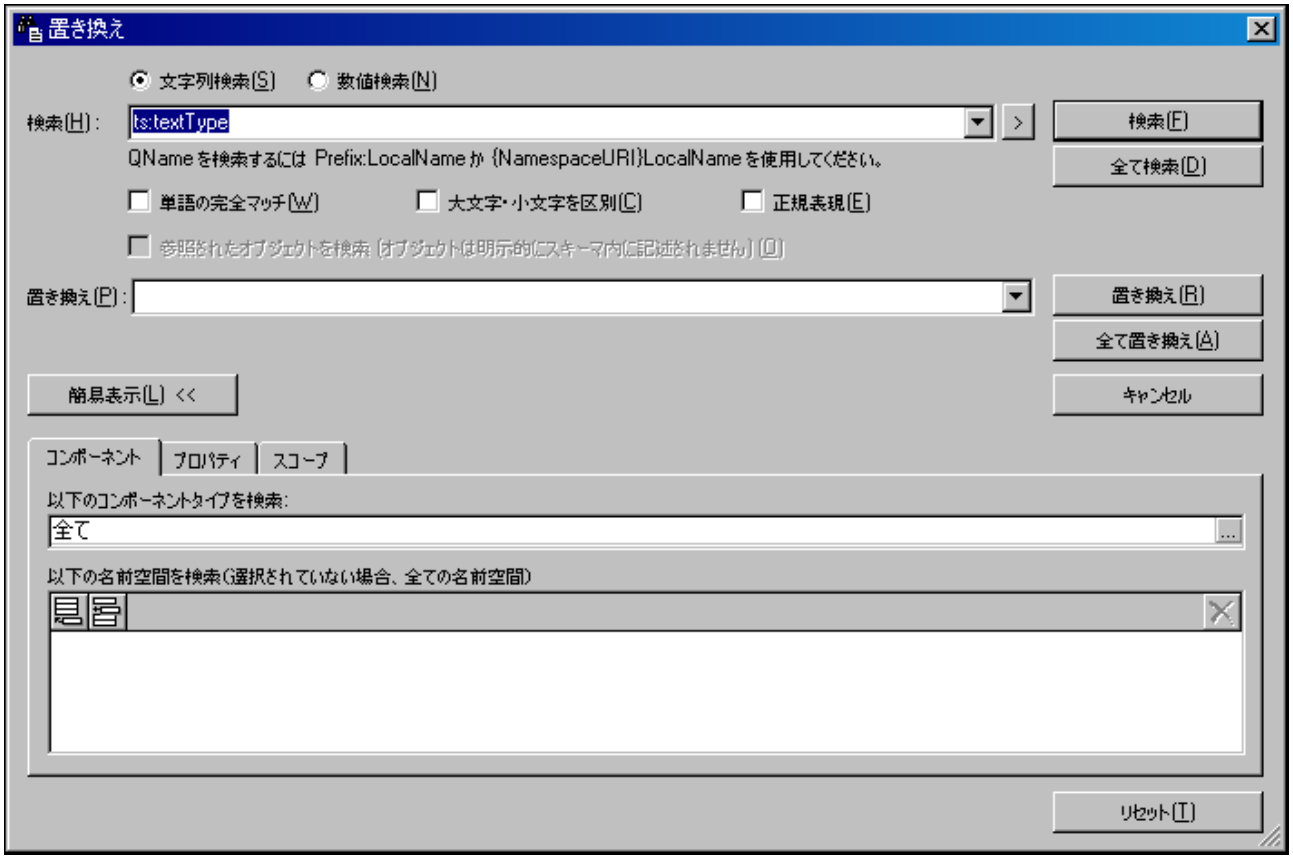
- メニューコマンドから「**編集 | 検索**」や「**編集 | 置換**」を選択する
- スキーマの中を検索ウィンドウにある「**検索**」または「**置き換え**」ボタンを押下する

コマンドを選択またはボタンをクリックすることで、検索または置き換えダイアログが表示されます。検索ダイアログと比べると、(*以下のスクリーンショットにある*) 置き換えダイアログには置き換えを行うテキストを表示するテキストフィールドが追加されています。

デフォルトの置き換えダイアログは以下のように表示されます:



「**詳細表示**」ボタンをクリックすることで、(*以下のスクリーンショットにあるように*) より詳細な検索を行うための条件が表示されます。



使用方法は以下のとおりです:

- 検索ならびに置き換えテキストフィールドにて検索、置き換えを行う単語を入力[417]
- コンポーネントタブにて検索するスキーマコンポーネントを指定[418]
- 検索するコンポーネントのプロパティを指定[419]して検索の対象を狭める
- 現在のドキュメント、プロジェクト、またはフォルダーを指定して検索の範囲をセット[422]
- コマンドの実行[423]
- スキーマの内を検索ウィンドウを使用[425]してコンポーネントの素早いナビゲーションを行う

ダイアログ下部にある「リセット」ボタンにより、以下にあるようなデフォルト状態に設定を戻します:

- 検索語無し、置き換え語無し
- コンポーネント: 全て
- 名前空間: 選択無し
- プロパティの制約: どこでも
- プロパティの追加制約: 無し
- スコープ: 現在のファイル

メモ 置換フィールド内で正規表現はサポートされていません。

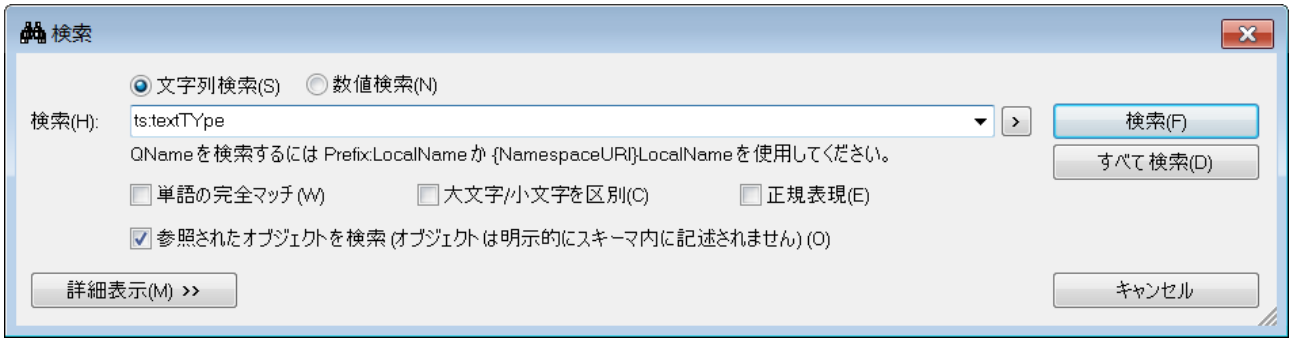
(C) 2015-2021 Altova GmbH

6.7.1　検索語

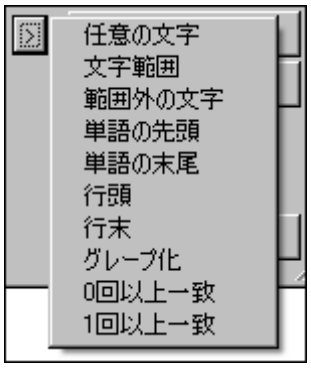
検索語は文字列(**文字列ラジオボタン**)または数値(**数値ラジオボタン**)として入力することができます。

文字列検索

文字列検索内で(*下のスクリーンショット*)エントリは以下であることができます: (I) テキスト (ii) QName または(iii) 正規表現QName を使った検索では、テキストフィールドに入力されたドキュメントで使用されているプレフィックス、もしくは名前空間 URI をベースに名前空間が決定されます。以下のスクリーンショットでは、`ts:`プレフィックスが名前空間を決定するためにドキュメント内で使用されているプレフィックスになります。



正規表現を使った検索を行うには、正規表現チェックボックスにチェックを入れて、テキストフィールドに正規表現を入力してください。(*以下のスクリーンショットに示される*) 検索語入力フィールドの右にある矢印ボタンをクリックすることで表示されるメニューから正規表現の入力を行うことができます。



検索語がドキュメント内の単語に完全一致するか、または大文字/小文字を区別した検索を行うか設定することができます。テキストフィールドの下にあるチェックボックスを使って、これらのオプションを使用してください。

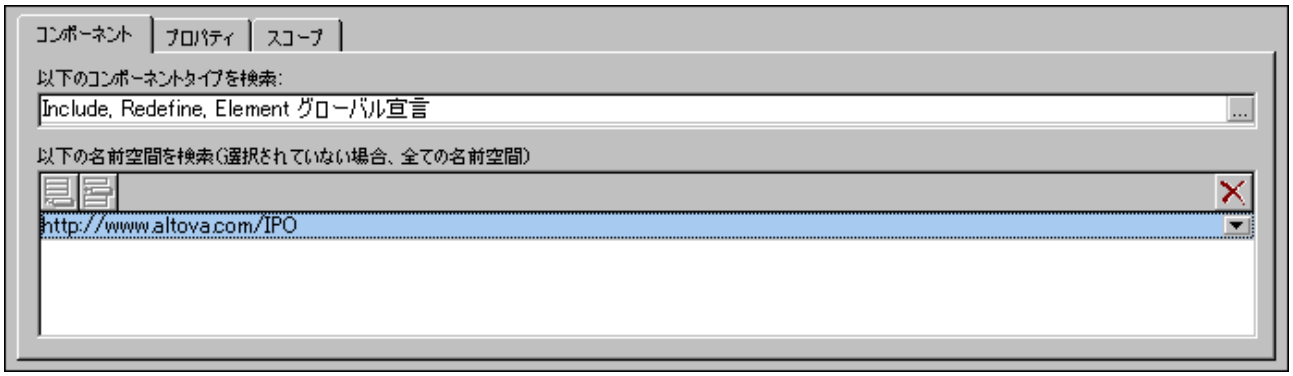
(複合型の定義やグローバル要素といった) 参照されているオブジェクト内で検索を行ないたい場合は、参照されたオブジェクトを検索チェックボックスのチェックを入れてください。このオプションは検索ダイアログだけで使用できるもので、置き換えダイアログでは無効化されます。

数値検索

数値検索ラジオボタンが選択されていれば、1 つの演算子と数値からなる検索条件か、2 つの検索条件を `AND` または `OR` の論理式により組み合わせることで検索を行うことができます。以下のスクリーンショットでは、2 つの検索パラメーターにより、「1 以上かつ 5 以下」という検索条件が指定されています。

6.7.2 コンポーネント

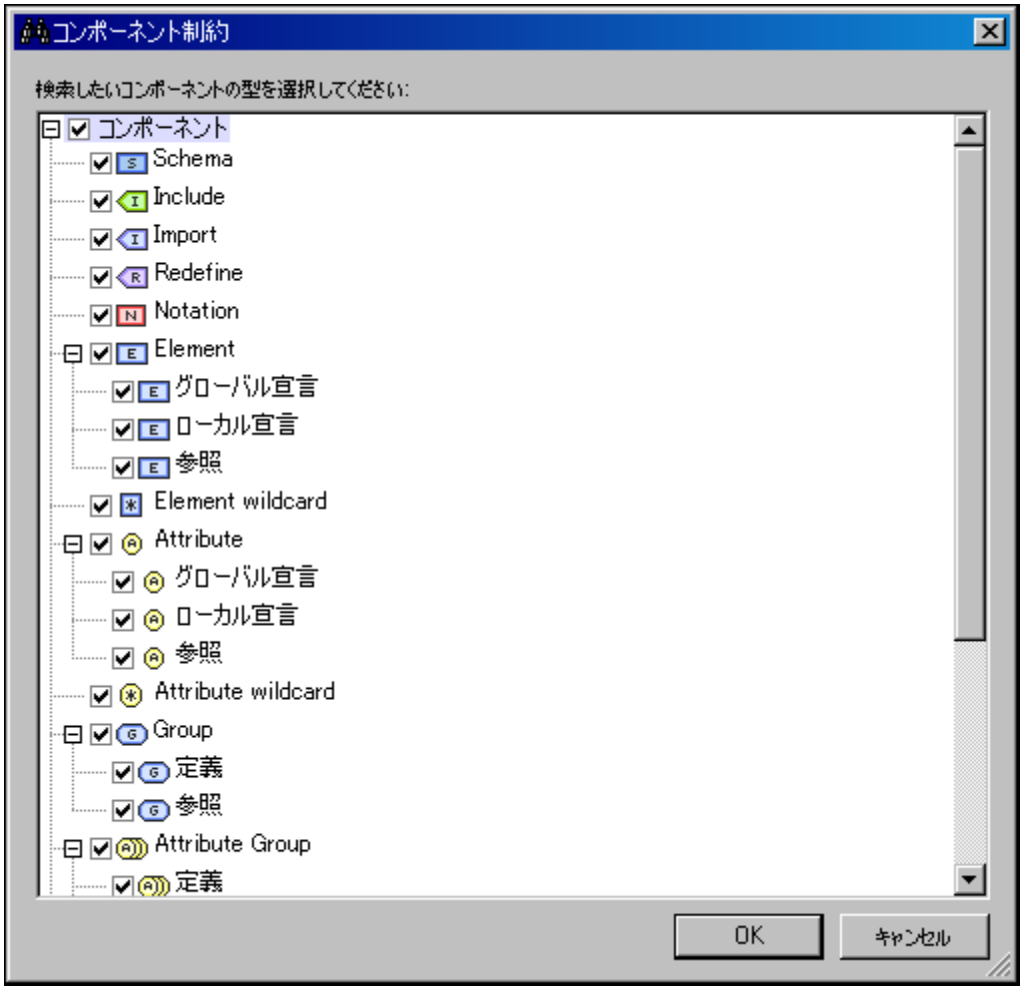
コンポーネントの種類やターゲット名前空間により検索を行う対象を制限することができます。これらのオプションはコンポーネントタブから利用することができます。検索または置き換えダイアログにて「**詳細表示**」ボタンをクリックしてください。検索の対象を狭めるタブが開かれ、そのうちの1つが(*以下のスクリーンショットに示される*)コンポーネントタブになります。



コンポーネントタブは2つの部分から構成されています: コンポーネントタブは、(i) 検索を行うコンポーネントの種類と、(ii) 検索を行うターゲット名前空間という2つの部分から構成されます。

コンポーネントの選択

テキストフィールドの右側にある **追加** アイコンをクリックすることにより検索されるコンポーネント型を入力することができます。 ... (*上のスクリーンショットを参照*)。この操作により(*以下のスクリーンショットに示される*)コンポーネント制約ダイアログが表示され、クリック操作で検索を行う対象コンポーネントの種類を選択することができます。リストの上の `Components` アイテムをチェックすると全てのコンポーネントが選択されます(テキストエントリ `all`)。チェックを解除すると選択されたコンポーネントのそれぞれが含まれる全てのコンポーネント選択が解除されます(テキストエントリ `none`)—。従って、各コンポーネントを個々に選択するには、コンポーネント アイテムの選択が解除されていなければなりません。選択されたコンポーネントはカンマ区切りでテキストフィールドに表示されます(*上のスクリーンショットを参照*)。



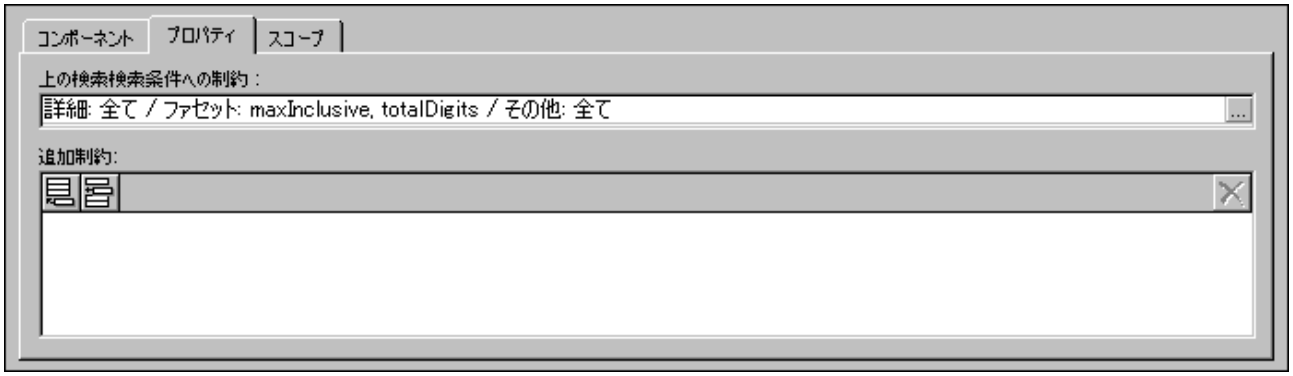
注意:　コンポーネントタブまたは検索、置き換えダイアログが再度開かれたとしても、前回のコンポーネント選択状態は保持されたままになります。

名前空間の選択

検索の対象とするターゲット名前空間を設定するには、「**追加**」または「**挿入**」アイコンをクリックして、目的の名前空間を入力してください。ターゲット名前空間が指定されていない場合、全てのターゲット名前空間が選択されます。ペインに入力されたターゲット名前空間を削除するには、そのターゲット名前空間を選択して、「**削除**」アイコンをクリックしてください。

6.7.3　プロパティ

プロパティタブのオプションを使用することで、コンポーネントプロパティ(詳細ならびにファセット)や、プロパティの内容で検索結果を絞り込むことができます。「**詳細表示**」ボタンをクリックすることで検索または置き換えダイアログが展開表示されるので、プロパティタブを選択します(*以下のスクリーンショットを参照*)。



プロパティタブは2つの部分から構成されています: (i) (メインの検索テキストフィールドに表示される) 検索対象の制約と、(ii) コンテンツ制約 (*追加の制約*[421] *を参照してください。*)

プロパティの選択

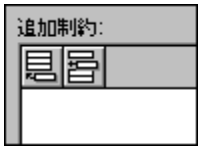
テキストフィールドの右側にある **追加** アイコンをクリックすることにより検索されるプロパティ型を入力することができます。 [...] (*上のスクリーンショットを参照*)。この操作により(*以下のスクリーンショットに示される*) プロパティ制約ダイアログが表示され、クリック操作で検索を行う対象コンポーネントの種類を選択することができます。プロパティ制約ダイアログが表示され、アイテムをクリックすることで検索を行うプロパティを選択することができます。プロパティは(i) 詳細、(ii) ファセット、(iii) その他の3 つのグループに分類されています。詳細、ファセット、またはその他をクリックすることで、グループにある全てのプロパティが選択されます。グループを再度クリックすると、そのグループのプロパティ全てが(個別に選択されたプロパティも含めて) 選択解除された状態になります。個々のプロパティはグループアイテムが選択されていない状態でないと選択することができません。選択されたコンポーネントはカンマ区切りでテキストフィールドに表示されます(*上のスクリーンショットを参照*)。

注意: プロパティタブまたは検索、置き換えダイアログが再度開かれたとしても、前回のプロパティ選択状態は保持されたままになります。

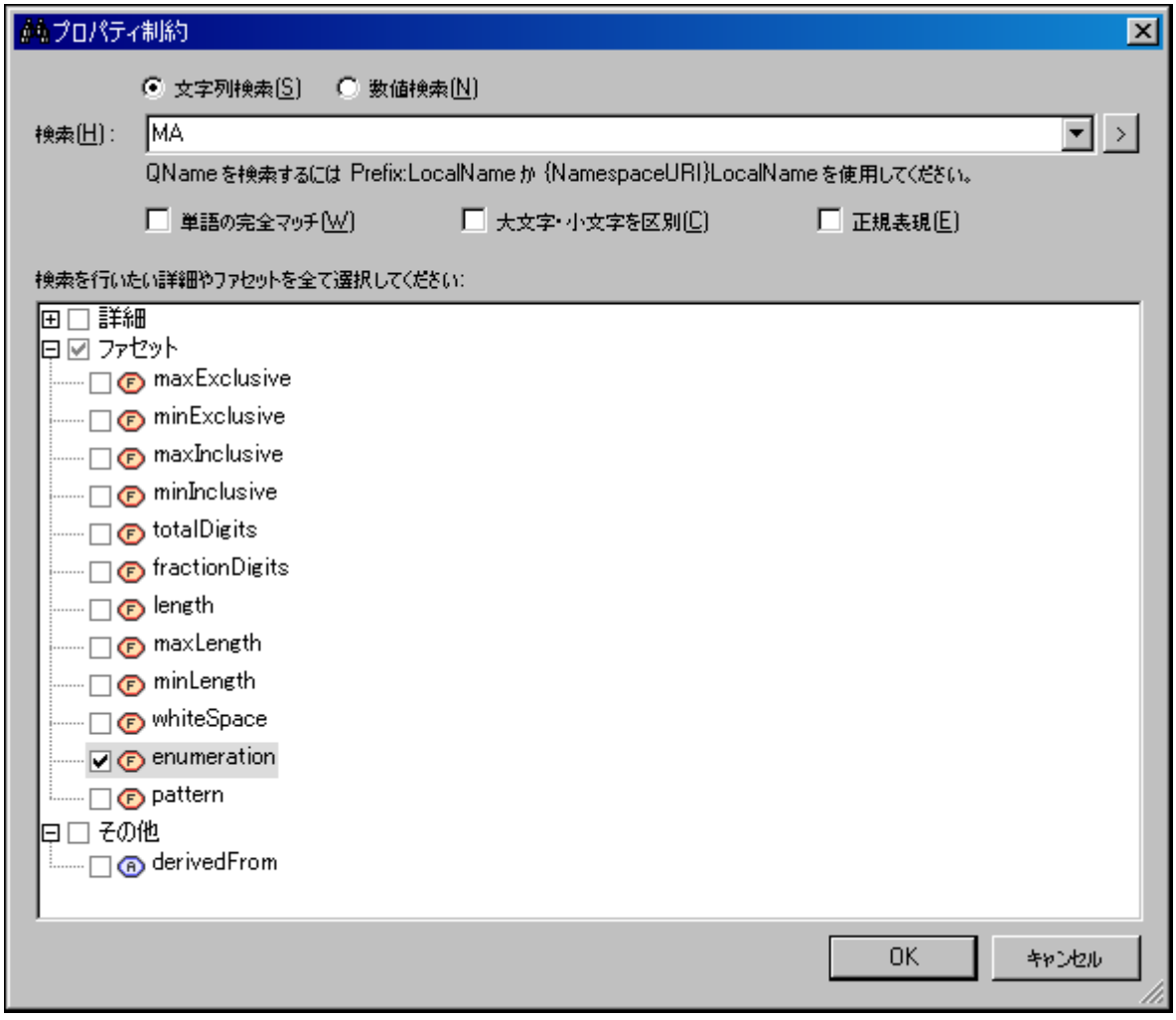
追加制約

追加制約により検索を行うプロパティの持つ値を指定することができます。例えば、列挙の`MA`（米国のマサチューセッツ州）という値を持つ`state`という名前の要素を探している場合、`enumeration`と`MA`という値を追加制約オプションで指定することで希望する検索を行うことができます。以下のような操作を行ないます:

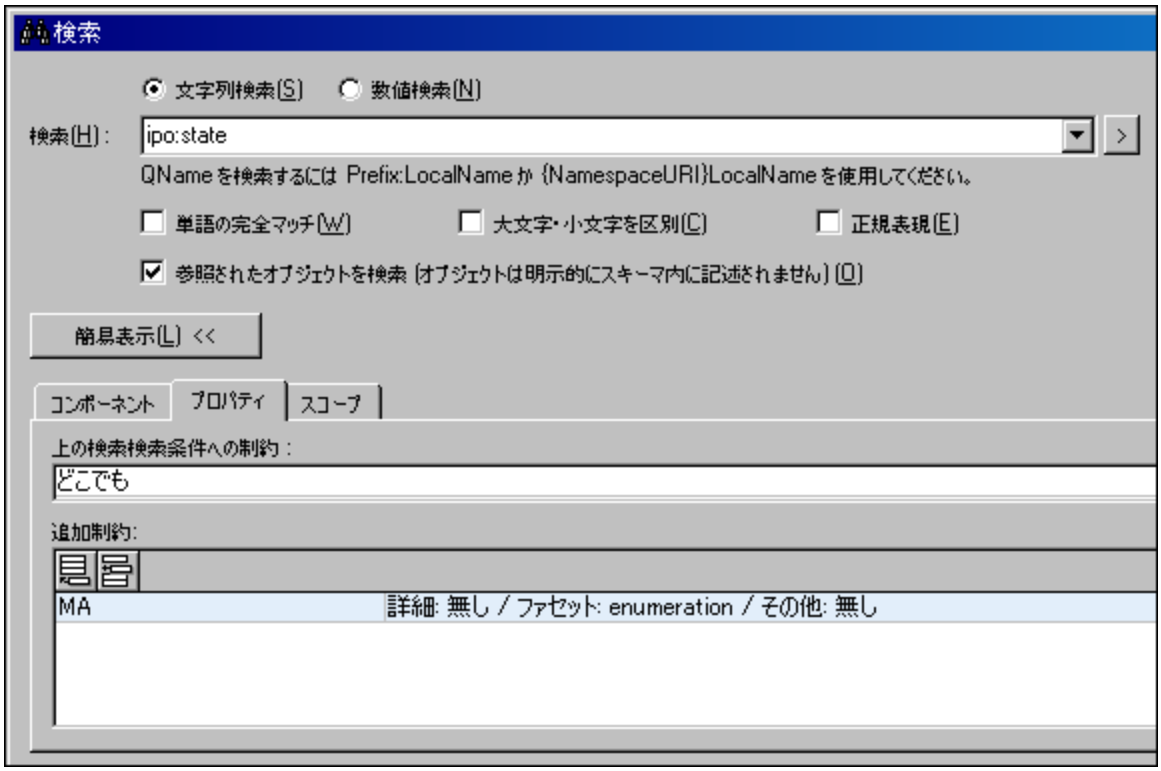
1. 追加制約ペインにて「**追加**」または「**挿入**」アイコンをクリックします（*下のスクリーンショット*）。



2. ペインに新たな行が追加され、プロパティ制約ダイアログが表示されます。全てのアイテムを選択解除して、`enumeration`プロパティのみを洗濯します（*下のスクリーンショット*）。



3. ダイアログの上部にあるテキストフィールドに、検索を行う列挙の値（この例では`MA`を入力します（*上のスクリーンショットを参照してください*）。
4. 「OK」をクリックします。（*以下のスクリーンショットにあるように*）追加制約ペインにて新たに作成された行に制約が入力されます。



上で示されるスクリーンショットにて検索語が `ipo:state` となっていることにご注目してください。プロパティタブにある「どこでも」により、全てのプロパティが検索の対象となりますが、追加制約により検索は `MA` という値を持つ列挙に制限されます。

複数の追加制約により、検索結果を更に狭めることができます。追加制約を削除するには、追加制約を選択して**削除**アイコンをクリックしてください。

注意: プロパティタブまたは検索、置き換えダイアログが再度開かれたとしても、前回の追加制約選択状態は保持されたままになります。

6.7.4 スコープ

検索の範囲を(以下のスクリーンショットにある)スコープタブにより設定することができます。検索範囲をファイル、またはスキーマビューで現在選択されているスキーマコンポーネントから選択することができます。

ファイルオプションが選択されている場合、更に以下のオプションから選択範囲の指定を行うことができます:

- **現在のファイル**：現在アクティブなファイルが検索範囲になり、インクルード、インポート、再定義されたファイルも検索対象に含めるオプションが表示されます。
- **開いたファイル**：XMLSpy で開かれている全ての XML スキーマ（XSD）ファイルが検索範囲になります。全て検索ならびに全て置き換えコマンドのみが有効になり、通常の検索を使用することはできません。
- **プロジェクト**：現在アクティブなプロジェクトが検索範囲になり、外部フォルダーをスキップするオプションが表示されます。全て検索ならびに全て置き換えコマンドのみが有効になり、通常の検索を使用することはできません。（**「ツール | オプション | ファイルの種類 | デフォルトビュー」**で設定することができる）`.xsd` ファイル拡張子のデフォルトビューがスキーマビューでない場合、`.xsd` ファイルは検索対象となりません。
- **フォルダー**：検索範囲となるフォルダーを指定することができ、サブフォルダーを含めるオプションが表示されます。全て検索ならびに全て置き換えコマンドのみが有効になり、通常の検索を使用することはできません。（**「ツール | オプション | ファイルの種類 | デフォルトビュー」**で設定することができる）`.xsd` ファイル拡張子のデフォルトビューがスキーマビューでない場合、`.xsd` ファイルは検索対象となりません。
- インクルード/インポート/再定義されたファイルのオプションをチェックすることで、これらのファイルを検索範囲に含めることができます。

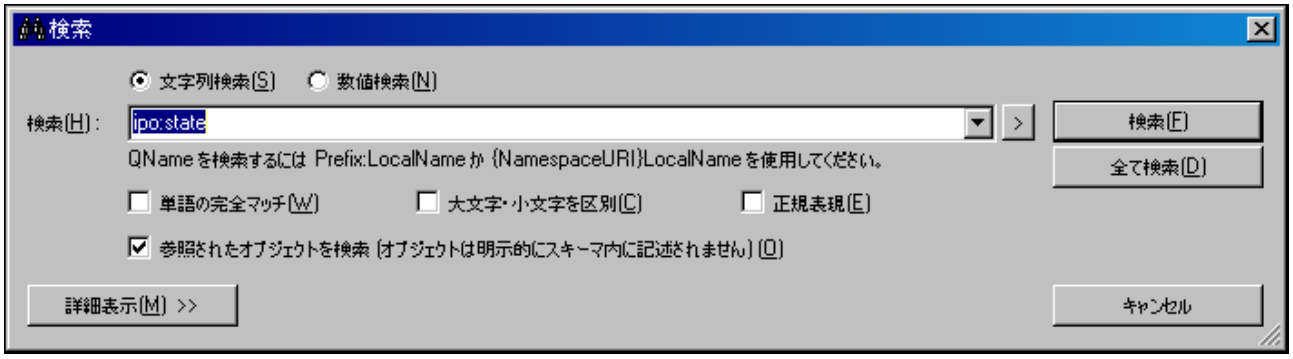
置き換えダイアログでは、ディスク上のファイルを直接置き換えるか、ファイルを XMLSpy にて開くかを選択することができます。ダイアログにあるボタンから選択してください。

6.7.5　検索と置き換えコマンド

検索ならびに置き換えダイアログ内の**「検索」**コマンドの動作はこれまでの検索方法とは多少異なっています。ここでは両ダイアログの**「検索」**コマンドならびに**「置き換え」**コマンドの解説を行ないます。

検索ダイアログ

検索語、そして必要な場合はその他の条件を入力した後、**「検索（F3）」**または**「全て検索」**コマンドをクリックします（*下のスクリーンショット*）。



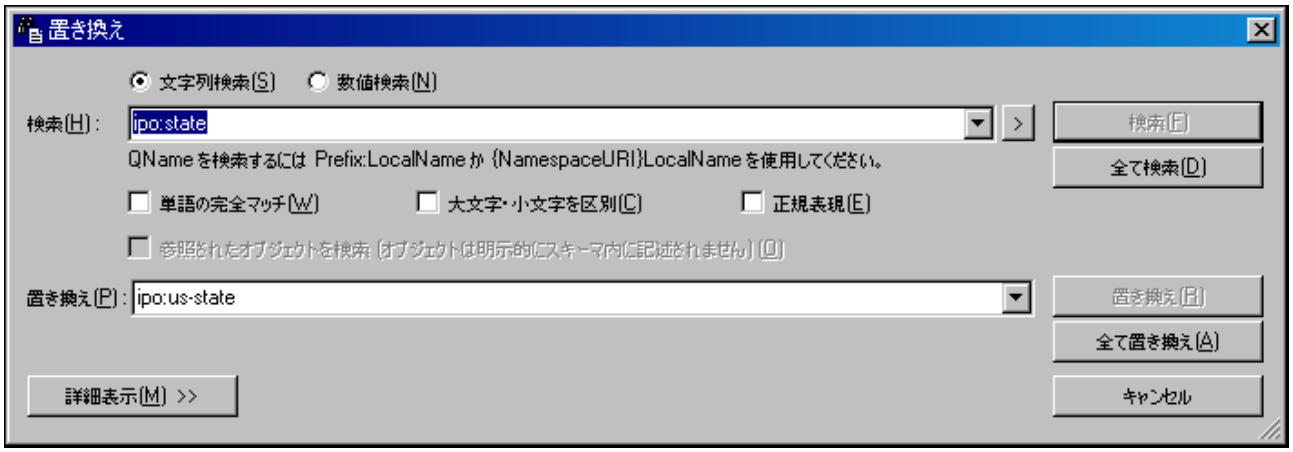
ダイアログにある「**検索** (Ctrl+F)」コマンドをクリックすることにより検索ダイアログが閉じられ、(検索条件や検索語という) 検索条件にマッチする検索範囲内の次の出現場所が特定されます。スキーマビューで現在選択されているコンポーネントの位置を基準にして、次の検索結果は求められます。検索範囲の最後まで検索が行われても、検索範囲の最初から自動的に検索が続けられることはありません。そのため、検索を始める前に、現在選択されているコンポーネントの位置が検索を行ないたい箇所の前にあることを確認する必要があります。

「**検索**」の結果はスキーマビューにてハイライトされ、スキーマの中を検索ウィンドウにもその結果が表示されます。スキーマの中を検索ウィンドウにある結果をクリックして、スキーマビューの対応するアイテムをハイライトさせることもできます。

「**全て検索**」コマンドをクリックすると、検索ダイアログが閉じられ、検索結果の全てがスキーマの中を検索ウィンドウに表示されます。

置き換えダイアログ

(*下のスクリーンショット*) 置き換えダイアログでは「**検索**」コマンドをクリックすることで、スキーマビューで現在選択されている箇所から、次の検索語の出現箇所を特定します。「**置き換え**」をクリックすることで、この出現を置き換えることができます。



「**全て検索**」コマンドを使うと、置き換えダイアログが閉じられ、検索の結果全てがスキーマの内を検索ウィンドウに表示されます。

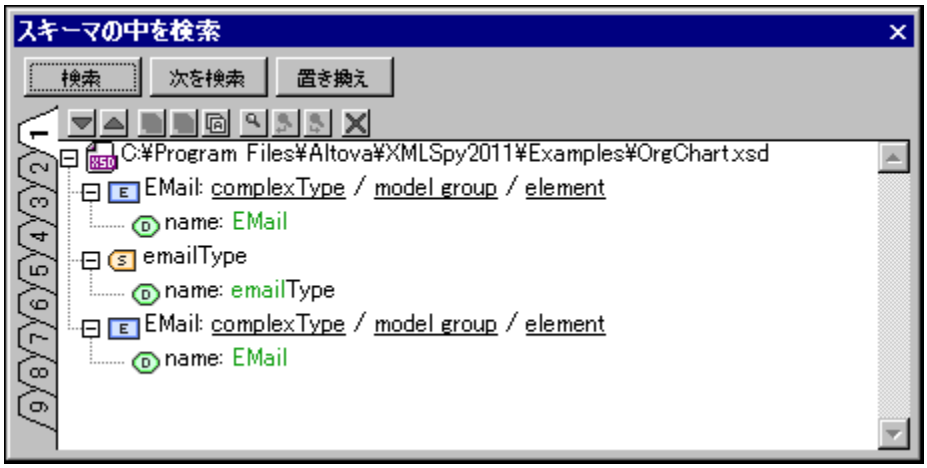
「**全て置き換え**」コマンドは、検索により見つかった箇所全ての置換えを行ないます。

メモ 置換フィールド内で正規表現はサポートされていません。

(C) 2015-2021 Altova GmbH

6.7.6　結果と情報

「検索」、「全て検索」、「置き換え」、または「全て置き換え」コマンドが実行されると、コマンド実行の結果が、以下のスクリーンショットにあるように、スキーマの中を検索ウィンドウに表示されます。検索された単語は緑色で表示されます（以下のスクリーンショットの例だと検索語の `email` が緑色で表示されます）。スキーマファイル内の位置も表示されていることにご注目してください。



「全て検索」または「全て置き換え」コマンドを使うと、ドキュメント内の出現箇所全てが表示されます。

メモ:　ウィンドウの上部にある検索や置き換えボタンを使えば、それぞれ検索、置き換えダイアログを表示させることができます。「次を検索」ボタンを使えば、検索語が次に出現する箇所を特定することができます。

スキーマの内を検索ウィンドウの機能

検索などの結果は（1 から 9 までの番号が振られた）9 つの異なるタブの中に表示されます。例えばある検索結果を 1 つのタブに収めておき、別のタブで新たな検索をすることにより結果の比較を行うことができます。スキーマの中を検索ウィンドウの結果をクリックすることで、メインウィンドウのスキーマビューに表示されている関連するコンポーネントがハイライトされます。これらのナビゲーション機能により、目的のコンポーネントへ素早くたどり着くことができます。

スキーマの中を検索ウィンドウでは以下のツールバーコマンドを利用することができます:

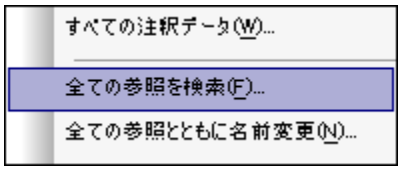
- 「次」ならびに「前」アイコンにより、現在選択されているメッセージの次（もしくは前）にあるメッセージが選択されます。
- 「メッセージをコピー」コマンドにより、選択されたメッセージ、子を含む選択されたメッセージ、そして全てのメッセージをクリップボードにコピーします。
- 「検索」コマンドによりスキーマの中を検索ウィンドウ内にあるテキストを検索します。
- 「クリア」コマンドにより、現在アクティブなタブのメッセージが全て消去されます。

6.7.7　グローバルコンポーネントの検索と名前変更

ユーザーは、選択されたファイルの XML スキーマや、関連する全てのスキーマファイル内にある名前付きグローバルコンポーネントの検索ならびに名前変更を行うことができます。名前を持つグローバルコンポーネントは以下を除いてすべてグローバルコンポーネントです: Include, Import, Redefine, Annotation, Comment, and PI components

このプロセスの仕組みは以下のとおりです:

1. スキーマ概要にて検索または名前の変更を行うグローバルコンポーネントを選択します。
2. 選択されたコンポーネントを右クリックすることで表示されるコンテキストメニューから、目的のコマンド（「**全ての参照を検索**」または「**全ての参照とともに名前変更**」）を選択します。

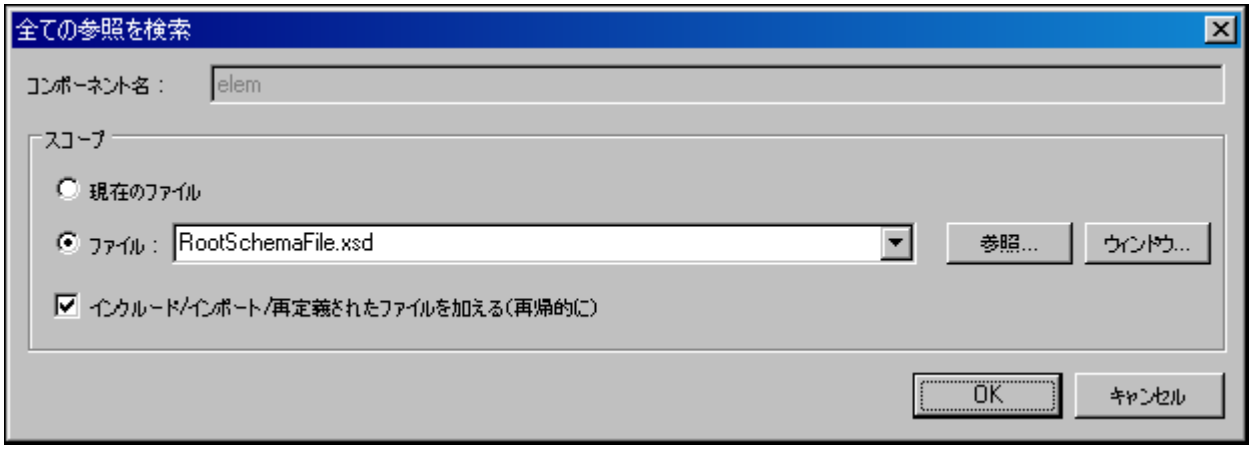


3. 表示されるダイアログにて、検索を行う範囲（または名前変更オペレーション）を選択します。名前変更オペレーションの場合、新たなグローバルコンポーネントの名前を入力します。
4. 「OK」をクリックすると検索の結果はスキーマウィンドウ内の検索内に表示されます（*下のスクリーンショットを参照してください*）。

グローバルコンポーネントへの参照を行っているファイルの場所が表示されます（*上のスクリーンショットを参照*）。名前の変更が行われたコンポーネントも全て表示されます。

全ての参照を検索

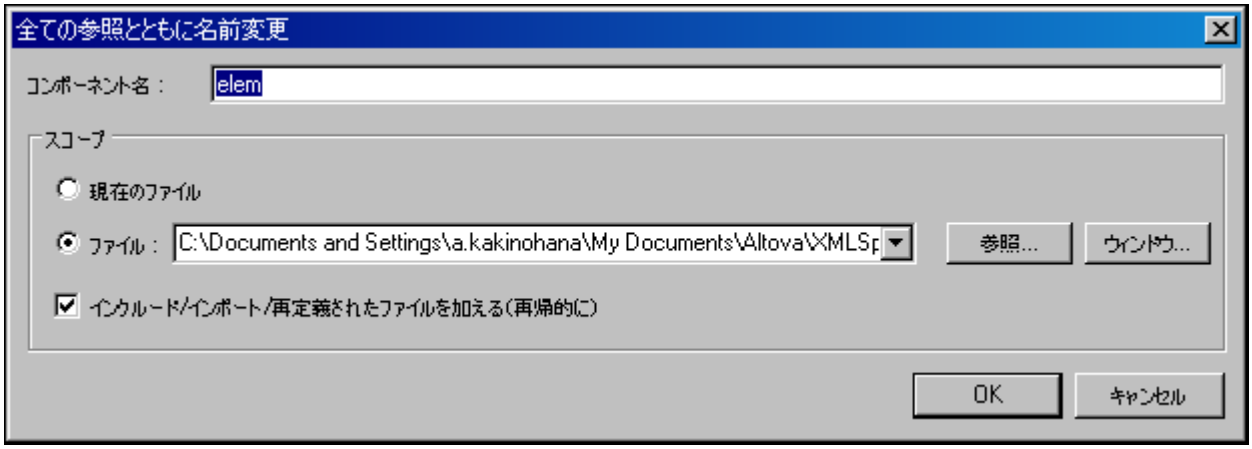
全ての参照を検索ダイアログ（*下のスクリーンショット*）を開くには以下を行います: **全ての参照を検索**ダイアログ（以下のスクリーンショットを参照）を開くには、(i) スキーマ概要にてグローバルコンポーネントを右クリックし、(ii) コンテキストメニューから、「全ての参照を検索」コマンドを選択します。



グローバルコンポーネント名が*コンポーネント名*フィールドにグレーで表示されます(編集は行えません)。検索の範囲対象を現在のファイルで行うか、または参照により選択したファイル内で行うかを指定することができます。ダイアログ下部にある*参照されているスキーマファイルの追加*チェックボックスを有効にすることで、(Include、Import、Redefine により)関連するファイルを検索の対象に加えることもできます。

全ての参照とともに名前変更

グローバルコンポーネントの名前を変更するには、コンポーネントを右クリックし、表示されるコンテキストメニューから「**全ての参照とともに名前変更**」を選択します。全ての参照とともに名前変更ダイアログが表示されます(*以下のスクリーンショットを参照*)。



選択されたグローバルコンポーネントに対して、新たに使用したい名前を*コンポーネント名*テキストフィールドに入力します。検索ならびに名前変更の範囲対象を、現在のファイルにするか、または参照により選択されたファイル内で行うかを指定することができます。ダイアログ下部にある*参照されているスキーマファイルの追加*チェックボックスを有効にすることで、(Include、Import、Redefine により)関連するファイルを検索の対象に加えることもできます。

7 XSLT

Altova ウェブサイト : XSLT エディター

このセクションでは以下のような構成でXSLT の説明を行ないます:

- XSLT ドキュメントの編集[429] : XMLSpy におけるXSLT ドキュメントの編集機能について説明します。
- XSLT 処理[431] : 選択されたエンジンを使って XMLSpy GUI からXSLT 変換を行う方法について解説します。このセクションではXMLSpy で行うことができるXSLT の設定についても説明されます。
- XSL アウトライン[433] : 複数のXSL ファイルの閲覧、ナビゲーション、そして管理を行うのに便利な XSL アウトラインならびに XSL 情報ウィンドウについて説明します。

XPath の評価

XML ドキュメントがアクティブになっている際、XPath ウィンドウ[113] を使用することで、XPath 条件式の評価を行うことができます。この機能により、XPath 条件式が評価される様子を素早く確認することができます。XPath 条件式を入力すると、ドキュメントルートまたはXML ドキュメント内で選択されたコンテキストノードに相対的な評価が行われます。条件式評価の結果は、すぐにXPath ウィンドウにて表示されます。XPath ウィンドウの使用方法については、イントロダクション | XPath ウィンドウ[113] のセクションを参照ください。

XSLT デバッガーとプロファイラー

XMLSpy にはXSLT プロファイラー[492] ならびにXSLT デバッガー[467] が搭載されており、正確で効率のよいXSLT スタイルシートを素早く作成することができます。これらの機能はXSLT とXQuery[428] セクションにて記述されます。

その他のXSLT 機能

このセクションで記述される様々な機能のより詳しい情報については、ユーザーリファレンスの関連するメニューコマンド[1277] を参照ください。

Altova XSLT エンジン

Altova XSLT 1.0、2.0 ならびに3.0 エンジンについては付録[1973] のXSLT とXQuery エンジンの情報[1974] を参照ください。

RaptorXML を使ったコマンドラインやバッチ処理

XMLSpy GUI のプロジェクト機能を使用して、バッチ処理を行うことができます。より柔軟性のある処理を求める場合は、速いXML 検証 XSLT 変換、XQuery 実行機能を搭載したAltova のRaptorXML 製品を使用してください。をお使いください。コマンドラインやバッチ処理によりXSLT 変換を行うのにRaptorXML は最適のツールです。

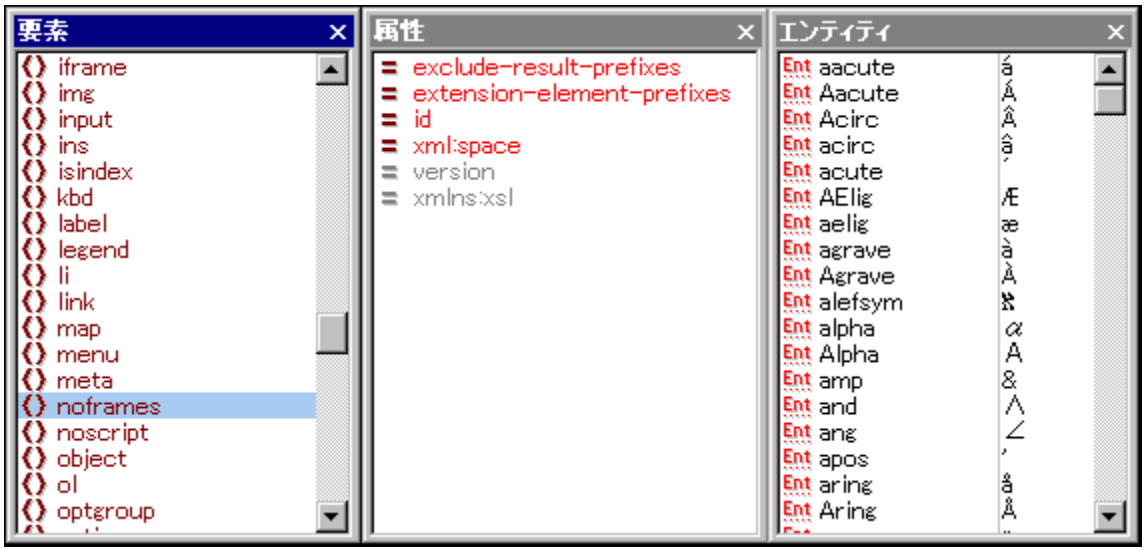
(C) 2015-2021 Altova GmbH

7.1 XSLT ドキュメント

XSLT 1.0、2.0、および3.0 ドキュメントは、その他のXML ドキュメントと同じようにテキストビュー[130]やグリッドビュー[149]にて編集することができます。ドキュメントが開かれるデフォルトのビューは、オプションダイアログのファイル型のセクションにて設定することができます。

入力ヘルパー

要素、属性、そしてエンティティ入力ヘルパーを利用することができます。入力ヘルパーにて表示されるアイテム情報はXMLSpy に内蔵されており、XSLT ドキュメント内で行われる参照とは独立したものです。



以下の点に注意してください:

1. 新規 XSLT ドキュメントが「**ファイル | 新規作成**」メニューの新規ドキュメント作成ダイアログにて作成されたとき、(XSLT 1.0、XSLT 2.0、またはXSLT 3.0 により作成されたドキュメントの種類によって違いはありますが) 適切な XSLT 要素と属性が入力ヘルパーにロードされます。それに加えてHTML 要素や属性、HTML 2.0 エンティティセット、Latin-1、特殊な文字、そしてシンボルがロードされます。
2. 「**ファイル | 新規作成**」の新規ドキュメントダイアログにてXML ドキュメントが作成され、XSLT コンテンツがそこから作成された場合、XML 文字エンティティ以外のアイテムは入力ヘルパーに表示されません。
3. 新規ドキュメントダイアログを通して作成されたXSLT ドキュメントが開かれた場合、上の1番目の様に、入力ヘルパーにアイテムが表示されます。
4. 新規ドキュメントダイアログにてXSLT ドキュメントとして作成されなかったXSLT ドキュメントが開かれた場合、上の1番目の様に入力ヘルパーにXSLT アイテムが表示されます。更に、XSL-FO 要素や属性がテキストビュー入力ヘルパーに表示されます。
5. 要素入力ヘルパーにある要素のプレフィックスは以下の様に固定されています: XSLT 要素の`xsl:`、HTML 要素のプレフィックス無し、XSL-FO 要素の`fo:`。入力ヘルパーを使用するには、入力ヘルパーに表示される内蔵のプレフィックスに合致させる形で、XSLT ドキュメントで宣言された名前空間にもプレフィックスが記述されている必要があります。

自動補完

テキストビューでは入力中に自動補完のポップアップが表示され、入力されたテキストにマッチした最初のアイテムがハイライトされます。要素が入力されている時、要素のリストがポップアップされ、アルファベット順で最も近いアイテムがハイライトされます。同様に、属性が入力されているとき、適用することのできる属性のリストがポップアップで表示されます。入力ヘルパーの以前のセクションで記述されたルールに従ってリストに表示されるアイテムは決定されます。

XPath インテリジェント編集

XSLT ドキュメントにて XPath 条件式を入力することができる場合（例えば、select 属性の値など）、インテリジェントな XPath 編集機能を利用することができます。

- 入力中にかっこを一致させるなどの、XPath のための構文の色分けをビルドします。
- XPath 関数にカーソルをポイントすると、関数に関する情報が含まれるヒントが表示されます。
- XPath 関数と軸は、入力中にポップアップ内で提示されます。上/下 カーソルを使用して、提示されたリストを上下に移動することができます。ポップアップ内で関数がハイライトされる場合、関数（署名）に関する情報が追加ポップアップ内に表示されます。
- XML ファイルが情報ウィンドウ[437]に割り当てられている場合、XML ファイルの要素ならびに属性がポップアップに表示されます。

XSLT ドキュメントの検証

XMLSpy に内蔵されている XSLT スキーマに対して XSLT ドキュメントの検証を行うことができます（「XML | XML 検証(F8)」をクリックします）。XSLT ドキュメントが XSLT 1.0、 2.0 および 3.0 （`xsl:stylesheet` 要素の `version` 属性により指定されます）により、適切な内蔵スキーマが選択されます。

(C) 2015-2021 Altova GmbH

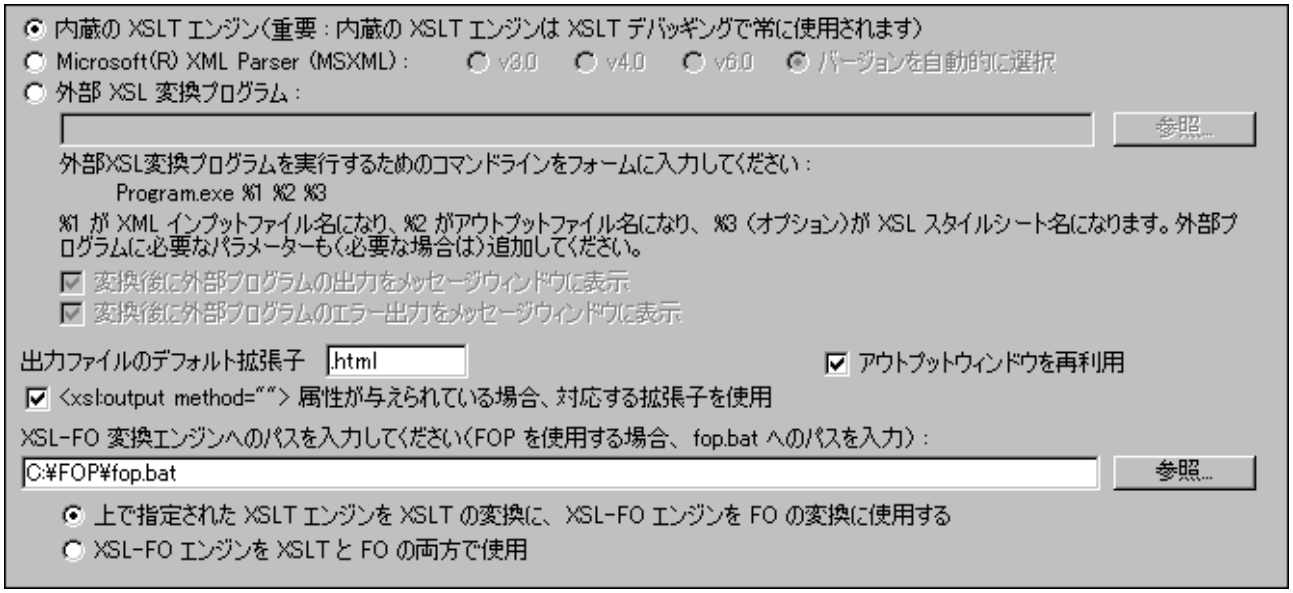
7.2 XSLT 処理

XMLSpy GUI では、2種類のXSLT 変換を利用することができます:

- 「XSL/XQUery | XSL 変換(F10)」コマンドにより、XSLT スタイルシートを使ったXML 変換処理を行ない、スタイルシートに記述されたフォーマットに従った変換が行われます。
- 「XSL/XQuery | XSL:FO 変換」コマンドにより、(i) XML からFO、そしてFO からPDF への2ステップの変換、または(ii) FO からPDF への変換が行われます。

変換を行うXSLT プロセッサーの指定

変換に使用されるXSLT エンジンはオプションダイアログのXSL タブ[1479]にて指定されます(以下のスクリーンショットを参照)。



利用可能なオプションについてはユーザーリファレンス[1479]セクションを参照ください。XSL セクションで指定されたエンジンがXSLT 変換で使用されます。XSL:FO 変換では、それに加えて他の(FO プロセッサーに同封されている XSLT という)XSLT エンジンオプションを使用することができます。このオプションを選択するには、XSL セクションの下部に表示される、対応するラジオボタンを選択してください。(*上のスクリーンショットを参照*)。

FO プロセッサーの指定

FO からPDF への変換時に使われるFO プロセッサーは、オプションダイアログのXSL タブ[1479]の下部にあるテキストフィールドから指定することができます(上のスクリーンショットを参照)。

XSLT 1.0、2.0、3.0 Altova のXSLT エンジン

スタイルシートのXSLT バージョンは`xsl:stylesheet`(または`xsl:transform`)要素の`version` 属性により指定されます。XMLSpy にはAltova XSLT 1.0、Altova XSLT 2.0、およびAltova XSLT 3.0 エンジンが内蔵されており、`version` 属性の値(`1.0` または`2.0` または`3.0`)により、適切なエンジンが使用されます。

XSLT 変換

XSL 変換 **(F8)** コマンドは以下のようなシナリオで使用することができます:

- GUI にてアクティブかつXSLTドキュメントが割り当てられている[1287] XMLドキュメントの変換。XSLTドキュメントが割り当てられていない場合、「XSL 変換(F8)」コマンドを選択した際に割り当てを行うよう求められます。
- GUI にてアクティブなXSLTドキュメントの変換。「XSL 変換(F8)」コマンドをクリックすることで、アクティブなXSLT スタイルシートにより変換を行なうXMLファイルを選択するよう促されます。
- プロジェクトフォルダーやファイルの変換。プロジェクトフォルダーまたはファイルを右クリックし、コマンドを選択してください。

バックマッピング

バックマッピング[1285] 機能が有効化されていると、XSLT 変換が実行され、結果ドキュメントが元のXSLT+XMLドキュメントにバックマッピングされます。結果ドキュメント内のノードをクリックすると、XSLT 命令 とその特定の結果フラグメントを生成したXML ソースデータ また、XSLT 命令、または、XML データノードをクリックすると、他の2つのドキュメント内の対応するノードがハイライトされます。詳細に関しては、XSL/XQuery | XSLT/XQuery バックマッピングの有効化[1285] を参照してください。

XSL:FO 変換

XSL:FO 変換コマンドは以下のようなシナリオで使用することができます:

- GUI にてアクティブかつXSLTドキュメントが割り当てられている[1287] XMLドキュメントの変換。XMLドキュメントは、まず指定されたXSLT エンジンを使ってFOに変換されます。その後、FOドキュメントは指定されたFO プロセッサーによりPDF 出力へと変換されます。XSLTドキュメントが割り当てられていない場合、「XSL:FO 変換」コマンドを選択した際に割り当てを行うよう求められます。
- 指定されたFO プロセッサーによりFOドキュメントをPDF へ変換。
- GUI にてアクティブなXSLTドキュメントの変換。「XSL:FO 変換」コマンドをクリックすることで、アクティブなXSLT スタイルシートにより変換を行なうXMLファイルを選択するよう促されます。
- プロジェクトフォルダーやファイルの変換。プロジェクトフォルダーまたはファイルを右クリックし、コマンドを選択してください。

XSL:FO 出力ダイアログ[1280]についての詳細は、ユーザーリファレンス[1280]を参照ください。

XSLT のパラメーター

Altova XSLT エンジンを使用しているのであれば、XSLT パラメーターをGUI ダイアログに保管しておくことができます。全ての保管されたパラメーターは、変換時にXSLTドキュメントへと送られます。詳しい情報については、XSLT パラメーター/XQuery 変数[1281]コマンドに関する記述を参照ください。

RaptorXMLを使ったバッチ処理

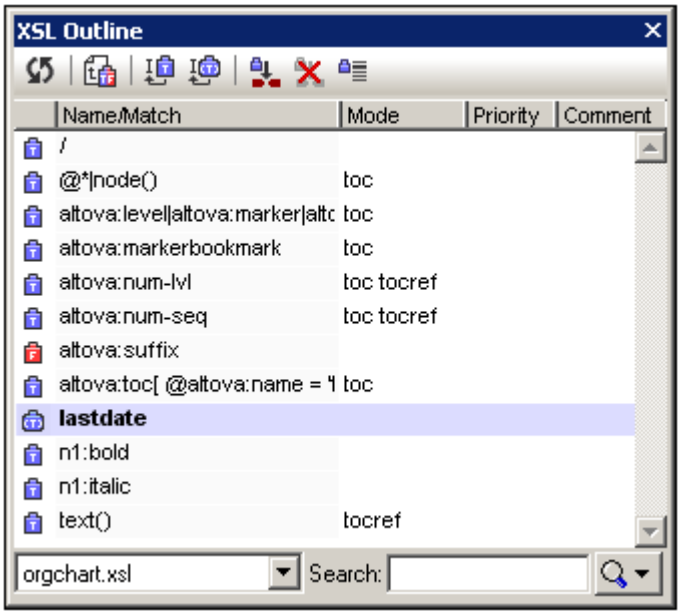
RaptorXMLは無料のスタンドアロンアプリケーションで、Altova の最新 XML バリデーター、XSLT エンジン、そしてXQuery エンジンが搭載されています。コマンドライン、COM インターフェース、Java プログラム、そして.NET アプリケーション内から使用することができ、XMLドキュメントの検証、XSLT スタイルシートを使用したXMLドキュメントの変換、そしてXQuery ドキュメントの実行を行うことができます。

RaptorXMLを使うとXSLT 変換タスクを自動化することができます。例えば、RaptorXMLを呼び出してドキュメントのセットを変換するバッチファイルを作成することができます。詳細はRaptorXML ドキュメンテーションを参照ください。

7.3 XSL アウトライン

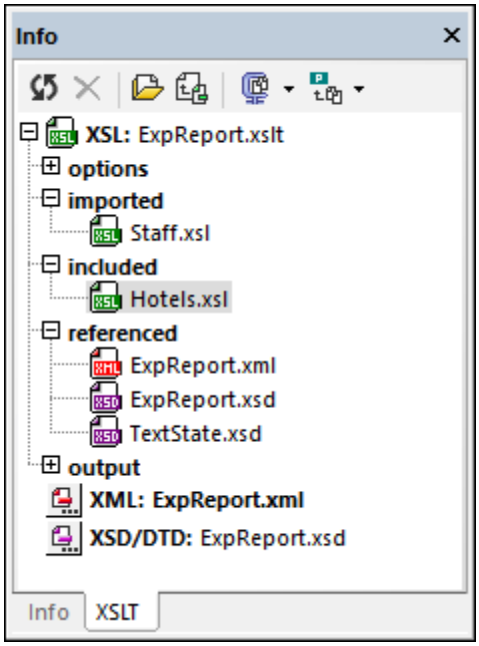
XSLT ドキュメントが XMLSpy のアクティブなドキュメントである場合、ドキュメントの構造に関する情報が XSL アウトラインウィンドウ[434]に表示され、XSLT ドキュメントに関連したファイル情報が(XSLT ドキュメントが XMLSpy でアクティブになっている時にだけ表示される)情報ウィンドウの XSLT タブ[437]に表示されます。更に、これら2つのウィンドウにて XSLT ドキュメントの編集や、関連するファイルの管理を行ないやすくする数々のコマンドが利用可能になります。

(下のスクリーンショットにある) XSL アウトラインウィンドウ[434]では、以下の操作を行うことができます:



- アクティブな XSLT ドキュメント、インポート/インクルードされた全てのドキュメントにあるテンプレートと関数を閲覧。
- 名前、マッチ、モード、優先度、またはコメントによるテンプレートと関数の並び替え。
- 名前/表現により特定のテンプレートを検索。
- XSL アウトラインを使った XSLT ドキュメント内の対応するテンプレートをナビゲーション。
- 名前付きテンプレートに呼出しを素早く追加
- 選択された名前付きテンプレートを変換のエントリポイントとしてセット

(下のスクリーンショットにある) 情報ウィンドウの XSLT タブ[437]を使って以下の操作を行うことができます:

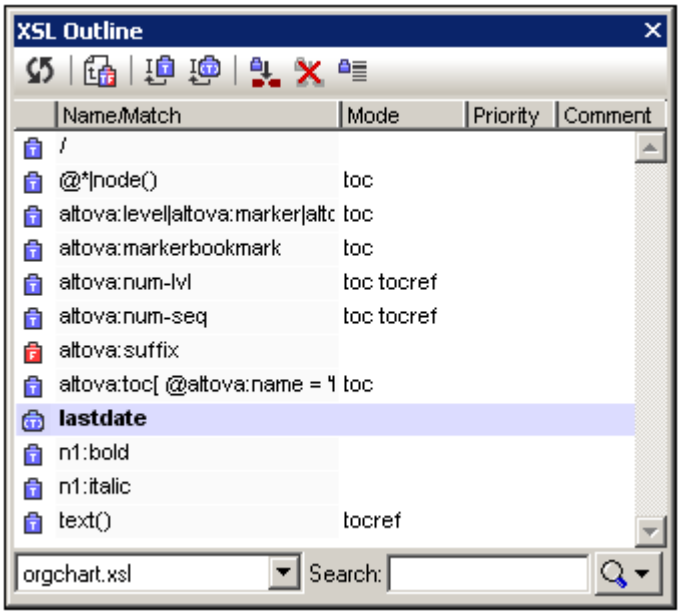


- インポート/インクルードされたファイルの位置といった、アクティブな XSLT ドキュメントに関連した全てのファイルに関する情報の閲覧。
- アクティブな XSLT ドキュメントを使って変換する XML ファイルのセット。更に、選択された XML ファイルの検証を行うスキーマ（XSD/DTD）ファイルをセットすることもできます。
- 情報ウィンドウから関連するファイルを開く。
- XMLSpy プロジェクトにて関連するファイルを素早く管理。
- 関連する全てのファイルを ZIP ファイルにまとめて、ユーザーの指定した場所に保存。

XSL アウトラインウィンドウ[434]ならびに情報ウィンドウの XSLT タブ[437]に関する詳しい情報は、このセクション以下のサブセクションを参照ください。

7.3.1 XSL アウトラインウィンドウ

（以下のスクリーンショットにある）XSL アウトラインウィンドウでは、アクティブな XSLT ドキュメントにある全てのテンプレートと関数が表示されます。テンプレートは青色のアイコン（パラメーターを持たないテンプレートは で、パラメーターを含むテンプレートは ）で表示されます。関数は赤の アイコンで表示されます。ウィンドウの左下にあるコンボボックスを使うことで、(i)（以下のスクリーンショットにあるように）アクティブな XSLT ドキュメントからのみ、または(ii) アクティブな XSLT ドキュメントとインポート/インクルードされた全てのスタイルシートから得られるテンプレートと関数を表示するよう指定することができます。



テンプレートには、(i) 名前付きテンプレートと、(ii) XPath 条件式にマッチするテンプレートの2つの種類があります。各テンプレートは以下の情報と共に表示されます:

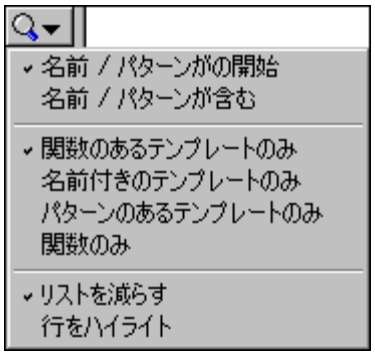
- (テンプレートに`name` 属性があれば) テンプレートの名前、そして(テンプレートに`match` 属性があれば) XPath 条件式。テンプレートが両方の属性を持っている場合、両方の情報が表示され、`name` 属性、`namevalue` 属性、そして`matchvalue` 属性の値と共に表示されます(スクリーンショットに太字で表示されているテンプレートに注目してください)。
- (モードがある場合) モード。テンプレートには1つ以上のモードがある可能性があることにご注意してください。(*上のスクリーンショット参照*)。
- (優先度がある場合) 優先度。
- (テンプレートまたは関数に先行するコメントがある場合) コメント

スタイルシートの関数は名前により表示されます。関数にはモードも優先度もありません。

オペレーション

XSL アウトラインウィンドウでは以下のオペレーションを実行することができます:

- *フィルタリング*: ウィンドウに表示されているリストを、以下のうちのどれかを表示するようにフィルタリングすることができます: (i) 全てのテンプレートと関数(XMLSpy が起動されるたびに適用されるデフォルト設定)、(ii) 名前付きテンプレートのみ、(iii) XPath 条件式テンプレートのみ、(iv) 関数のみ。目的のフィルタリングを選択するには、ウィンドウ右下に位置している検索ボックスをクリックして、表示されたリスト(メニューの第2グループ)から目的のフィルターを選択します。選択されたフィルターはただちに適用され、設定が変更されるかXMLSpy を終了するまで有効になります。



- *並べ替えと検索*：列のヘッダーをクリックすることで、各列をアルファベット順に並べ替えることができます。再度クリックすることで並べ替えの順序を反転させることができます。この操作で列の並べ替えを行った後、リスト内で任意のアイテムを選択し、並べ替えられた列にて単語を入力すると、その単語を含むリスト内のアイテムがハイライトされます。この機能を使えば、目的の名前、条件式、モード、または優先度を持つテンプレートへ素早くたどり着くことができます。
- *検索*：(ウィンドウの右下にある) 検索テキストフィールドに、検索を行ないたい名前またはXPath 条件式を入力することができます。入力中にも検索の結果は表示されます。(上のスクリーンショットにある) 検索ボックスに表示されるリストから、以下のオプションを指定することができます: (i) 名前または条件式が検索語を含む (または検索語から始まる)。検索語から始まるオプションはXMLSpy が起動されるたびにデフォルトオプションとして指定されます。(ii) 検索結果をハイライトするか、それ以外を表示しないよう設定。表示を減らすオプションはXMLSpy が起動されるたびにデフォルトオプションとして指定されます。これらの選択は直ちに適用され、設定を変更するか、XMLSpy を終了するまで有効になります。
- *再ロード*：スタイルシートが変更された後、ウィンドウのツールバーにある「**同期**」アイコン によりXSL アウトラインを更新することができます。
- *アイテムへ移動*：XSL アウトラインウィンドウにてテンプレートまたは関数が選択されたとき、ウィンドウのツールバーにある「**定義へ移動**」アイコン をクリックすることで、ドキュメント内にあるテンプレートまたは関数がデザインビューでハイライトされます。もしくはエントリーをダブルクリックすることでも同様の効果を得ることができます。
- *名前付きテンプレートアクション*：名前付きテンプレートに関しては、2種類の操作を行うことができます。(i) (`xsl:call-template` を使った) 名前付きテンプレートへの呼び出しをスタイルシート内のカーソル位置に挿入することができます。(ii) 名前付きテンプレートを変換のエントリポイントとしてセットすることができます。これらのオペレーションのコマンドは以下に記述されるようにツールバー上のアイコンから起動することができます。

変換のためのテンプレートモード

変換のための設定モードと呼ばれるツールバーのコンボボックスには、以下がリストされています (i) スタイルシートの全てのモード (ii) XSLT 3.0 スタイルシートの場合 (デフォルトモードを選択する) 空のエントリ (iii) `#unnamed` モード。ドロップダウンリストからモードを選択して、選択されたモードを変換のために設定します。(すべてのXSLT バージョンの) `#unnamed` モードは、`mode` 属性を持たないすべてのテンプレートモードに適用することができます。

XSLT 1.0 とXSLT2.0 スタイルシートの場合、デフォルトのモードは `#unnamed` です。ですから、空のエントリを選択することは、(`#unnamed` モードで、結果すべてのテンプレートに`mode` 属性無しで適用される) デフォルトのモードを選択することになります。

XSLT 3.0 スタイルシートでは、トップレベルの`xslt` 要素は、変換のためのデフォルトモードを持つ`default-mode` 属性を持つことができます。変換のためのモード設定コンボボックスでは、空のエントリ (デフォルトモード) が選択されていて、`default-mode` 属性で指定されているモードが変換モードとして使用されます。`#unnamed` モードがコンボボックスで選択されている場合、変換はすべてのテンプレートに名前のないモードで適用されます。すなわち、テンプレートにはモード属性がありません。

メモ テンプレートにモードの値 `#all` を与え、全てのモードに適用することができます。

名前付きテンプレート

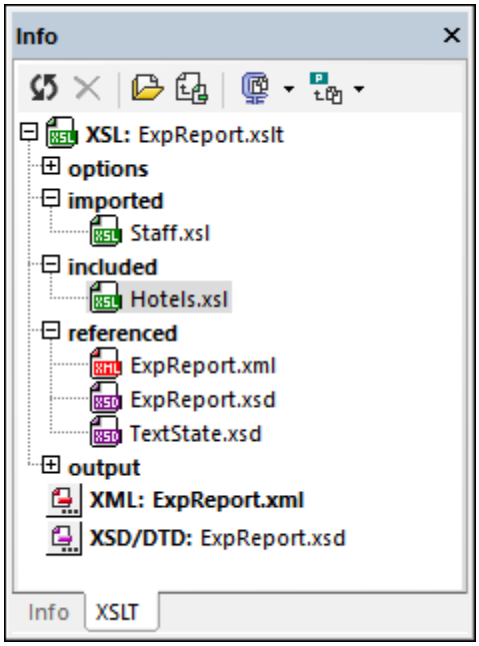
名前付きテンプレートが選択されていれば、（以下のスクリーンショットにあるように）ウィンドウのツールバーにある名前付きテンプレートに関連したコマンドが選択可能になります。

ツールバーのコマンドは左から順に以下のようになります:

- *xsl::call-template の挿入* : このコマンドは XSL アウトラインウィンドウにて名前付きテンプレートが選択された時に有効になります。このコマンドにより `xsl:call-template` 要素がスタイルシートのカーソル位置に挿入されます。スタイルシートに挿入される `xsl:call-template` 要素の `name` 属性値は、選択された名前付きテンプレートの `name` 属性の値になります。これにより `xsl:call-template` が選択された名前付きテンプレートへの呼び出しになります。
- *パラメーターを含む xsl:call-template の挿入* : このコマンドは XSL アウトラインウィンドウにて選択された名前付きテンプレートが `xsl:param` 要素を持っている時に選択可能になります。**xsl:call-template の挿入**コマンドのように、`xsl:call-template` 要素が挿入されますが、このコマンドの場合、選択された名前付きテンプレートの `xsl:param` 子要素に対する `xsl:with-param` 子要素も一緒に挿入されます。挿入される `xsl:call-template` と、その `xsl:with-param` 子要素の名前は、選択された名前付きテンプレートとその `xsl:param` 子要素の名前に対応します。
- *選択された名前付きテンプレートを変換のエントリポイントとしてセット* : 名前付きテンプレートが変換のエントリポイントとしてセットされている場合、XMLSpy で実行される変換はこの名前付きテンプレートから開始されます。XSL アウトラインウィンドウでは、エントリポイントとなる名前付きテンプレートが（このセクションの始めにあるスクリーンショットにあるように）太字で表示されます。
- *名前付きテンプレートを変換のエントリポイントから解除* : 名前付きテンプレートが変換のエントリポイントとしてセットされると選択可能になります。
- *変換のエントリポイントとしてセットされた名前付きテンプレートへ移動* : 名前付きテンプレートが変換のエントリポイントとしてセットされると選択可能になります。変換のエントリポイントとしてセットされた名前付きテンプレートとは別のアイテムに XSL アウトラインウィンドウのフォーカスがある場合、このアイコンをクリックすることで XSL アウトラインウィンドウにある名前付きテンプレートがハイライトされ、素早くアクセスすることができるようになります。

7.3.2 情報ウィンドウ

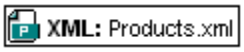
XMLSpy 内のアクティブなドキュメントが XSLT ドキュメントの場合のみ、情報ウィンドウの XSLT タブが表示されます。全てのインポートされたファイルおよびアクティブな XSLT ドキュメントに関連した XSLT ファイルを表示します。XSLT がアクティブドキュメントとして変換が開始された場合、XSLT を使用して変換する XML ファイルを選択することもできます。



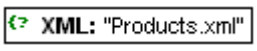
情報ウィンドウのXSLT タブには以下のファイルが表示されます:

- *XSLT ファイル*: (上のスクリーンショットにあるように) インポート/インクルードされた全てのXSLT ファイルが表示されます。ファイル上にマウスを移動させると、各ファイルの位置がポップアップで表示されます。インポートまたはインクルードされたファイルをダブルクリックするか、選択して 情報ウィンドウツールバーにある「開く」アイコンをクリックすることで、ファイルが新しいウィンドウで開かれます。ツールバーにある「Include/Import 位置へ移動」アイコンにより、アクティブな XSLT ドキュメントのinclude/import 宣言部がハイライトされます。

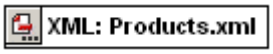
- *XML ファイル*: 変換を行うためにXML ファイルをアクティブな XSLT スタイルシートに割り当てることができます。割り当てられたXML ファイルの場所は、ファイル上にマウスを移動すると表示されるポップアップで確認することができます。XML ファイルを指定した後、そのXML ファイル上でメニューコマンド「XML/XQuery | **XSL 変換(F10)**」を選択することで、定義済みのXSLT スタイルシートを使った変換が実行されます。XML ファイルは**XML** アイコンをクリックすることで選択することができます(選択されたファイルは太字で表示されます)。その他にもプロジェクトプロパティダイアログ(*XSL ファイルのXSL 変換*) やXSLT ドキュメントの処理命令(`<?altova_samplexml "products.xml"?>`)からXML ファイルを割り当てることができます。どのような方法でも、以下のようなアイコンと共にXML ファイルはInfo ウィンドウに表示されます:

  プロジェクトプロパティダイアログにより割り当てられている

  XSLT ドキュメントの処理命令により割り当てられている

  XML アイコンをクリックし、ファイルの参照を行うことにより割り当てられている(太字で表示されます)

  同じファイルに対して2つ以上の方法で割り当てがなされている場合、選択の優先順位は:(i) プロジェクト、(ii) 処理命令、(iii) ユーザーによる参照、となります。アイテムをダブルクリックするか、選択後に「開く」ツールバーアイコンをクリックすることでXML ファイルを開くことができます。

- *XSD/DTD ファイル*: 選択されたXML ファイルがスキーマ(XML スキーマまたはDTD)への参照を含む場合、XSD/DTD エントリーに、そのスキーマファイルが表示されます。その他にも、(XML ファイルの場合と同じように) プロジェクトプロパティダイアログ(*検証*) からスキーマファイルを選択し、「XSD/DTD」アイコンをクリックし、目的のスキーマファイルを選択することもできます。スキ

ーマファイルがプロジェクトプロパティダイアログから選択された場合、エントリーの隣にプロジェクトアイコンが表示され、それ以外の場合には(「XSD/DTD」アイコンによりスキーマが選択された場合は太字で表示される)ファイルエントリーの隣に「XSD/DTD」アイコンが表示されます。スキーマファイルが2つ以上の方法で割り当てられている場合、選択の優先順位は:(i) プロジェクト、(ii) ユーザーによる参照、(iii) XML ドキュメント内の参照、となります。ファイルにマウスを移動させると、割り当てられた XSD ファイルの位置がポップアップで表示されます。ダブルクリックまたは選択後に「開く」ツールバーアイコンをクリックすることでスキーマファイルを開くことができます。

注意: XML または XSD/DTD ファイルがプロジェクトプロパティダイアログから選択された場合、この選択を解除するにはプロジェクトプロパティダイアログを開き、そこで選択を解除する必要があります。選択が「XML」または「XSD/DTD」アイコンにより選択がなされた場合、設定を解除するには、ファイルを選択して、情報ウィンドウツールバーにある「クリア」アイコンをクリックしてください。

オプション

XPath インテリジェント編集:XML ファイルが割り当てられている場合、XML ドキュメントの構造が走査され、インテリジェント XPath 編集[429]が要素と属性にも拡張されます。XSLT ドキュメント内の XPath 条件式を入力することができる場所では、使用可能な要素と属性がポップアップで表示されます。このオプションはデフォルトで有効になっており、チェックボックスのチェックを外すことで XPath インテリジェント編集を無効にすることができます。設定はファイルが閉じられるときに各 XSLT ファイルに個別に保存され、ファイルが開かれる際にその設定が読み込まれます。

ツールバーアイコン

(以下のスクリーンショットにある)情報ウィンドウツールバーアイコンを、左から右に説明します:

- *再ロード*:XSLT ドキュメント内で行われた修正を反映するよう情報ウィンドウを更新します。
- *XML/XSD 割り当てをクリア*:XML または XSD/DTD アイコンを使ってなされた XML や XSD/DTD の割り当てを解除します。解除を行うファイルを選択して、このアイコンをクリックしてください。
- *ドキュメントを開く*:選択されたドキュメントを開きます。
- *import/include 位置へ移動*:インポートまたはインクルードされたファイルが選択されているとき、このアイコンをクリックすることで XSLT ドキュメント内の関連するインポートまたはインクルード宣言がハイライトされます。
- *全てのドキュメントを圧縮*:情報ウィンドウに表示されている全てのドキュメントを圧縮して、ユーザーの定義した場所に保存します。または、ドキュメントを選択して、アイコンのドロップダウンメニューから「**選択したドキュメントを圧縮**」コマンドを選択することで、選択されたファイルだけを圧縮することができます。
- *プロジェクトに全てのファイルを追加*:現在のプロジェクトに全てのファイルを追加します。または、ドキュメントを選択し、アイコンのドロップダウンメニューから「**プロジェクトに選択したファイルを追加**」コマンドを選択することで、選択したファイルだけをプロジェクトに追加することができます。

7.4 XSL スピードオプティマイザー

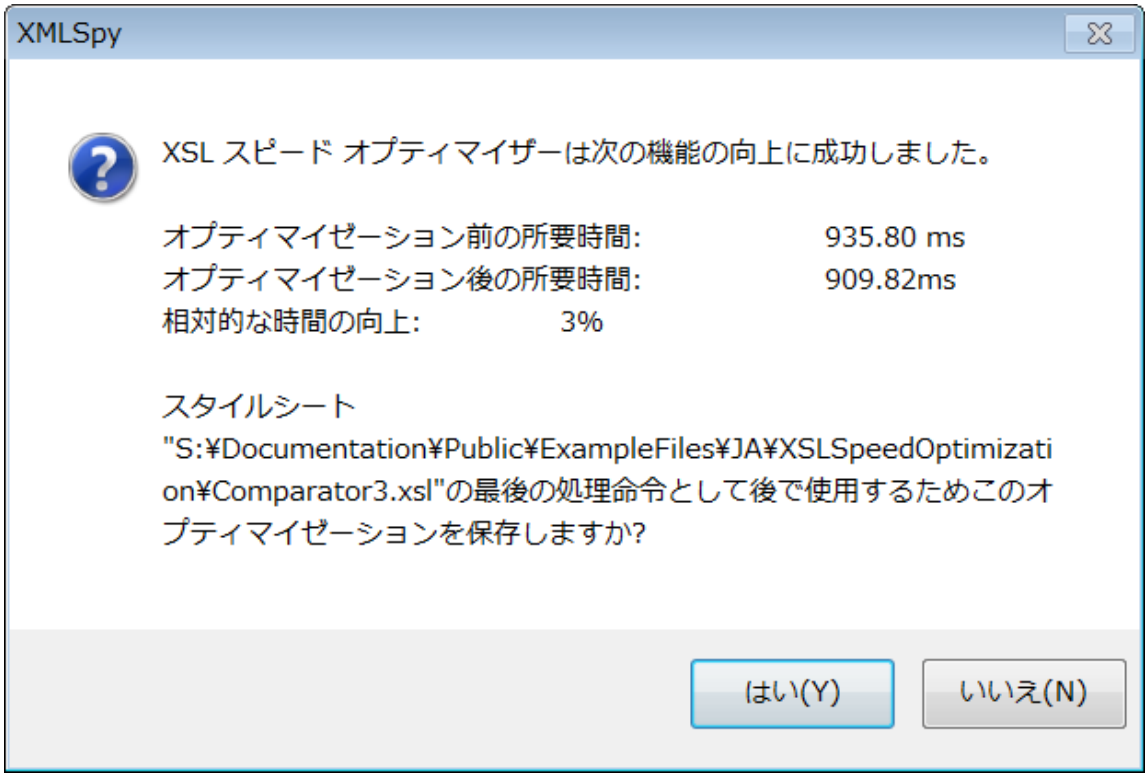
(このセクションではオプティマイザーと称される) XSL スピードオプティマイザーにより、XSLT スタイルシートを最適化することができ、結果変換が速く行うことができます。オプティマイザーは、XML データセット(1つ以上のXMLドキュメント)に対してXSLT スタイルシートを実行することにより作動し、スタイルシートのパフォーマンスを分析します。最適化の方略はこの分析から割り出され、(処理情報として)スタイルシートの最後にXSLT スタイルシートと共に保存されます。スタイルシートは、後により早い変換のために使用されます。

XSLT スタイルシートの最適化

XSLT スタイルシートを最適化するには、XSLT スタイルシートに加え、サンプルデータセットとしてサーブするXMLドキュメントが必要です。XSLT スタイルシートが的確に分析されるために、データセットは、のすべてのXSLT スタイルシート部分のために十分な大きさである必要があります。

オプティマイゼーションを以下のとおり行います:

1. XSLT スタイルシート またはXMLドキュメントがアクティブな場合、 メニューコマンド「XSL/XQuery | XSL スピード オプティマイザー」[1279] をクリックまたは、メインツールバー内のオプティマイザーのアイコンをクリックします。
2. XSLT またはXMLドキュメントがアクティブかどうかにより、XMLドキュメント またはXSLT スタイルシートを選択するようにプロンプトされます。「OK」をクリックすると、分析が開始されます。(XSLT またはXMLドキュメントが既に割り当てられている場合、このステップはスキップされます。分析開始されると、コマンドが直接呼び出されます。)
3. オプティマイゼーション 分析に失敗した場合、この結果についてのメッセージが表示されます。(オプティマイゼーション 分析の失敗の理由[441] は、下に説明されています) 分析に成功すると、分析の結果を表示するダイアログがポップアップします(下のスクリーンショット)。



ダイアログは、オプティマイゼーション(命令) を(スタイルシートの最後に処理命令として) XSLT スタイルシートに保存するオプショ

ンを与えます。「はい」をクリックしてオプティマイゼーションを保存、「いいえ」をクリックして、オプティマイゼーションを破棄します。オプティマイゼーションが保存されるたびに、以前に保存されたオプティマイゼーションを上書きします。

最適化されたスタイルシートをより速い変換のために使用することができます。

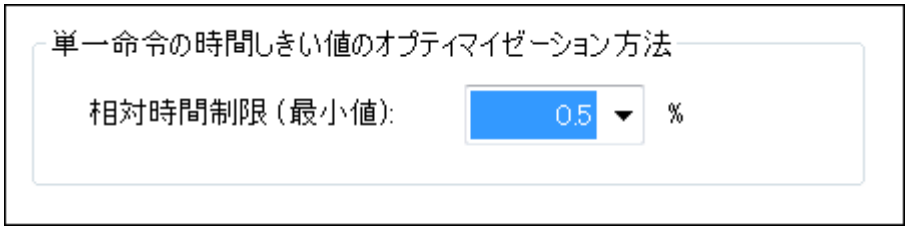
オプティマイゼーション分析の失敗の理由

XSL スピード オプティマイザーがオプティマイゼーションを引き起こせない場合、1 つまたは複数の以下の理由の可能性があります:

- XSLT スタイルシート 既に効率化されており、最適化される必要がない場合。
- 提出された、XML データセットが最適化するには小さすぎる可能性があります。より大きなデータセットで再試行してください。
- オプティマイゼーションのしきい値が高すぎる可能性があります。[オプションダイアログのXSL スピード オプティマイザータブ](#) 1481 のしきい値を変更してください。以下を参照。
- この特定のXSLT 構造のためのオプティマイゼーション構造を オプティマイザーで使用することができません。Altova サポートに問い合わせてください。

XSL スピードオプティマイザーの設定

オプティマイザーのための設定は、[オプションダイアログのXSL スピード オプティマイザータブ](#) 1481 で行うことができます(「ツール | オプション」、 *下のスクリーンショット*)。

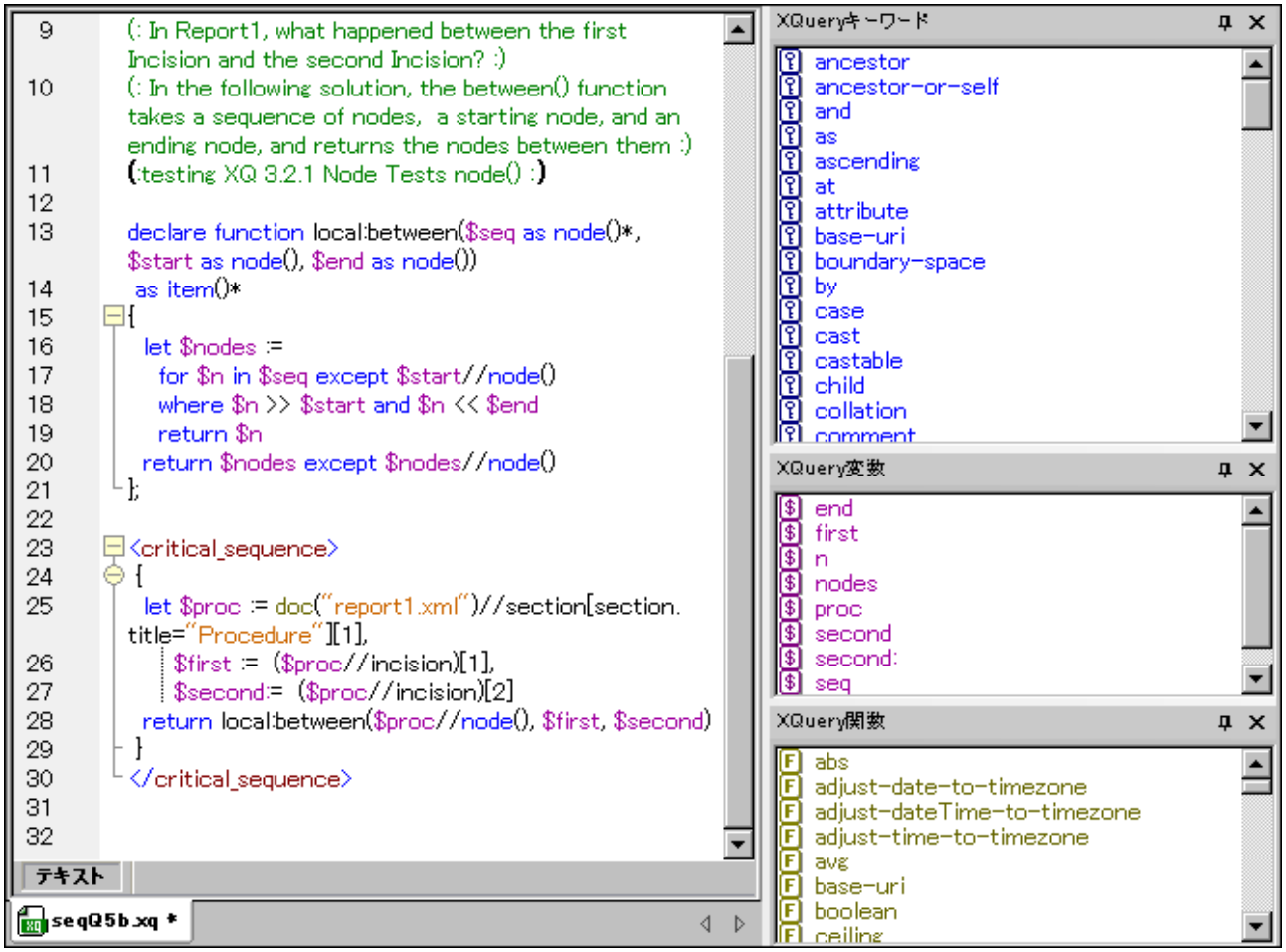


XSLT スタイルシート内の単一のXSLT 命令ためのしきい値がオプティマイザーのために指定されることができます。値の範囲を0.1% から、変換の総時間の99% に設定します。命令がしきい値として指定されたものよりも長く実行にかかる場合、分析が開始されます。それ以外の場合、分析は行われません。オプティマイゼーション 分析に失敗した場合、その理由としては、オプティマイザー 設定内のしきい値の設定が高すぎた可能性があります。しきい値を低くすることを考えてください。

8 XQuery

Altova ウェブサイト : XQuery エディター

XQuery および XQuery Update 456 ドキュメントはテキストビューで編集することができます。このビューにより入力ヘルパーや、構文による色分け、インテリジェント編集といった機能を使用することができます。(*スクリーンショット参照*) また、XQuery の検証を行ない、XQuery ドキュメント内にあるコードを内蔵の Altova XQuery エンジンを使って実行することができます。



注意 : XQuery ファイルはテキストビューでのみ編集することができます。その他のビューは XQuery に対してご利用になれません。

XQuery と XQuery Update ファイルの関連付け

XMLSpy 内では、XQuery と XQuery Update ドキュメントは 2 つの異なるドキュメントのタイプとして認識されます。通常 XQuery ドキュメントの拡張子は **.xq** で、XQuery Update ドキュメントのファイル拡張子は **.xqu** です。追加ファイル拡張子をこれらのファイルの型と関連付けることができ、ファイルの型の関連付けも、オプションダイアログのファイル型のセクション(ツール | オプション | ファイルの種類 1458)で変更することができます。

XQuery 実行または XQuery アップデート 456 により、「XQuery/Update 実行」コマンドが実行されるため、ファイル拡張子のドキュメントタイプの関連付けは重要です。

(C) 2015-2021 Altova GmbH

このセクション

このセクションは以下のように構成されています:

- XQuery ドキュメントの編集 444
- XQuery 評価 451
- XQuery の検証 452
- XQuery/Update 実行 453
- XQuery Update Facility 456
- XQuery とXML データベース 462

他の関連した機能と情報:

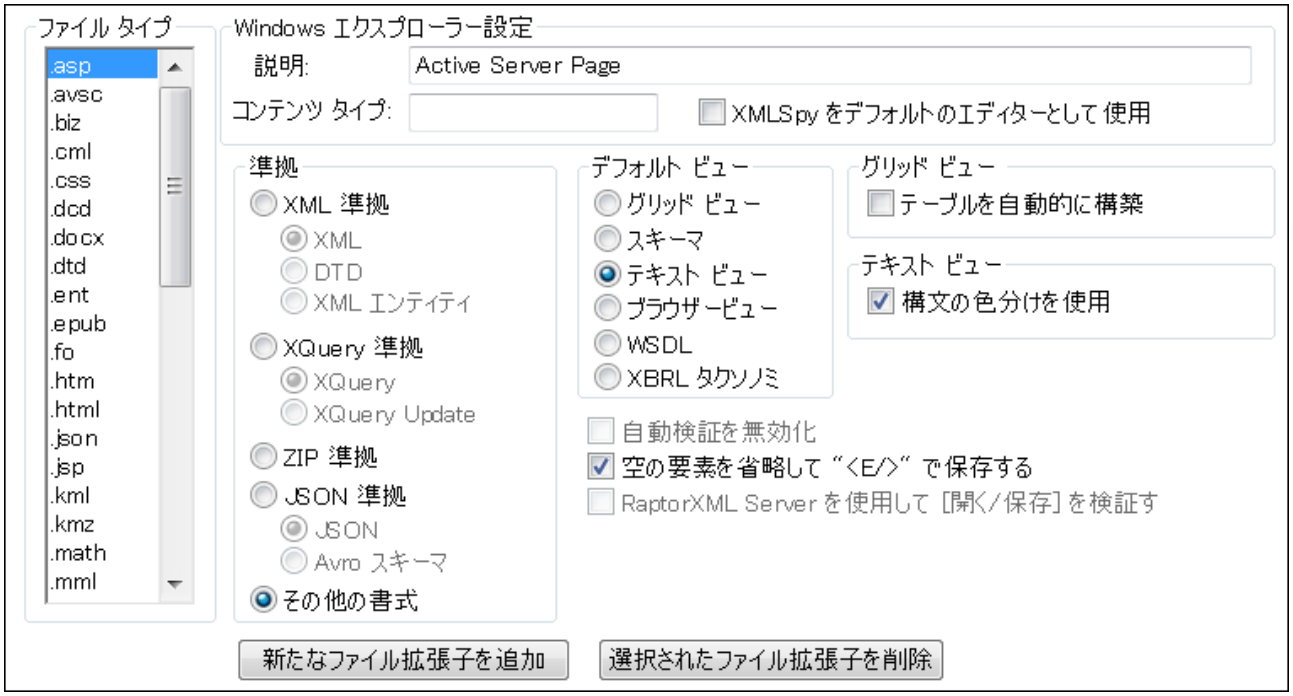
- XSLT/XQuery デバッガーとプロファイラー 466
- XQuery エンジンの実装 1974
- 出力ウィンドウ: XPath/XQuery 113
- ツール| オプション| ファイルの型 1458
- ツール| オプション| XQuery 1482

コマンドラインやバッチ処理のためのRaptorXML

XMLSpy のプロジェクト機能を使うことで、GUI からバッチ処理を行うことができます。より柔軟性のある処理をお望みの場合、XMLSpy にも搭載されている Altova XQuery エンジンを含む Altova のRaptorXML 製品を使用することができます。AltovaXML はコマンドラインやバッチ処理からXQuery の実行を行う理想的なツールになります。

8.1 XQuery ドキュメントの編集

XMLSpy 内では、XQuery と XQuery Update ドキュメントは 2 つの異なるドキュメントのタイプとして認識されます。ドキュメントタイプ (XQuery または XQuery Update) オプションダイアログのファイルの型内のファイル拡張子に割り当てられます。(ツール | オプション | ファイルの種類、1458 下のスクリーンショット)。XQuery または XQuery Update タイプのファイルが、XMLSpy で開かれた場合、そのファイルのテキストビューの XQuery 編集機能を使用することができます。



XMLSpy で XQuery と XQuery Update として現在定義されているファイル拡張子

XQuery	**.xq .xql .xqr .xquery**
XQuery Update	**.xqu**

メモ このセクションで説明される編集機能は XQuery と XQuery Update ドキュメント内の説明と同様です。

XQuery /Update 実行

XSL/XQuery | **XQuery/Update 実行** の GUI コマンドは、自動的に選択された、XQuery ファイルの XQuery 実行 または XQuery /Update を実行します。詳細に関しては、「XQuery /Update 実行」453 を参照してください。

8.1.1 XQuery ドキュメント

XQuery または XQuery Update ドキュメントは XQuery または XQuery Update に準拠している場合、自動的に、テキストビューの XQuery 編集モードで開かれます。ファイルはオプションダイアログ(ツール | オプション | ファイルの型1458 下のスクリーンショット) のファイル型のセクションの特定のドキュメント型に準拠するように定義されています。

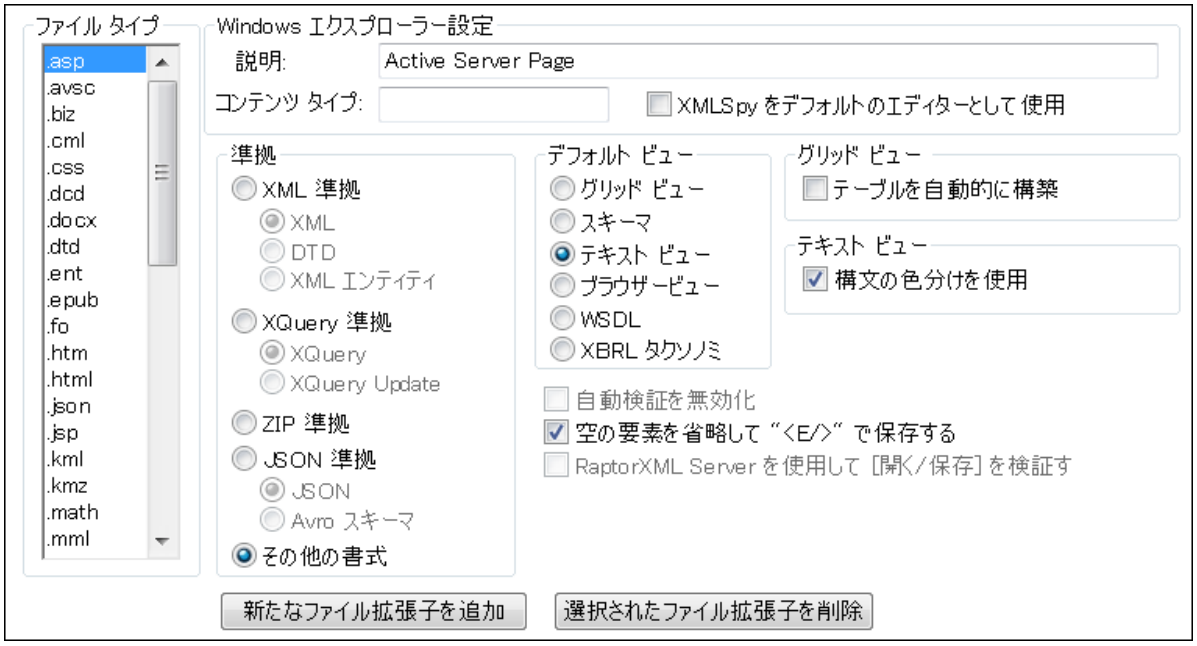
XMLSpy でXQuery とXQuery Update として現在定義されているファイル拡張子

XQuery	**.xq .xql .xqr .xquery**
XQuery Update	**.xqu**

XQuery 準拠のファイル拡張子を追加する

他のファイル拡張子を持つドキュメントをXQuery 準拠として認識させるには:

1. 「**ツール | オプション**」を選択。(以下のスクリーンショットにある)オプションダイアログが表示されます。



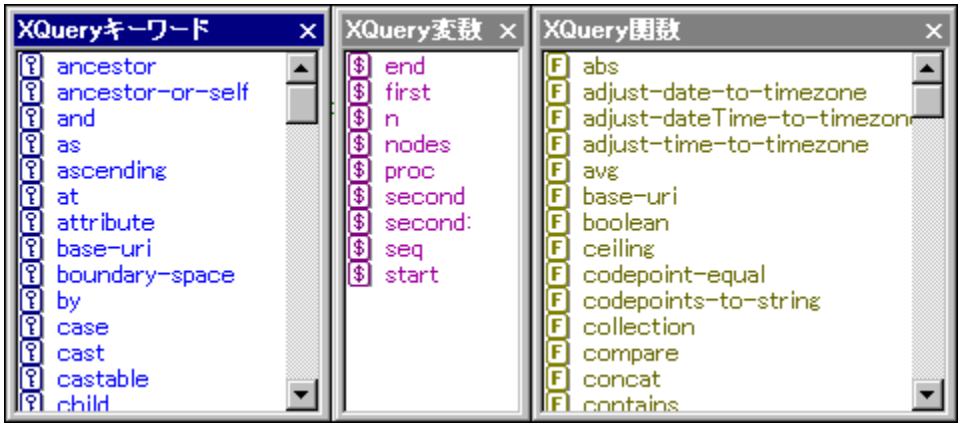
2. 「**ファイルの型**」セクションを選択。
3. 「**新たなファイル拡張子を追加**」をクリックして、ファイルの型のリストに新たな拡張子を追加。
4. 「**準拠**」にて、「**XQuery 準拠**」を選択。*XQuery* または*XQuery Update* を選択する。

次の設定も行ってください:

- *説明:* XML Query 言語 または XQuery Update Facility
- *コンテンツタイプ:* text/xml
- XMLSpy をXQuery ファイルの編集を行うデフォルトエディターとして使用する場合、「**XMLSpy をデフォルトのエディターとして使用**」チェックボックスのチェックを入れてください。

8.1.2 XQuery 入力ヘルパー

テキストビューのXQuery モードには3つの入力ヘルパーがあります:XQuery キーワード(青色)、XQuery 変数(紫色)、XQuery 関数(オリーブ色)。



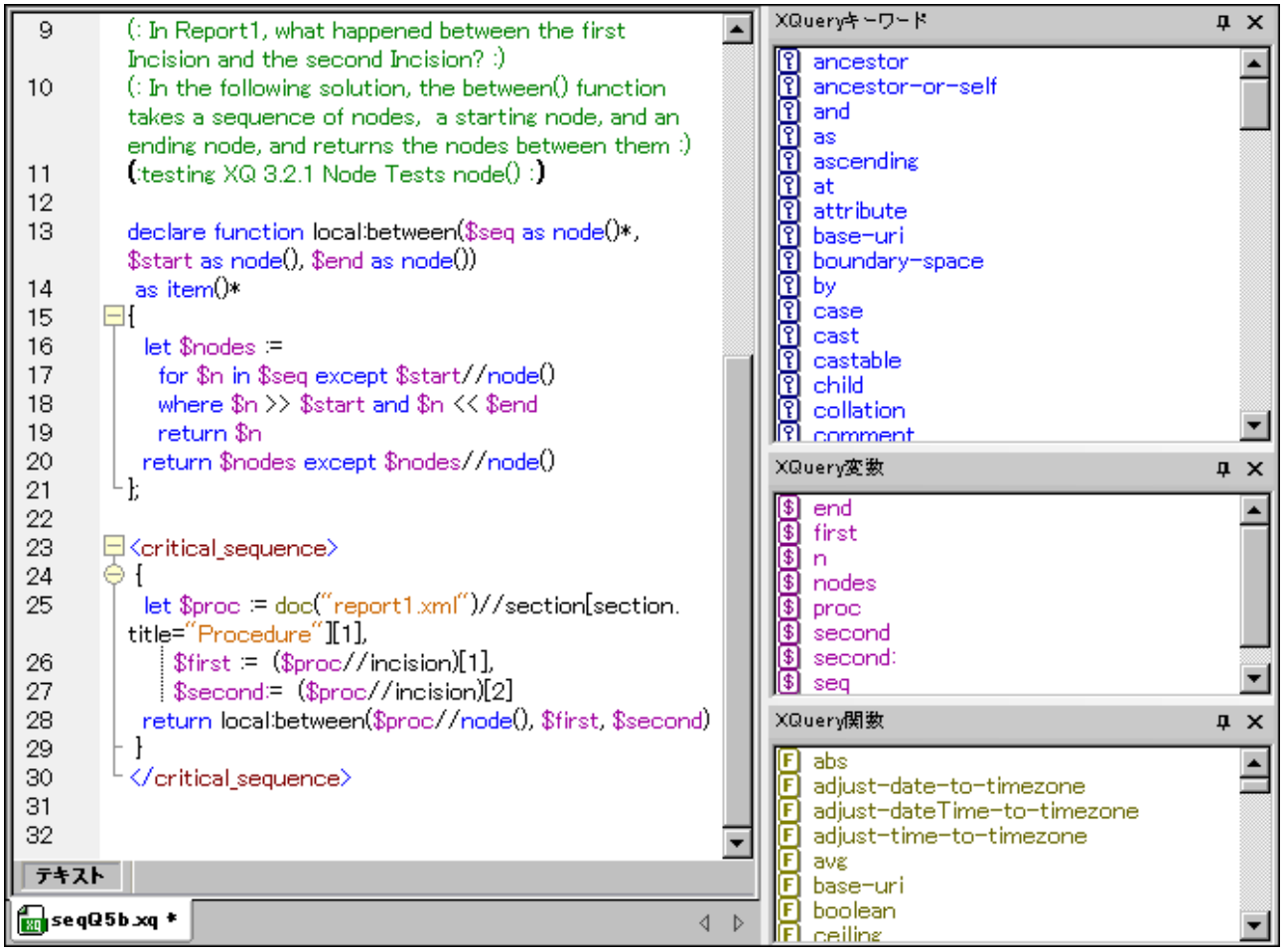
以下の点に注意してください:

- 入力ヘルパー内にあるアイテムの色は、それぞれテキストにて使用されている構文の色に対応しています。これらの色を変更することはできません。
- 表示されているキーワードと関数はAltova XQuery エンジンによりサポートされているものです。
- 関数はXQuery ドキュメント内で定義されたものです。テキストビューにて $ と文字列が入力された場合、その文字列が変数入力ヘルパーに表示されます。入力された変数が既に存在する変数名と同一の場合、新たに入力された変数名が入力ヘルパーから削除されます。
- 入力ヘルパーにてナビゲーションを行うには、入力ヘルパー内のアイテムをクリックして、スクロールバー、マウスホイール、または page-down や page-up キーを使って、リストを移動することができます。

入力ヘルパーにて表示されているアイテムをドキュメント内に挿入するには、挿入を行ないたい場所にカーソルを移動させ、アイテムをダブルクリックします。XQuery では文字列によってキーワードと関数の両方を表す場合があります(例: empty、unordered または except)。これらの文字列は(たとえ関数入力ヘルパーからその名前の関数を選択しても)常に(青色の)キーワードとして入力されます。関数が青色で表示された場合、その名前の後に続くかっこにより、関数と認識させることができます。

8.1.3 XQuery の構文による色分け

XQuery ドキュメントはXQuery コードとXML コードを内部に含むことができます。このセクションではデフォルトのXQuery 構文による色分けについて説明します。XQuery ドキュメント内のXML コードに対して使用される構文色分けは、通常のXML ドキュメントに対するものと変わりません。構文による色分けの色は(XQuery ならびにXML コードに対して)全てオプションダイアログ(「ツール | オプション」)のテキストフォントセクションにて設定することができます。(以下のスクリーンショットの例にあるように)中かっこ { } を使うことで、XML 要素内部にXQuery コードを埋め込めることに注意してください。



XQuery モードのXQuery コード内では、以下の構文色分けが使用されます:

- (: コメントは区切り文字も含めて緑色で表示されます :)
- XQuery キーワードは青色で表示されます: キーワード
- XQuery 変数は、ドル記号も含めて紫色で表示されます: $start
- かっこを除くXQuery 関数はオリーブ色で表示されます: function()
- 文字列はオレンジ色で表示されます: "Procedure"
- パス表現など、その他のテキストは(以下の下線部分にあるように)黒色で表示されます:
  for $s in doc("report1.xml")//section[section.title = "Procedure"]
  return ($s//incision)[2]/instrument

これらの色やフォントプロパティはオプションダイアログ(「ツール | オプション」)のテキストフォントセクションで変更することができます。

注意:上のスクリーンショットではコメント部分にあるかっこのペアの1つが太字の黒色で表示されていますが、これはかっこのマッチング機能によるものです(XQuery インテリジェント編集 448 を参照ください)。

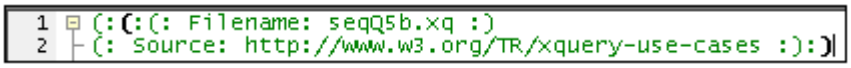
8.1.4 XQuery インテリジェント編集

テキストビューの XQuery モードでは以下のインテリジェントな編集機能が提供されます。

- かっこのマッチング 448
- キーワード 448
- 変数 448
- 関数 449
- ビジュアルなガイド 449

かっこのマッチング

かっこのマッチング機能により、かっこのペアとなる開始ならびに終了かっこがハイライトされ、かっこ内に含まれるコンテンツを簡単に確認することができます。この機能は、特に XQuery コメント内にてかっこがネストされているような状況で役に立ちます。

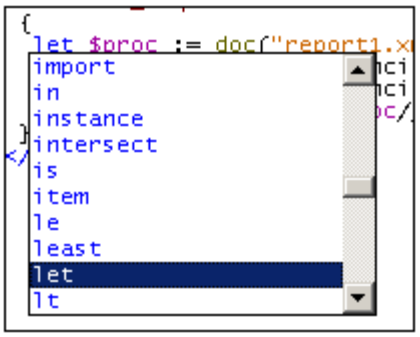


- かっこのマッチングはカーソルが(開始または終了)かっこの直前または直後に位置しているときに有効になります。かっこは対となるかっこと一緒に(太字に黒色で)ハイライトされます。上のスクリーンショットにあるカーソルの位置に注目してください。
- かっこのマッチングは、丸かっこ () 、角かっこ [] 、そして中かっこ {} で有効になります。例外は、XML タグに使用される不等号記号 <> です。

メモ: XML 要素の開始または終了かっこのすぐ内にカーソルがあると、両方のかっこがハイライトされます。「Ctrl」+ E のキーを押すことで、対となるペアがハイライトされます。「Ctrl」+ E を連続して押下することで、カーソルの位置を開始ならびに終了かっこ間で切り替えることができます。この機能を使うと、ドキュメントのナビゲーションを素早く行うことができます。

キーワード

XQuery キーワードはクエリー表現で使用される命令で、青色で表示されます。カーソルをキーワードの内部、直前、または直後に移動させることでキーワードを選択できます。キーワードが選択された状態で、「Ctrl」+「Space」を押すことにより、キーワードのリストがポップアップで表示されます。リストをスクロールし、目的のキーワードをダブルクリックすることで、選択されていたキーワードを置き換えることができます。

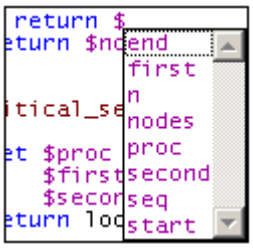


上のスクリーンショットでは、カーソルが `let` キーワード上に位置されており、ポップアップにあるキーワードをダブルクリックすることで、`let` キーワードを置き換えることができます。

変数

変数名の前には $ 記号が配置され、紫色で表示されます。このインテリジェント編集機能の仕組みはキーワードの編集機能のそれと似たようなものです。ドキュメント内の変数リストが表示されるポップアップへアクセスするには2つの方法があります:

- $ 文字を入力後に Ctrl+Space を押下
- 変数を選択後に Ctrl+Space を押下（$ 文字の直後、変数名の内部や直後にカーソルを位置させることで、その変数が選択された状態になります）

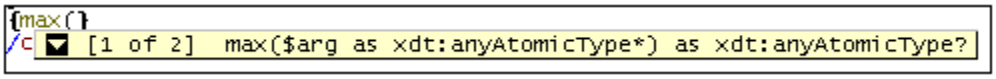


$ 記号の直後に変数を挿入する、または選択された変数を置き換えるには、ポップアップメニュから目的の変数をダブルクリックしてください。

関数

キーワードならびに変数の場合と同様に、（オリーブ色で表示される）関数を選択するか、「Ctrl」+「Space」を押下することで内蔵の関数がポップアップメニューにより表示されます。関数内、または関数名の直前、直後にカーソルを位置させることで、関数が選択された状態になります。関数名の直後にあるかっこ内部にカーソルを位置させても選択されている状態にはなりません。ポップアップメニュー内の関数名をダブルクリックすることで、選択された関数名をポップアップメニューの関数に置き換えることができます。

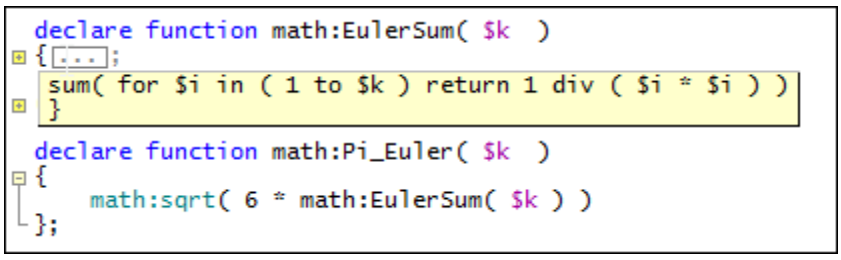
（以下のスクリーンショットにあるように）関数の開始かっこの直後にカーソルをおき、「Ctrl」+「Space」を押下することで、その関数のヒントを表示することができます。標準 XQuery 関数でしかヒントは表示され無いことにご注意してください。



ヒントの先頭にある下向きの矢印は、異なる引数や戻り値の型を持つ同名の関数が存在することを表しています。矢印をクリックすることで次の関数を表示することができ、クリックを繰り返すことでその名前の関数を全て表示させることができます。他にも「Ctrl」+「Shift」+「Up」または「Ctrl」+「Shift」+「Down」キーの組み合わせで、同様の操作を行うことができます。

ビジュアルなガイド

テキストの折りたたみ（またはソースのおりたたみ）はXQuery 中かっこ、XQuery コメント、XML 要素、XML コメントにより有効化され、これらのノードを展開または折りたたむ機能を意味します。このようなノードは、+/- サインによりソース折りたたみ余白内で表示されます（下のスクリーンショット参照）。余白はテキストビュー設定ダイアログ[1363]により切り替えのオンとオフを行うことができます。ノードが折りたたまれると、視覚的に省略記号により表示されます（下のスクリーンショット参照）。マウスのカーソルが省略にポイントされると、ポップアップで折りたたまれたノードの内容が表示されます（スクリーンショット参照）。内容がポップアップでの表示に大きすぎる場合、ポップアップのしたの部分に省略記号で表示されます。

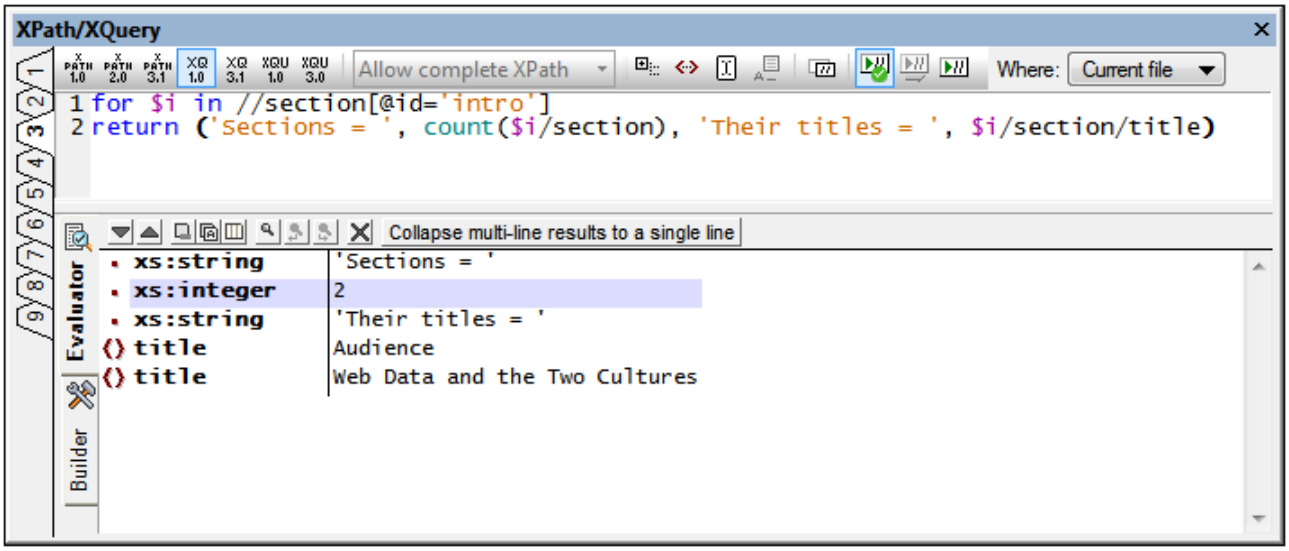


テキストツールバー内の「全ての折りたたみを切り替える」アイコン は、全てのノードを展開された書式でまたはトップレベルのドキュメント要素まで折りたたみ表示します。
ノードの +/- アイコンをクリックすることにより以下のオプションを使用することができます:

Click [-]	ノードを折りたたみます
Click [+]	子孫ノードが展開されて表示されるか、ノードが折りたたまれた前の状態に従い折りたたまれ、ノードが展開されます。
Shift+[-]クリック	すべての子孫ノードを折りたたみ、展開フォーム内でクリックされたノードを残します。
Ctrl+ [+]クリック	クリックされたノードとそのすべての子孫ノードを展開します。

8.2 XQuery 評価

XQuery 式は、XPath/XQuery 出力ウィンドウ内の 1 つまたは複数のドキュメントに対して評価されることができます（*下のスクリーンショット*）。



以下をおこなってください：

1. ウィンドウの上のペインにXQuery 式 を入力します。
2. *Where* コンボボックス内から（*上のスクリーンショット参照*）クエリされるXMLドキュメントの場所を選択します。 オプション: (i) 現在のファイル; (ii) ファイルを開く; (iii) プロジェクト; (iv) フォルダー.
3. 「XPath/XQuery 式の評価」(F5) をクリックします。XML ファイルに対して式が評価されます。指定された（*Where*）ロケーションが、1 つ以上のXML ファイルを含む場合、XML ファイル内で、式に一致するデータ構造またはコンテンツが検索されます。使用可能なすべての一致は、下のペイン内に表示されています。

上のスクリーンショット内で、属性 `@id='intro'` を持つセクション要素のためにクエリが作成されます。クエリは、この `intro` セクション サブセクションの数量とタイトルを返します。

詳細に関しては、出力ウィンドウ: XPath/XQuery [113] および アップデートのプレビューと適用 [456] を参照してください。

8.3 XQuery 検証

XQuery または XQuery Update ドキュメントを検証するには以下を行います:

1. XQuery ドキュメントをアクティブなドキュメントにします。
2. 「XML | 検証」を選択、「F8」キーを押す、または「検証」ツール バーアイコンをクリックします。

検証ツール バーアイコン

XQuery 構文に対してドキュメントが検証されます。

(C) 2015-2021 Altova GmbH

8.4 XQuery/Update 実行

XQuery または XQuery Update ドキュメントは以下の方法で実行することができます:

- XQuery または XQuery Update ドキュメントがアクティブな場合。
- XML ドキュメントがアクティブな場合。

メモ ドキュメントが XQuery ドキュメントまたは XQuery Update ドキュメントであるかは、ドキュメントのファイル拡張子によって決定されます。XMLSpy は、ファイルの種類の関連付けをオプションダイアログのファイルの型セクションによる定義に従い認識します。(ツール | オプション | ファイルの型[1458])。

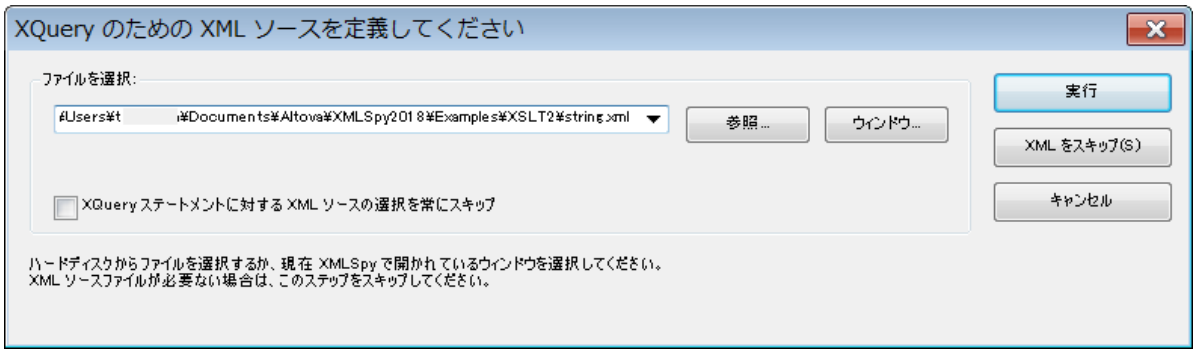
メモ XQuery Update には、XPath/XQuery 出力ウィンドウにアップデート式を入力することができます。アップデートが受け入れられた場合、アップデートを適用して、アップデートされたファイルを保存します。詳細に関しては、XQuery Update Facility[456] とアップデートのプレビューと適用[456] を参照してください。

XQuery または XQuery Update ドキュメントを実行する

XQuery または XQuery Update ドキュメントを *「XQuery / XQuery Update ドキュメントをアクティブにする」* で実行する場合は以下を行います。

1. XQuery または XQuery Update ドキュメントをアクティブなドキュメントにします。
2. 「XSL/XQuery | XQuery/Update 実行」を選択または、コマンドのツールバーアイコンをクリックします。これにより XQuery のための XML ソースを定義するダイアログが表示されます(下のスクリーンショット)。

   XQuery/Update 実行ツールバーアイコン



3. XML ファイルを参照するか、XML ソースの選択をスキップします。

通常、XQuery ドキュメントは、単一のドキュメントに関連付けられていません。これは、XQuery 式が `doc()` 関数を使用して、XML ドキュメントをいくつも選択することができるからです。XMLSpy 内では、しかしながら、個別の XQuery ドキュメントを実行する前に、実行のためのソース XML ドキュメントを選択することができます。この場合、選択された XML ソースのドキュメントノードは、XQuery ドキュメントのルートレベルで使用することのできるコンテキストメニューから開始します。行頭スラッシュから開始するパスは、コンテキストアイテムのドキュメントノードとしてこのドキュメントノードと共に解決されます。

メモ 「XQuery/Update」実行コマンドは、プロジェクトウィンドウ[110] アイテムのコンテキストメニューから使用することができます。

実行 / アップデートの結果

- *XQuery 実行:* 実行の結果は一時ファイルとして生成され、希望するファイル形式やファイル拡張子で任意の場所に保存することができます。

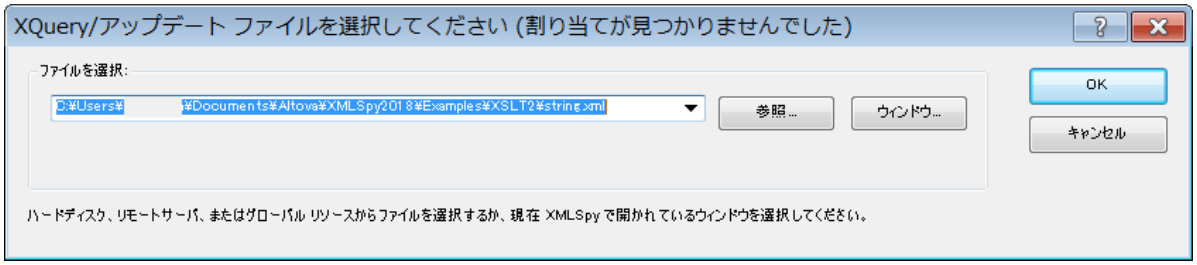
- *XQuery アップデート:* アップデートはファイルに保存されるか、アップデートされたファイルが開かれると、プレビューし保存または閉じることができます。どちらのアクションを実行するか指定することができます。これはオプションダイアログの *XQuery* セクション(「ツール | オプション | XQuery」[1482])で行うことができます。

アクティブな XML ドキュメントでの実行

XQuery または XQuery Update ドキュメントをアクティブなドキュメントで実行するには以下を行います。

1. XML ドキュメントをアクティブにします。
2. 「XSL/XQuery | XQuery/Update 実行」を選択するか、 ツールバーアイコンをクリックします。これにより XQuery のための XML ソースを定義するダイアログが表示されます。「XSL/XQuery | XQuery/Update 実行」を選択または、コマンドのツールバーアイコンをクリックします。これにより XQuery のための XML ソースを定義するダイアログが表示されます(下のスクリーンショット)。

   XQuery/Update 実行ツールバーアイコン



3. XQuery または XQuery Update ファイルを参照して「OK」をクリックします。

実行/アップデートの結果

- *XQuery 実行:* 実行の結果は一時ファイルとして生成され、希望するファイル形式やファイル拡張子で任意の場所に保存することができます。
- *XQuery アップデート:* アップデートはファイルに保存されるか、アップデートされたファイルが開かれると、プレビューし保存または閉じることができます。どちらのアクションを実行するか指定することができます。これはオプションダイアログの *XQuery* タブ(「ツール | オプション | XQuery」[1482])で行うことができます。

バックマッピング

バックマッピング[1285] 機能が有効化されていると、XSLT 変換が実行され、結果ドキュメントが元の XQuery+XML ドキュメントにバックマッピングされます。結果ドキュメント内のノードをクリックすると、XQuery 命令 とその特定の結果フラグメントを生成した XML ソースデータがハイライトされます。また、XQuery 命令、または、XML データノードをクリックすると、他の2つのドキュメント内の対応するノードがハイライトされます。詳細に関しては、XSL/XQuery | XSLT/XQuery バックマッピングの有効化[1285] を参照してください。

XQuery 変数

Altova XQuery エンジンを使用している場合、XQuery 変数を便利な GUI ダイアログ内に補完することができます。XMLSpy を使ったXQuery ドキュメントの実行時に、全ての保管された変数は XQuery ドキュメントに渡されます。より詳しい情報については、XSLT パラメーター/XQuery 変数[1281] コマンドの記述を参照ください。

Altova XQuery エンジン

Altova XQuery エンジンの実装や XQuery ファイルの処理に関する詳細については、XQuery エンジンの実装[1974] を参照ください。

8.5 XQuery Update Facility

XQuery Update Facility は、ドキュメントの部分を変更することのできるXQuery 言語の拡張機能です。通常のXQuery 実行では、ドキュメント全体が再生成され、そして、元の場所に保管される必要があります。これはドキュメントの小さいパートのみが変更される場合、非効率になる可能性があります。Update Facility を使用することにより変更された部分のみを更新することができます。

XQuery Update Facility は、XQuery 1.0 およびXQuery 3.1 の拡張機能として次の仕様内で説明されています:

- XQuery Update Facility 1.0 (2011 年 3 月 17 日版W 3C 勧告)
- XQuery Update Facility 3.0 (2013 年 1 月 8 日版W 3C 作業ドラフト)

XMLSpy 内のXQuery Update Facility

以下の要点は、XQuery Update がXMLSpy 内でどのように作動するかを説明しています:

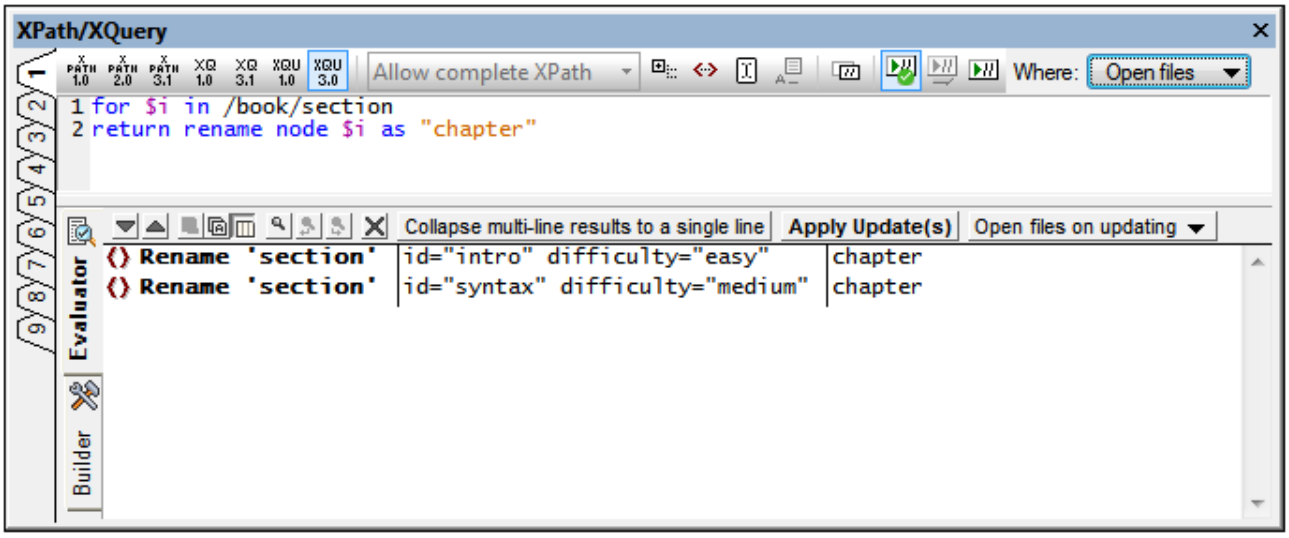
- アップデートはアップデート式により行われます。例えば、アップデート式は、XML ドキュメント内のノードの名前の変更を指定することができます:
  ```
  rename node /documents/doc-01 as "document-01"
  ```
- 実際のところは、複数のアップデート式が単一のXQuery Update ドキュメントに入力されます。
- アップデートドキュメント内の各 アップデート式が実行されると、結果はすぐに適用されませんが、*保留中の アップデート リスト (PUL)*に追加されます。 この結果、PUL は、すべてのアップデート式の結果を含みます。すべてのupdates in PUL のすべてのアップデートは、すべてに一度に適用されます。
- XMLSpy 内で、PUL アップデート 以下の2 つの方法で適用されます:
  (i) GUI 内でユーザーによりプレビューする利点は、プレビューは、予期しない結果を表示します。XQuery/Update 実行コマンド[453] または XPath/XQuery 出力ウィンドウ[456]. 内のXQuery アップデート式評価中でプレビューを使用することができます。プレビューオプションに関しては、関連する箇所で説明されています。
  (ii) 直接 そして ユーザーによる中断なし。利点は、アップデートが警告なしにユーザーによる中断なしに実行されます。アップデートの直接のアプリケーションはXQuery/Update 実行コマンド[453] またはXPath/XQuery 出力ウィンドウ[456] 内でXQuery アップデート式の評価を実行中に使用することができます。(プレビューなしの) 直接的アップデートのアプリケーションがが直接のアップデートオプションで使用可能な場合、オプションに関する情報は関連する箇所で説明されています。

XMLSpy は、アクティブなXML ドキュメントに対して使用されたアップデート式の効果をプレビューすることのできる、パワフルなXQuery Update プレビュー機能を与えます。この機能については、アップデートのプレビューと適用[456] のセクションで説明されています。

8.5.1 アップデートのプレビューと適用

XQuery Update を使用して、XML ドキュメントを修正する場合、XML ドキュメントに適用する前にプレビューして、変更されたドキュメントを保存することができます。

XPath/XQuery 出力ウィンドウ(*下のスクリーンショット*)内に 1 つまたは複数のアップデート式を入力することができます、そして下のペインに表示された**保留中のアップデートリスト (PUL)** 内でアップデートをプレビューすることができます(*下のスクリーンショット参照*)。PUL が希望する状態の場合、ドキュメントのアップデートを適用し、変更されたドキュメントを保存することができます。PUL に変更を付け加える場合は、変更を適用しない、または保存しないことを選択できます。

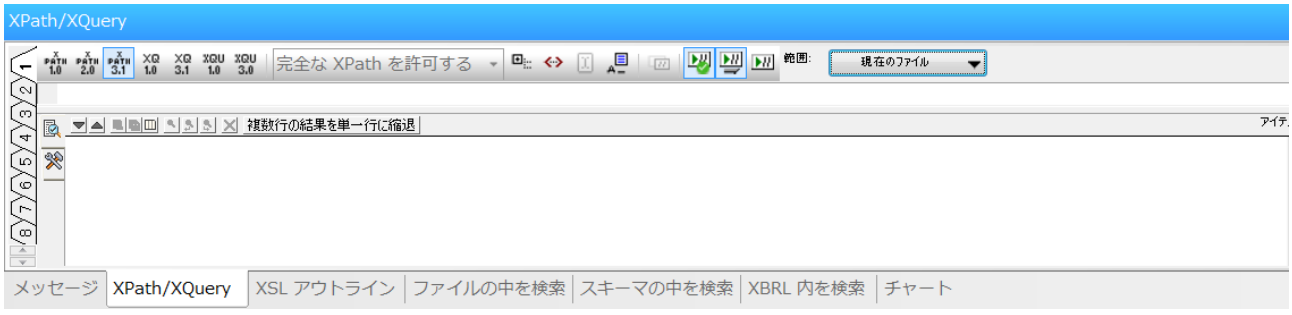


アクティブな XML ファイルのために PUL を作成するには、次を行います:

1. XPath/XQuery 出力ウィンドウのツールバーから（*上のスクリーンショット*） **XQU 1.0** または **XQU 3.0** アイコン（それぞれ XQuery Update 1.0 または XQuery Update 3. 0 のため）に選択します。
2. 1 つまたは複数の*アップデート*式をウィンドウの上のペインに入力します。*アップデート*式と構文の説明に関しては、[操作と構文のアップデート](#) 458 のセクションを参照してください。
3. ツールバーの*スキャンロケーション*コンボボックスから、*アップデート*のためにスキャンされるロケーションを選択します:
   ファイルを開く: XMLSpy で現在開かれているすべてのファイルがスキャンされます。
   現在のファイル: 現在アクティブな*ファイル*のみがスキャンされます。*スキャン*のために選択されたロケーションが*現在のファイル*の場合、「**XPath/入力中に XQuery 式の評価する**」ツールバーアイコンが有効化されます。
   プロジェクト: 現在アクティブな*プロジェクト*がスキャンされます。
   フォルダー: スキャンするフォルダーを選択することができます。
4. *アップデート*式を実行して、PUL を表示するには、「**XPath/XQuery 式の評価**」ツールバーアイコンをクリックします。

XPath/XQuery 出力ウィンドウツールバー

以下の XPath/XQuery 出力ウィンドウツールバーコマンド（*下のスクリーンショット内で赤枠の中*）を使用することができます。

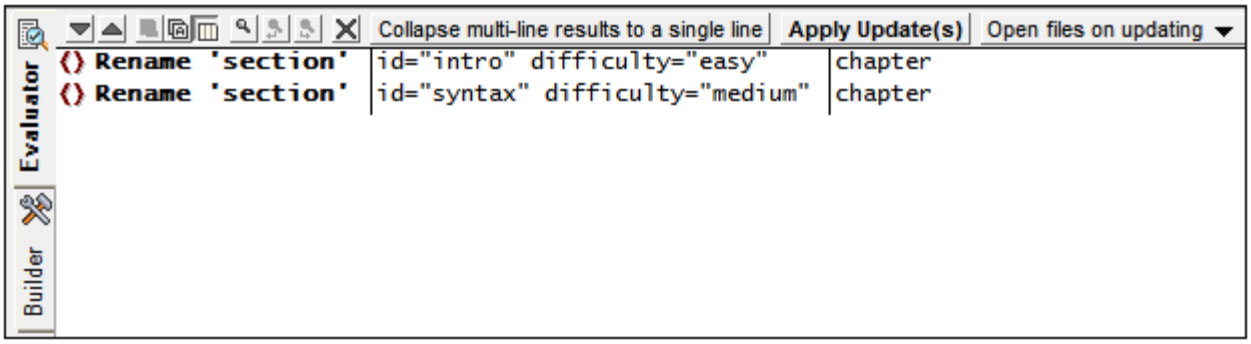


- ヘッダーの表示は、*アップデート*ロケーションのファイル名の表示の切り替えを行います。
- 完全な結果の表示は、*ノード*コンテンツ全体または*ノード*の属性のみの表示の切り替えを行います。
- XPath/XQuery 起点の設定は、*アクティブな*ファイル内のカーソルロケーションを相対的な XPath 式の起点として設定します。式は、この*ノード*に対して相対的に評価されます。

- 手動でノードのロケーターパス式を入力するかわりに、XMLSpy が入力を行うことができます。これは以下のように行います: (i) XPath 式にロケーターパスを入力する箇所にカーソルをポイントします、(ii) ターゲットするノードの開始タグ内にカーソルをポイントします、(iii) 式内にロケーターパスを入力するために、編集フィールドアイコン の現在の選択の XPath のコピーをクリックします。
- XML の検証 ファイルは、XML ファイル検証の切り替えを行います。
- 入力中の XPath/XQuery 式の評価コマンドは、解析オプションが現在のファイルに設定されていると有効化されます。コマンドがオンに切り替えられている場合、アップデート式が入力されている場合でも PUL は、生成されます。
- XPath/XQuery 式の評価 コマンドは、アップデート式を評価し、PUL を生成します。
- ロケーションのスキャン コンボボックス オプションは、上で説明されています。

保留中のアップデートリスト (PUL) ペイン

PUL ペインは、実行されるすべてのアップデートを表示します。ヘッダーの表示オプションがウィンドウのツールバー内で切り替えられている場合、ターゲットファイルのロケーションが表示されます。PUL の表示は、3 つの垂直のセクションに分割されています(*下のスクリーンショット参照*): (i) 実行するアップデート アクション、(ii) ターゲットノードのコンテンツはアップデートされます。(iii) アップデートアクションの結果



次の PUL ペイン ツールバー コマンドを使用することができます:

- **次と前** アイコンは、現在選択されているメッセージの次および前のメッセージをそれぞれ選択します。
- **ラインのコピー コマンド** は、選択されたラインおよびすべてのラインをクリップボードにそれぞれコピーします。
- **検索** コマンドは、PUL ペイン内のテキストを検索します。
- **クリア** コマンドは、PUL ペイン内のすべてのラインを削除します。
- **複数ラインの結果のおりたたみ** コマンドは、複数のラインのアップデートアイテムを単一のラインに折りたたみます。
- **アップデートの適用** コマンドは、保留中の アップデートをターゲットの場所に適用します。アップデートする際に、アップデートは、ファイルに保存、または、アップデートされたファイルが表示されます (そして、手動で 保存されるかを決定することができます)。次のオプションを参照してください。
- アップデート ファイル ディレクトリ コンボボックスにより以下を選択することが許可されます (i) ファイルがダイアログを表示せずにディスクにアップデート、または (ii) アップデートされたファイルが開かれそしてアクティブになるか。後者のオプション が選択された場合、非-アクティブなターゲット ファイルはアクティブに設定されます。変更されたドキュメントを保存または保存しない オプションがあります。

メモ 1 つまたは複数のファイルが直接ディスクにアップデートされた場合、変更されたファイルのリストが表示されます。リストの各ラインは、ファイルのロケーションおよびファイルへのクリックすることのできるリンクを表示します。

8.5.2 Update オペレーションと構文

XQuery Update Facility では以下のオペレーションを使用することができます:

- 1つまたは複数のノードの削除 459 する

- 1つまたは複数のノードを指定されたノードの前、後、または内部に挿入[459]する
- ノードの名前の変更[460]をする
- アイテムのシーケンスのノードと置換[460]える
- アイテムのシーケンスの文字列の値とノードの値を置換[461] える

これらのオペレーションのキーワード及び構文はこのセクションのサブセクションで説明されています。

8.5.2.1 ノードの削除

説明と構文
1つ以上のノードを削除します。

```
delete node nodeSequence
delete node nodeSequence
```

詳細

- *nodeSequence* 式は削除するノードのシーケンスを返します。選択されたノードは削除されるためマークされます。
- 単一 ノード または複数の ノード が使用されることができます。*nodeSequence* 内のアイテムの数に対するコレスポンデンスは必要ありません。

サンプル

```
for $i in /book/section return
delete nodes $i/@id
```

8.5.2.2 ノードの挿入

説明と構文
1つまたは複数のノードを指定されたターゲットノードの前、後、または内部に挿入する。

```
insert (node|nodes) items into targetNode
insert (node|nodes) items as first into targetNode
insert (node|nodes) items as last into targetNode
insert (node|nodes) items before targetNode
insert (node|nodes) items after targetNode
```

詳細

- *items* 式は、アイテムのシーケンスを返さなければなりません。キーボード **node|nodes** が使用されていますが、*items* は、非ノードアイテムのシーケンスである場合があります。
- *targetNode* 式は、単一のターゲットノードを指す必要があります。
- **into** キーボードが使用されていますが、*targetNode* は要素ノードまたはドキュメント要素ノードである必要があります。
- キーフレーズ **as first** または **as last** が使用されていますが、挿入は、最初または最後の子です。
- キーワード **into** は単独で仕様され、属性は既存の属性に追加され、要素は最初の子として挿入されます。
- キーワード **before** または **after** が使用されており、*targetNode* はあらゆる型であることができます。
- 属性が挿入される場合、既存の属性の名前を重複してはなりません。

サンプル

```
for $i in /book/section return
insert nodes (attribute id { 'somevalue' }, <newelement>some content including the
numbers "{ 1 to 3}"</newelement>)
into $i
```

8.5.2.3 ノードの名前の変更

説明と構文
要素、属性、または指示ノードの名前を変更する。

```
rename node targetNode as name
```

詳細

- 要素、属性、または処理命令であることのできる *targetNode* 式は単一ターゲットノードを指す必要があります。
- *name* 式は、QName または文字列を評価する必要があります。
- QName が作成されると、必須名前空間がローカルで宣言されます。

詳細

```
rename node /book/title as 'header-1'

rename node /book/title as QName("http://www.altova.com/xquf", "header-1")
```

8.5.2.4 ノードの置換

説明と構文
アイテムのシーケンスとノードを置き換えます。

```
replace node targetNode with items
```

詳細

- *targetNode* 式は、単一のターゲットノードを指す必要があります。
- *items* 式はアイテムのシーケンスを返す必要があります。このシーケンスはターゲットノードを置き換えます。
- 属性ノード以外、ターゲットノードはどの型のシーケンスとも置換えられることができます。
- 属性ノードは属性ノードと置き換えられます。下のサンプルを参照してください。

サンプル

```
replace node //hr with '<line/>'

for $i in //@height return
replace node $i with (attribute line-height{'12pt'})
```

8.5.2.5 ノードの値の置換

説明と構文
アイテムのシーケンスの文字列の値とノードの値を置き換えます。

```
replace value of node targetNode with items
```

詳細

- *targetNode* 式は、単一のターゲットノードを指す必要があります。
- *items* 式はアイテムのシーケンスを返す必要があります。
- ターゲットノードのコンテンツは *items* 式により返されたシーケンスの文字列の値と置き換えられます。これは、ターゲットノードはテキストノード1つのみを含むことを意味します。

サンプル

```
for $i in //title return
replace value of node $i with ('Draft Title')
```

8.5.2.6 fn:put 関数

fn.put 関数はXQuery 内蔵関数ライブラリの拡張子としてQuery Update Facility 1.0 により与えられています。（ **fn:** このセクションの名前空間のプレフィックスは、以下の名前空間にバインドされているものと見なされます: `http://www.w3.org/2005/xpath-functions`.)

```
fn:put($node as node(), $uri  as xs:string) as empty-sequence()
```

関数は、ドキュメントまたは要素を `$uri` により指定された場所に保管します。ファイルシステムまたはデータベースなどの外部ストレージシステムのリソースを作成するために呼び出されます。`fn:put` の外部効果は、XQuery のドメイン外で起こるため、実装定義済みです。目的は、`fn:put` がドキュメントノードにより呼び出され、エラーが指摘されなかった場合、次のクエリは、同じ URI を持つ `fn:doc` を呼び出して保管されたドキュメントにアクセスすることができます。

詳細は 仕様 を参照してください。

8.6 XQuery と XML データベース

XQuery ドキュメントを使えば、XML データベース（XML DB）への問い合わせを行うことができます。このXQuery 機能は、現在 IBM DB2 データベースでしかサポートされておりません。XQuery を使ったXML DB への問い合わせの仕組みは、(i) XQuery エンジンに対してDB 内部のXML へ問い合わせを行う通知と、(ii) DB 内部にあるXML データへのアクセス、により成り立っています。

この仕組みを実装するには、以下のようなステップを踏む必要があります。各ステップの詳細を以下に記します:

1. XQuery ドキュメントの設定[462] を行い、XQUERY キーワードをドキュメントの最初に挿入することでXML DB の問い合わせを行うようにする。.
2. アクティブなXQuery ドキュメントに対して、(情報ウィンドウから)DB のサポートを有効[462]にして、(クイック接続ダイアログを使って)DBへ接続する[462]。
3. XQuery ドキュメントにて、DB 固有のXQuery 拡張[462]を使用することで、DB データにアクセスし、XQuery オペレーションも行えるようにする。
4. XQuery ドキュメントの実行[462]をXMLSpy から行う。

XML DB の問い合わせを行うようXQuery ドキュメントを設定する

XQuery ドキュメントを使ってXML DB の問い合わせを行うには、XQuery ドキュメントを開き(または新しいXQuery ドキュメントを作成し)、XQUERY キーワードを、ドキュメントの最初に入力します。以下の例を参照ください。

```
XQUERY (: Retrieve details of all customers :)
declare default element namespace "http://www.altova.com/xquery/databases/db2";
<a> {db2-fn:xmlcolumn("CUSTOMER.INFO")} </a>
```

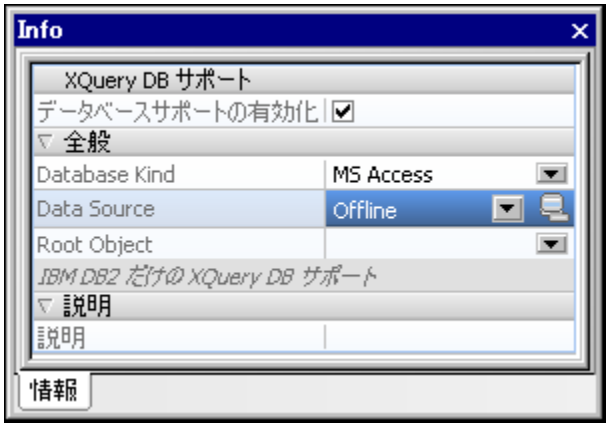
ドキュメントがオプションのxquery version 表現を使用する場合でも、XQUERY が必要です:

```
XQUERY xquery version "1.0"; (: Retrieve details of all customers :)
declare default element namespace "http://http://www.altova.com/xquery/databases/db2";
<a> {db2-fn:xmlcolumn("CUSTOMER.INFO")} </a>
```

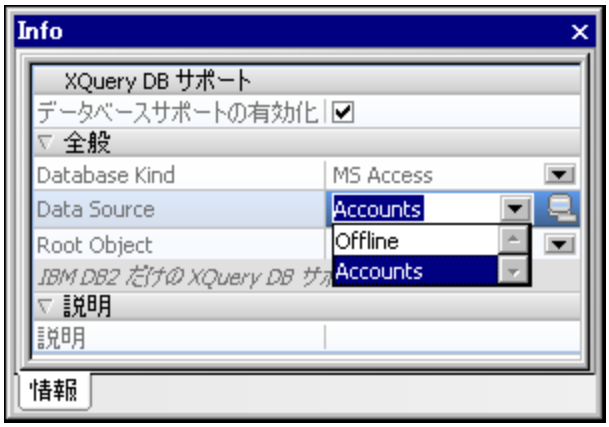
メモ XMLSpy の内蔵 XQuery エンジンがXQUERY キーワードを読むことにより、XML DB がアクセスの対象になることが示唆されます。その結果、XQUERY キーワードを含むXQuery ドキュメントを、XML DB を含まないXML ドキュメントに対して実行してもエラーとなります。

DB サポートを有効化し、DB へ接続する。

(下のスクリーンショットの様に)情報ウィンドウにあるデータベースサポートを有効化チェックボックスにチェックを入れることで、XQuery ドキュメントのDB サポートが有効になります。DB サポートは、各 XQuery ドキュメントに対して有効にする必要があり、XQuery ドキュメントを開き直したときにも再度設定する必要があることにご注意してください。



情報ウィンドウにて DB のサポートを有効にすると、接続ウィザードが起動され、データベースへの接続を行うことができます。現在のところ、IBM DB2 データベースのみがサポートされています。データベースへの接続方法については[データソースへ接続](847)、のセクションにて記述されています。データソースへの接続が既に存在する場合、これらの接続が(以下のスクリーンショットにある)情報ウィンドウの Data Source コンボボックスに表示され、データソースのリストに表示されているアイテムの1つを、アクティブな XQuery ドキュメントのデータソースとして選択することができます。情報ウィンドウでは、Root Object コンボボックスからルートオブジェクトを選択することもできます。



情報ウィンドウにある アイコンをクリックすることで、(DB への接続を行う)接続ウィザードにいつでもアクセスすることができます。

メモ XQuery ドキュメントを閉じると、DB への接続も同様に切断されます。その後 XQuery ドキュメントを開く際に、DB への接続を再度行う必要が生じます。

IBM DB2 固有の XQuery 言語拡張

XQuery ドキュメントでは IBM DB2 データベースからデータを取得するために、2つの IBM DB2 固有の関数を使用することができます:

- `db2-fn:xmlcolumn` により、列の検索やフィルタリングを行うこと無く、XML 列全体を取得することができます。
- `db2-fn:sqlquery` により、SQL `SELECT` ステートメントをベースにした値の取得を行ないます。

これらの関数を使用して取得された XML データは、標準的な XQuery コンストラクトにより開くことができます。以下の例を参照ください。

`db2-fn:xmlcolumn(`: この関数の引数は大文字/小文字で区別される文字列リテラルで、テーブル内にある XML 列の参照に使用されます。文字列リテラル引数は XML タイプの修飾された列名でなければなりません。列内にある全ての XML データがシーケンスとして関

数から返され、ここでは検索条件などは適用されません。以下の例では、CUSTOMER テーブル内の INFO (XML) 列のデータ全てが、トップレベルの<newdocelement> 要素内に戻されます:

```
XQUERY (: Retrieve details of all customers :)
declare default element namespace "http://www.altova.com/xquery/databases/db2";
<newdocelement> {db2-fn:xmlcolumn("CUSTOMER.INFO")} </newdocelement>
```

取得されたデータはXQuery コンストラクトを使って絞り込むことができます。以下の例では、CUSTOMER テーブルにある INFO (XML) 列から取得されたデータが、XQuery コンストラクトによりフィルタリングされ、トロント (Toronto) にある顧客データのみが取得されるようになっています。

```
XQUERY (: Retrieve details of Toronto customers :)
declare default element namespace "http://www.altova.com/xquery/databases/db2";
<newdocelement> {db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo[addr/city='Toronto']}
</newdocelement>
```

メモ 上の例では、各セルにある XML ファイルのドキュメント要素は customerinfo であり、db2-fn:xmlcolumn により得られた XML シーケンスのルートノードは customerinfo ノードの上にある abstract ノードと見なされます。

db2-fn:sqlquery:.この関数はSQL SELECT ステートメントを引数として受け取り、その結果を XML 値のシーケンスとして戻します。取得されたシーケンスはXQuery コンストラクトにより絞り込むこともできます。以下の例では、CUSTOMER テーブルにあるINFO 列のデータのうち、CID フィールドの値が1000 から1004 間にある行が取得されます。SQL ステートメントでは大文字/小文字の区別は行われませんが、XQuery では異なるものとして扱われることにご注意してください。

```
XQUERY (: Retrieve details of customers by Cid:)
declare default element namespace "http://www.altova.com/xquery/databases/db2";

<persons>
   {db2-fn:sqlquery("SELECT info FROM customer WHERE CID>1000 AND CID<1004")/
   <person>
      <id>{data(@Cid)}</id>
      <name>{data(name)}</name>
   </person>}
</persons>
```

上のXQuery ドキュメントは以下のような出力を生成します:

```
<persons xmlns="http://www.altova.com/xquery/databases/db2">
   <person>
      <id>1001</id>
      <name>Kathy Smith</name>
   </person>
   <person>
      <id>1002</id>
      <name>Jim Jones</name>
   </person>
   <person>
      <id>1003</id>
      <name>Robert Shoemaker</name>
   </person>
</persons>
```

以下の点にご注意してください:

- デフォルトの要素名前空間宣言(デフォルトのelement namespace)はXQuery 全体に対して適用され、XML ドキュメントや新たな要素の構築時のナビゲーションに利用されます。これは、XQuery セレクター name が、<default-element-namespace>:name に拡張され、構築された要素(例:persons)がデフォルトの名前空間にあることを意味します。

- SQL SELECT ステートメントでは大文字/小文字の区別が行われません。
- SELECT ステートメントの `WHERE` 句は、データベースアイテムを参照するもので、アクセスされている XML ファイルのノードを参照するものではありません。
- `db2-fn:sqlquery` 関数の後にある "/" は戻されたシーケンスの最初のアイテムを表わしており、このアイテムがコンテキストノードになります。

XQuery の実行

XQuery ドキュメントを実行するには、XSL/XQuery メニューにある XQuery を実行コマンドを選択します。他にも、Alt+F10 を押下し、XQuery を実行アイコン をクリックすることで、同様の操作を行えます。実行の結果は新しいドキュメント内に表示されます。

9 XSLT/XQuery デバッガー とプロファイラー

XMLSpy には、ドキュメントを速く作成する手助けをするXSLT/XQuery デバッガー[467] とXSLT/XQuery プロファイラー[492] が搭載されています。

これらの2 つの機能に関しては、このセクションのサブセクションで説明されています。

(C) 2015-2021 Altova GmbH

9.1 XSLT と XQuery デバッガー

Altova W eb サイト: XSLT デバッガー、XQuery デバッガー

XSLT and XQuery Debuggerを使ってXSLT スタイルシートやXQuery ドキュメントの試験やデバッグを行うことができます。XSLT and XQuery Debuggerのインターフェースにはお互いに連動するXSLT/XQuery ドキュメント、結果ドキュメント、そしてソースXML ドキュメントのビューが搭載されており、XSLT/XQuery ドキュメントをステップ毎に実行することができます。対応する出力もステップごとに生成され、ソースXML ファイルが表示されていれば、XML ファイル内の対応する箇所が各ステップでハイライトされます。インターフェース内にあるウィンドウにはデバッグ情報が表示されます。

XSLT and XQuery Debuggerは常にデバッガーセッション内で起動されます。デバッガーセッションは以下のうちのどれかです:

- 内蔵のAltova XSLT 1.0 エンジンを使用するXSLT 1.0
- 内蔵のAltova XSLT 2.0 エンジンを使用するXSLT 2.0
- 内蔵のAltova XSLT 3.0 エンジンを使用するXSLT 3.0
- 内蔵のAltova XQuery 1.0 エンジンを使用するXQuery 1.0
- 内蔵のAltova XQuery 3.1 エンジンを使用するXQuery 3.1

メモ　XQuery Update Facility 1.0 とXQuery Update Facility 3.0 は現在 XSLT and XQuery Debugger でサポートされていません。

開かれるデバッガーセッションの種類は、デバッガーセッションが開かれたドキュメント（*アクティブドキュメント* または*アクティブファイル*）の種類により自動的に決定されます。XSLT デバッガーセッションはXSLT ファイルから開かれ（どのバージョンかは、XSLT スタイルシート（XSLT 1.0 には`"1.0"` 、XSLT 2.0 には`"2.0"` 、XSLT 3.0 には`"3.0"`）の`xsl:stylesheet` や`xsl:transform` 要素にある`version` 属性の値により変化します）、XQuery デバッガーセッションはXQuery ファイルから開かれます。アクティブファイルがXML ファイルであれば、XML ファイル上で動作させるのがXSLT かXQuery かにより、デバッガーセッションの種類が決定されます。前者の場合選択はスタイルシートがXSLT 1.0、XSLT 2.0、またはXSLT 3.0 スタイルシートであるかにより決定されます。

この情報は下のテーブルにまとめられています。

アクティブなファイル	関連したファイル	デバッグ セッション
XSLT 1.0	XML; (必須)	XSLT 1.0 (内蔵のAltova XSLT 1.0 エンジン使用)
XSLT 2.0	XML; (必須)	XSLT 2.0 (内蔵のAltova XSLT 2.0 エンジン使用)
XSLT 3.0	XML またはJSON; (必須)	XSLT 3.0 (内蔵のAltova XSLT 3.0 エンジン使用)
XQuery 1.0	XML; (オプション)	XQuery 1.0 (内蔵のAltova XQuery 1.0 エンジン使用)
XQuery 3.1	XML またはJSON; (オプション)	XQuery 3.1 (内蔵のAltova XQuery 3.1 エンジン使用)
XML	XSLT 1.0, 2.0, または3.0, またはXQuery 1.0 または3.1; (必須)	XSLT 1.0、XSLT 2.0、3.0 またはXQuery 1.0 または3.1 。XSLT スタイルシートの`xsl:stylesheet` (または`xsl:transform`) 要素にある`version` 属性の値により、XSLT スタイルシートを決定。

3つのAltova エンジンに関する詳細は、付録[1973]内のXSLT とXQuery エンジン情報[1974]を参照してください。

Altova RaptorXML 2021 を使用したXSLT とXQuery タスクの自動化

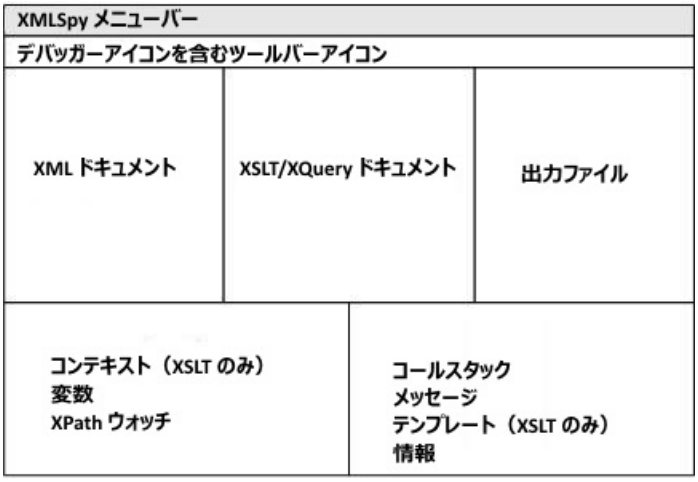
Altova RaptorXML Server は、Altova のXML バリデーター、XSLT 1.0、XSLT 2.0、XSLT 3.0、そしてXQuery 1.0 と 3.1 エンジンを搭載した無料アプリケーションです。コマンドライン、COM インターフェース、Java プログラム、そして .NET アプリケーションからXML ドキュメントの検証を行ったり、XSLT スタイルシートを使ったXML ドキュメントの変換や、XQuery ドキュメントの実行を行うことができます。

RaptorXMLを使えばXSLT ならびにXQuery 処理を自動化することができます。例えば、RaptorXMLを呼び出して、XML ドキュメントのセットを変換し、XQuery ドキュメントのセットを実行するバッチファイルを作成することができます。詳細はRaptorXML Server ドキュメントを参照ください。

9.1.1 メカニズムとインターフェース

XSLT ならびにXQuery ファイルのデバッグにて使用されるXSLT and XQuery Debugger の概要を以下に示します。(「XSL/XQuery | デバッガーの開始/実行」コマンドにより両方とも開始されるものですが) デバッグセッションとデバッガーは異なるものであることご注意してください。まずはデバッグセッションを開始して、デバッガーによりXSLT またはXQuery ドキュメントをステップ実行していく必要があります。

- 「XSL/XQuery | デバッガーの開始/実行」コマンドによりデバッグセッションを開きます。アクティブファイルを元に適切なセッション(XSLT 1.0、XSLT 2.0、XSLT 3.0、XQuery 1.0 またはXQuery 3.1) XSLT とXQuery デバッガー 467 を参照ください)。XSLT and XQuery Debuggerはテキストビューならびにグリッドビューでのみ動作します。アクティブドキュメントのビューがテキストビューでもグリッドビューでもない状態でデバッグセッションを開始した場合、XSLT and XQuery Debuggerのデフォルトビューであるテキストビューにビューを変更する許可を求められます。デバッガー設定ダイアログ 473 にて、常に許可を行うよう設定することもできます。
- 「XSL/XQuery」メニューにある「ステップイン」、「ステップアウト」、そして「ステップオーバー」コマンドを使用することでXSLT またはXQuery ドキュメントをステップごとに実行することができます。ここでデバッガーの開始/実行コマンドをクリックすると、ブレークポイントに到達するまで変換または実行を行ないます。ブレークポイントが1つもセットされていない場合、デバッグの結果を表示すること無く、1ステップで全ての変換や実行を行なうことになります。このセッションにXML ファイルが割り当てられている場合、XML ファイルの対応する箇所がハイライトされます。更に、対応するステップにより生成された結果は、結果ファイルに蓄えられ、ステップごとに結果ドキュメントが更新されます。この方法により、XSLT やXQuery ファイルがステートメントごとに行う動作を確認、分析することができます。



上で示されている3つのドキュメント(XML、XSLT/XQuery、出力)の代わりに、2つのドキュメント(XSLT/XQuery、出力)、またはどれか1つのドキュメントビューを選択することもできます。

- デバッグセッションが開いている間、インターフェースの情報ウィンドウには変換/実行に関する様々な情報が表示されます(変数、XPath Watch、コールスタック、メッセージなど)。
- デバッグセッションが開いている間、デバッガーを停止(デバッグセッションは停止していないことに注意)して、ドキュメントを修正することもできます。XMLSpy で使用することのできる編集機能は、デバッグセッション中にファイルを編集する際にも全て使用することができます。デバッガーが停止している間でも XSLT and XQuery Debugger のインターフェースは開かれたままになり、情報ウィンドウに表示されている情報の参照を行うことができます。デバッグセッションにてデバッガーを停止させた後、同じデバッグセッションにて(XSLT/XQuery ドキュメントの初めから)デバッガーを再度開始することもできます。
- XSLT ファイルにブレークポイントをセットすることで、選択された箇所で割り込みを発生させることができます。この機能により、XSLT または XQuery ドキュメント内のステップを手動で個別に実行する必要が無くなり、デバッグセッションの効率を大幅に向上させることができます。
- XSLT にはトレースポイントをセットすることもできます。トレースポイントがセットされた命令にて処理が実行された場合、その命令の値が出力されます。
- デバッグセッションを停止します。この操作により XSLT and XQuery Debugger インターフェースが終了し、元の XMLSpy 環境へ切り替わります。情報ウィンドウにある情報は利用できなくなります。セットされたブレークポイントとトレースポイントに関する情報は、そのファイルが閉じられるまで保持されたままになります(従って、新たなデバッグセッションが開始された場合、新たに表示されるデバッグセッションにもブレークポイントやトレースポイントは表示されます)。

メモ: デバッガーアイコンのあるデバッガーツールバーは、デバッグセッションの開始時に自動的に表示されます。

9.1.2 コマンドとツールバーアイコン

XSL/XQuery メニューやツールバーアイコンからデバッグコマンドへアクセスすることができます。デバッガーアイコンは、デバッグセッションが開始されたときに、ツールバーに自動的に表示されます。以下にデバッガーアイコンを示します。

デバッガーの開始/実行 (Alt+F11)

XSLT/XQuery ドキュメントの処理を開始、または最後まで継続します。ブレークポイントがセットされている場合、そこで処理が停止します。デバッグセッションが開始されていない場合、このボタンによりデバッグセッションが開始され、最初に処理されるノードで停止します。セッションが動作中の場合、XSLT/XQuery ドキュメントはドキュメントの最後または次のブレークポイントまで処理されます。トレースポイントがセットされている場合、(処理命令がそこにたどり着いたときに)各トレースポイントにある命令の値がトレースウィンドウに表示されます。

アクティブなドキュメントのみを表示

デバッガーインターフェースにて現在アクティブなドキュメントのウィンドウを最大化します。

XSLT/XQuery と出力結果を表示

XSLT または XQuery ドキュメントウィンドウを表示し、XML ドキュメントを隠します。

XML、XSLT/XQuery、出力結果を表示

XML、XSLT/XQuery、そして出力ドキュメントを表示します。これはXMLドキュメントがデバッグセッションに割り当てられた時の標準のビューです。

デバッガーを停止

デバッガーを停止します。デバッグセッションの停止とは異なることにご注意してください。デバッグを行っている最中にドキュメントの編集を行ったり、同じデバッグセッションにて別のファイルを使用するときに便利な機能になります。デバッガーを停止させた後は、XSLT/XQueryドキュメントの最初からデバッガーを再度開始させることができます。

ステップイン(F11)

スタイルシートにある全てのノードやXPath 条件式の内部を処理します。このコマンドはデバッガーを再度開始するためにも使用することができます。

ステップオーバー(Ctrl+F11)

現在のノードから同じレベルの次のノードへ移るか、1つ上のレベルにある次のノードに移ります。このコマンドはデバッガーを再度開始するためにも使用することができます。

ステップアウト(Shift+F11)

現在のノードから、次の兄弟ノードに移動するか、1つ上のレベルにある次のノードに移ります。

現在の実行ノードを表示

XSLT/XQuery ドキュメントにある現在の実行ノードを表示、選択し、XML ドキュメント内の対応するノードをハイライトします。他のタブを選択し、XSLT スタイルシートまたはXML ファイルのコードを選択した後、以前作業していた場所を特定するのに便利な機能です。

デバッガーを再起動

出力ウィンドウに表示されている内容を破棄し、現在選択されているファイルでデバッグセッションを再度開始します。

ブレークポイントを挿入/削除(F9)

現在のカーソル位置でブレークポイントを挿入、または削除します。インラインブレークポイントはXSLT/XQuery とXML の両方のノードに対して定義することができ、処理がどこで停止するかを決定することができます。赤色の点線がブレークポイントをセットしたノードの上に表示されます。終了ノードにブレークポイントをセットすることはできず、XSLT ドキュメントの属性に対するブレークポイントは無視されます。このコマンドは右クリックして表示されるコンテキストメニューからも利用することができます。

トレースポイントを挿入/削除(Shift+F9)

現在のカーソル位置でトレースポイントを挿入、または削除します。インライントレースポイントはXSLT ドキュメントのノードに対して定義することができます。デバッグ中にトレースポイントがセットされたステートメントにたどり着くと、そのステートメントの結果がトレースウィンドウに出力されます。青色の点線がトレースポイントをセットしたノードの上に表示されます。終了ノードにトレースポイントをセットすることはできません。このコマンドは右クリックして表示されるコンテキストメニューからも利用することができます。

ブレークポイントの有効/無効化(Ctrl+F9)

このコマンド(ツールバーアイコンにはありません)により、既に定義されたブレークポイントを有効/無効化することができます。ブレークポイントが無効化されると、赤色のブレークポイントが灰色で表示されます。デバッガーは無効化されたブレークポイントでは停止しません。ノード名までカーソルを移動させ、ブレークポイントの有効/無効化コマンドをクリックして、ブレークポイントを有効/無効化することができます。このコマンドはブレークポイントのある位置で右クリックすることでも使用可能です。

トレースポイントの有効/無効化(Shift+Ctrl+F9)

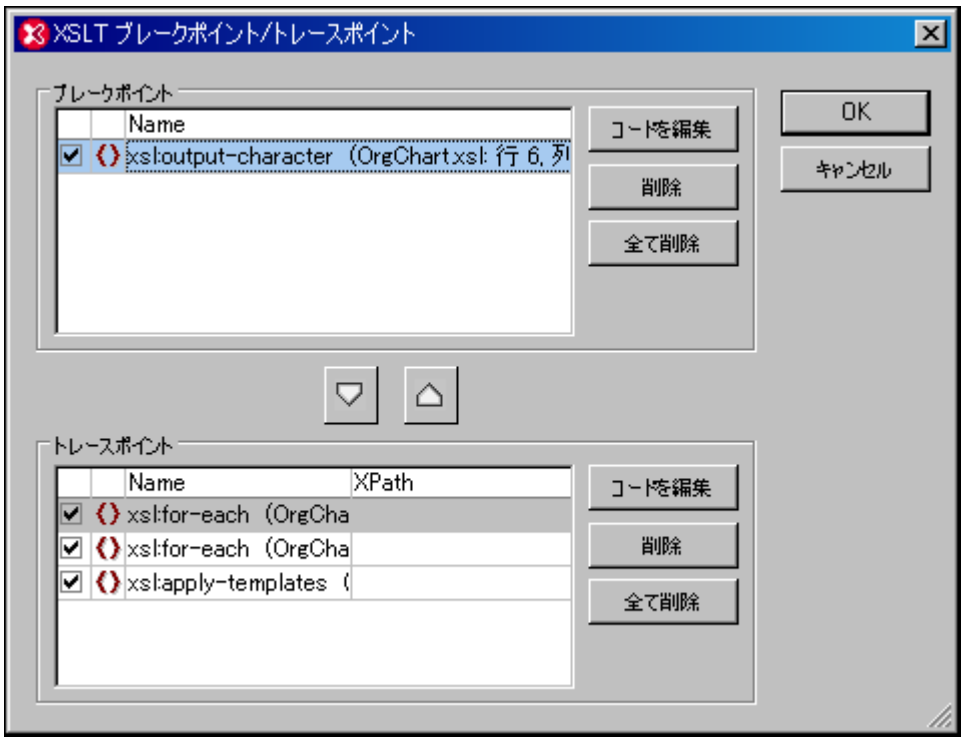
このコマンド(ツールバーアイコンにはありません)により、既に定義されたトレースポイントを有効/無効化することができます。トレースポイントが無効化されると、青色のトレースポイントが灰色で表示されます。無効化されたトレースポイントにおいて出力は行われません。ノード名までカーソルを移動させ、トレースポイントの有効/無効化コマンドをクリックして、トレースポイントを有効/無効化することができます。このコマンドはトレースポイントのある位置で右クリックすることでも使用可能です。

デバッグセッションの終了

デバッグセッションを終了し、デバッグセッションを開始する直前の、通常のXMLSpy ビューに戻ります。デバッグセッションで開かれた出力ドキュメントを開かれたままにするかは、XSLT/XQuery デバッガー設定ダイアログ473で設定することができます。

XSLT ブレークポイント/トレースポイントダイアログ

このコマンドによりXSLT/XQuery ブレークポイントならびにトレースポイントダイアログが開かれ、デバッグセッションにあるファイルで現在定義されている(無効化されているものも含めて)全てのブレークポイント/トレースポイントが表示されます。

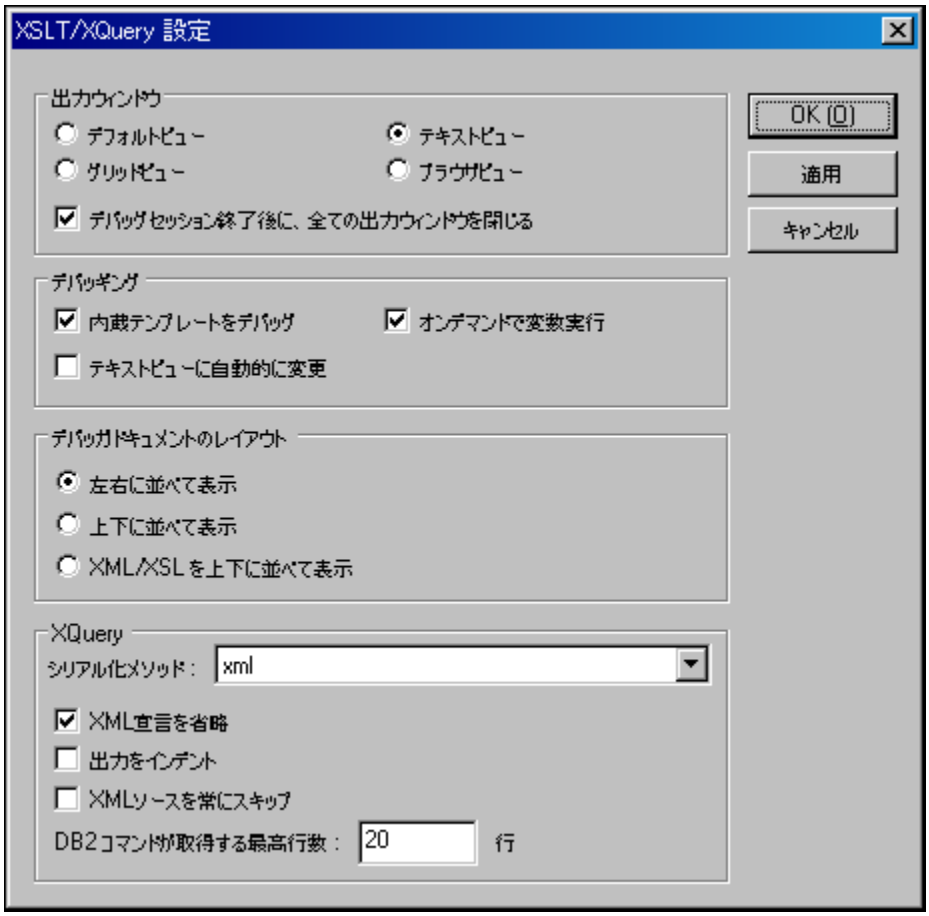


チェックボックスはブレークポイント/トレースポイントが有効になっているか(チェックされていれば有効)を表します。ブレークポイント/トレースポイントを選択し、「削除」ボタンを押すことで、ブレークポイント/トレースポイントの削除を行え、「全て削除」ボタンを押すことで、全てのブレー

クポイントまたはトレースポイントを削除することができます。「コードを編集」ボタンにより選択されたブレークポイント/トレースポイントへ移動します。

9.1.3 デバッガー設定

XSLT and XQuery Debugger 設定ダイアログでは、全てのデバッグセッションに適用されるデバッグならびに出力オプションを設定することができます。設定ダイアログを開くには、「XSL/XQuery | XSLT/XQuery の設定」を選択するか、ツールバーにある アイコンをクリックしてください。このセクションでは設定の詳細について記述します。



出力ウィンドウ

出力ドキュメントウィンドウを（デフォルト、テキスト、グリッド、またはブラウザービューから）指定します。デフォルトビューとはオプションダイアログ（「ツール| オプション」）のファイルの型のセクションにある（.xslt や .xq といった）ファイルの型にて選択されているビューのことです。XSLT 変換では、出力ファイルの型はXSLT ファイルにて定義されます。XQuery の実行では、出力ファイルの種類は、このダイアログの XQuery 設定で指定するシリアル化メソッドにより決定されます（下記参照）。

全ての出力ウィンドウを閉じるオプションにより、デバッグセッション中に開かれていた出力ドキュメントを、デバッグセッションが終了した後にも開いたままにしておくか指定することができます。

デバッグ

内蔵テンプレートをデバッグ設定により、適切な箇所で内蔵テンプレート内のコードに、「**ステップイン**」することができるようになります。これはテンプレートタブの中にあるテンプレートエントリのクリックや、コールスタックにより「**表示**」される内蔵テンプレートファイルとは関係ありません。

XSLT はテキストビューまたはグリッドビューでのみ動作します。テキストビューに自動的に更新するオプションにチェックを入れておくと、デバッグを行うドキュメントがテキストビューやグリッドビューで表示されていない場合に、そのビューが自動的にテキストビューに切り替わります（XQuery デバッガーはテキストビューでのみ動作します）。オンデマンド変数実行チェックボックスがチェックされていれば、変数が呼び出されたときに変数の定義にステップインすることができるようになります。このオプションにチェックが入れられていない場合、変数の呼び出し場所に処理が到達しても、変数の定義へのステップインは行われず、次のステップへ移動します。

デバッガードキュメントのレイアウト

デバッガードキュメントとはデバッガーで開かれているドキュメントのことです。ここではこれらドキュメントを、左右、上下、またはXML/XSLT ドキュメントを上下に並べて配置し、結果ドキュメントが左右になるような配置をすることができます。

9.1.4 デバッグセッションの開始

デバッグセッションを開始する最も単純な方法は、単一のXSLT、XQuery、またはXML ファイルから開始するというものです。必要な関連ファイル（関連するファイルのテーブル 467 を参照）が既にアクティブなファイルに関連付けられている場合、デバッグセッションがすぐに開始されます。そうでない場合は必要な関連ファイルを選択するよう促されます。XQuery ファイルはXML ファイルの割り当てを含まず、必要ともしないので、XQuery ファイルからXQuery デバッグセッションを開始するたびにオプションのXML ファイル割り当てを促さないように設定することができます。

定義済みの割り当て

定義済みの割り当てはXSLT デバッグセッションでのみ有効になるもので、アクティブなファイルに対して既に割り当てファイルが存在する場合に参照されます。XML またはXSLT ファイルにて割り当てを行うには、以下の操作を行ってください:

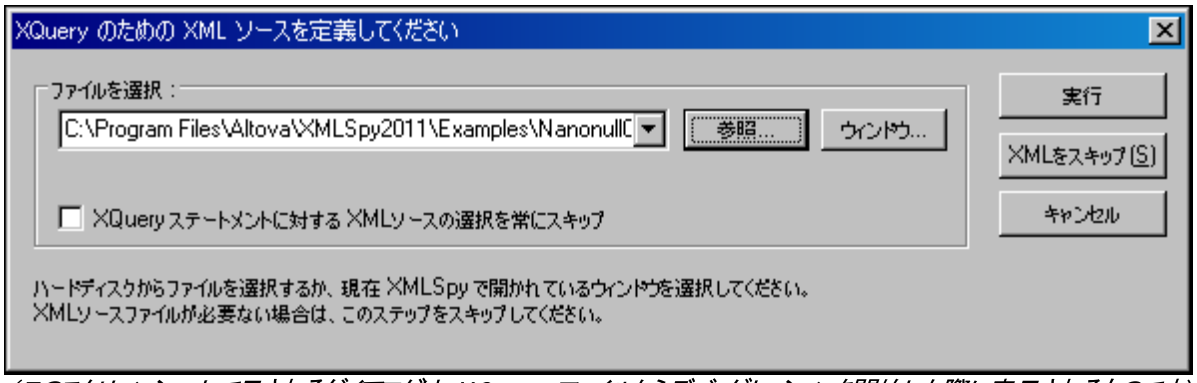
- XML/JSON ファイル: ファイルを開き、「**XSL/XQuery | XSL の割り当て**」からXSLT ファイルを選択します。
- XSLT ファイル: ファイルを開き、「**XSL/XQuery | サンプルXML ファイルの割り当て...**」からXML ファイルを選択します。

「**XSL/XQuery | デバッガーの開始/実行**」を選択すると、デバッグセッションが直接開始されます。アクティブなファイルに対してファイルの割り当てを行うよう促されることはありません。

直接の割り当て

アクティブなファイルに対して定義済みの割り当てが存在しない場合、割り当てを行うよう促されます。「**XSL/XQuery | デバッガーの開始/実行**」を選択すると、以下のような動作が発生します:

- XML ファイルの場合：XSLT またはXQuery、XQuery Update ファイルを選択するよう促されます。
- JSON ファイルの場合： XSLT またはXQuery、XQuery Update ファイルを選択するよう促されます。
- XSLT ファイルの場合：XML ファイルを選択するよう促されます。
- XQuery ファイルの場合：XML ファイルを選択するオプションが与えられます。スキップすることもできます。



（このスクリーンショットで示されるダイアログは、XQuery ファイルからデバッグセッションを開始した際に表示されるものです）

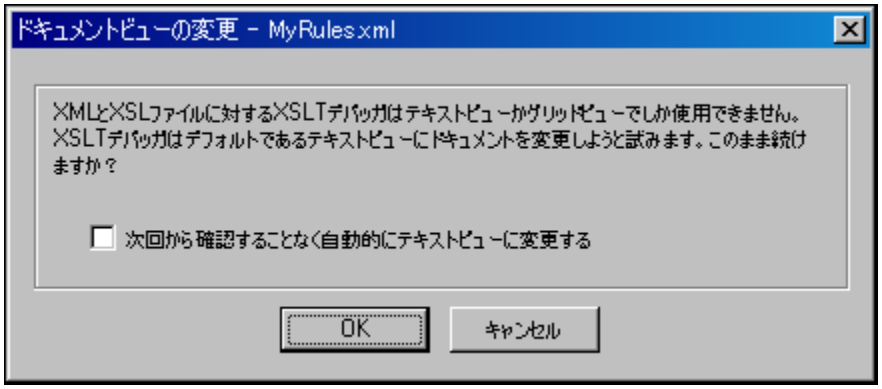
必要な割り当てファイルを選択するか、オプションの割り当てをスキップすると、デバッグセッションが開始されます。

その他のファイル割り当てを行う方法

プロジェクト プロパティダイアログでは、定義済みの割り当てを行うことができます。「**プロジェクト | プロパティ**」を選択し、「**使用する XSL / 使用する XML**」チェックボックにて必要なファイルを割り当ててください。

デバッガービュー

XSLT and XQuery Debuggerはテキストビューおよび拡張グリッドビューでのみ動作します。お使いの XML または XSLT ファイルが、テキストビューまたはグリッドビュー以外のビューで開かれている場合、または SPS ファイルが XML ファイルに割り当てられている場合、デバッグセッションの起動時に以下のダイアログが表示されます。



「**OK**」をクリックすることで、ドキュメントがテキストビューで開かれます。XQuery ファイルは常にテキストビューで表示されることにご注意してください。

9.1.5　情報ウィンドウ

デバッグセッション中、XSLT and XQuery Debuggerのインターフェースに表示される情報ウィンドウには XSLT 変換または XQuery 実行に関する様々な情報が表示されます。この情報を XSLT や XQuery ファイルのデバッグを行う際に利用することができます。

XSLT デバッグセッションには8つの情報ウィンドウがあり、XQuery デバッグセッションには5つの情報ウィンドウが与えられます。これらウィンドウはデフォルトで2つのグループに分けられ、XSLT and XQuery Debuggerインターフェースの下部に配置されます(以下の図を参照)。このセクションでは、これらウィンドウと表示される情報に関する説明を行ないます。

XSLT and XQuery Debuggerインターフェースのデフォルトレイアウト。

情報ウィンドウにある最初のグループは、以下のタブとして単一のウィンドウに表示されます。

- コンテキスト[476] (XSLT デバッグセッションのみ)
- 変数[477]
- XPath-Watch[478]

情報ウィンドウにある第2のグループは、以下のタブとして単一のウィンドウに表示されます。

- コールスタック[478]
- メッセージ[479]
- テンプレート[480] (XSLT デバッグセッションのみ)
- Info[480]
- トレース[481]

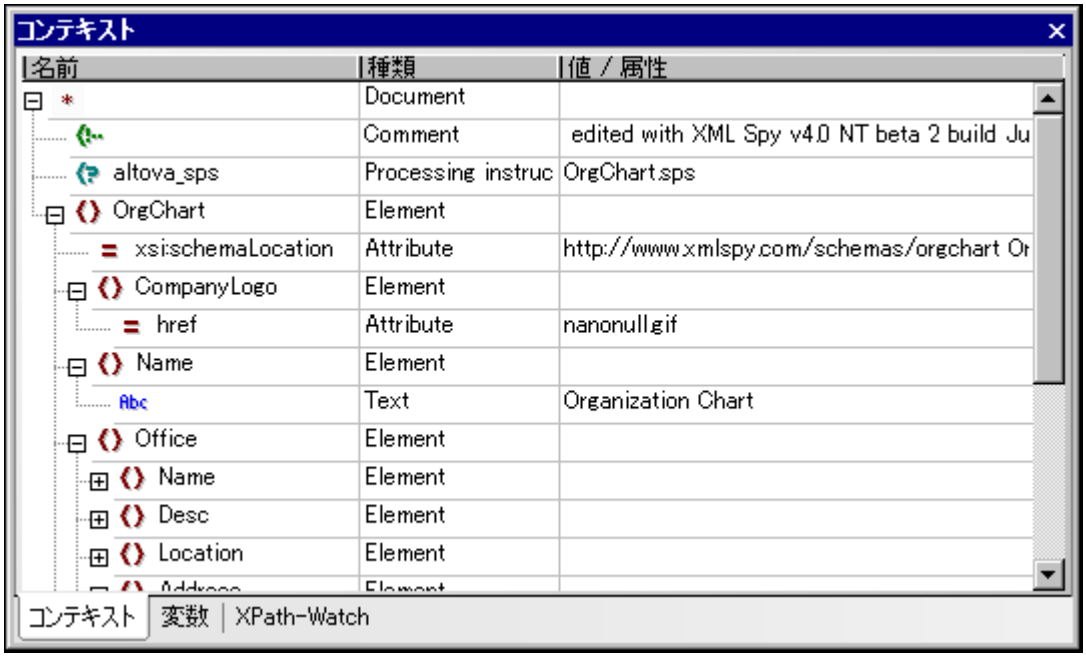
デフォルトのレイアウトでは2つのウィンドウグループが存在し、各ウィンドウには様々なタブが含まれています。同時にアクティブにすることのできるタブは1つのみです。例えば、最初のウィンドウグループにて変数に関する情報を確認するには「変数」タブをクリックします。これにより変数情報ウィンドウが表示され、コンテキストならびにXPath-Watch 情報ウィンドウが隠されることになります。タブによってはナビゲーションを行うためのツールとしても使えることに注目してください。アイテムをクリックすることで、XML、XSLT、またはXQuery ファイル内の対応するアイテムに移動することができます。詳細は各情報ウィンドウのドキュメント(コンテキスト[476]、コールスタック[478]、テンプレート[480])を参照ください。

これら2つの情報ウィンドウグループは、ウィンドウの境界線をドラッグすることで大きさを変更することができます。タブ名をクリックして、ウィンドウの外にドラッグすることで、各情報ウィンドウを切り離すこともできます。ウィンドウのタイトルバーを、ウィンドウグループのタイトルバーにドラッグすることで、グループに加えることもできます。変更したレイアウトをデフォルトのレイアウトにリセットする方法は無いことに注意してください。

9.1.5.1 コンテキストウィンドウ

コンテキストウィンドウはXSLT デバッグセッションでしか利用できません。XQuery デバッグセッションでは表示されません。

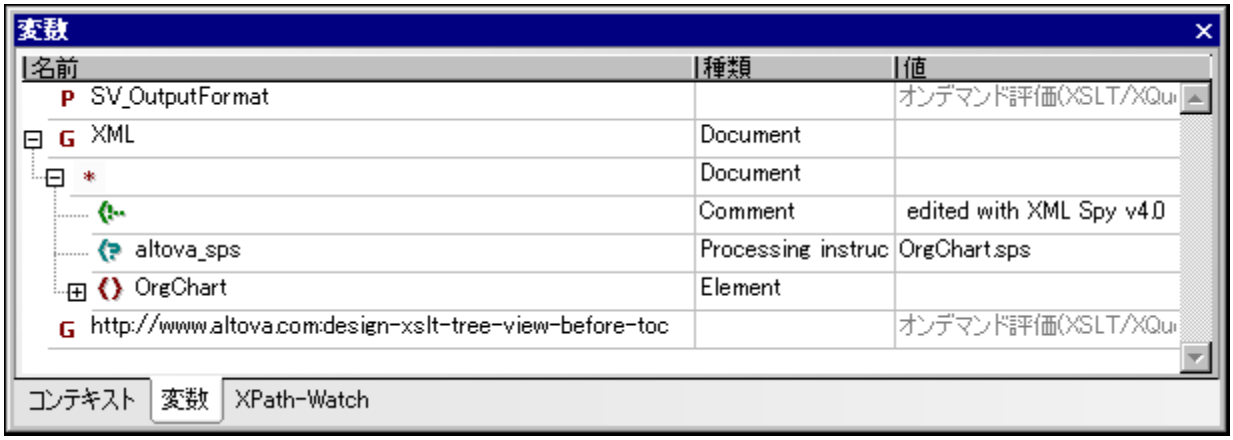
XSLT スタイルシートの処理中、プロセッサーのコンテキストは、常に(ノードやアトム値の)シーケンスにマッチするテンプレート内にあります。コンテキストウィンドウには、現在処理中のコンテキストである、ノード、シングルノード、または(文字列のような)アトム値のシーケンスが表示されます。コンテキストアイテムの種類に従い、値や属性が表示されます。例えばコンテキストアイテムが要素であれば、その要素の属性が表示され、属性またはテキストノードであれば、その値が表示されます。



コンテキストウィンドウのエントリをクリックすることで、XML ドキュメント内にあるそのアイテムが表示されます。XML ドキュメントがインターフェースに表示されていない場合、そのXML を表示するウィンドウが開かれます。

9.1.5.2 変数ウィンドウ

変数ウィンドウはXSLT ならびにXQuery デバッグセッションの両方で利用することができます。XSLT/XQuery ドキュメントのスコープ内で使用される変数やパラメーターが、値と共に表示されます。

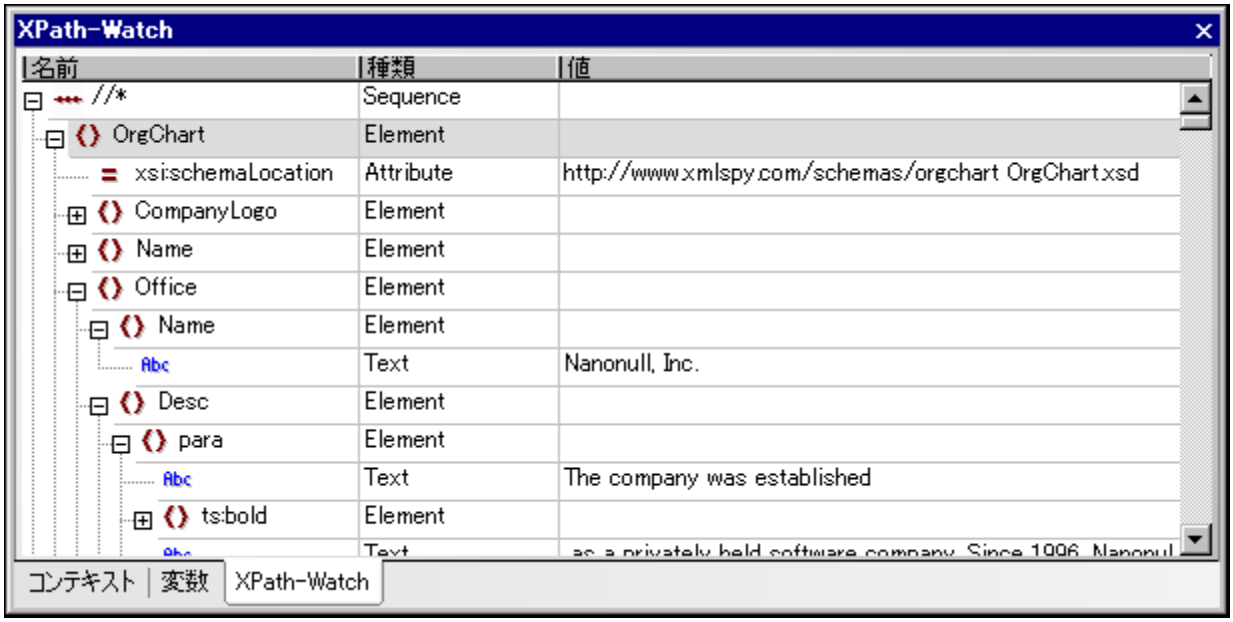


パラメーターは P と共に表示され、(スタイルシートのトップレベルで宣言される) グローバル変数は G と共に表示されます。(XSLT テンプレート内で宣言される) ローカル変数は L で表示されます。変数やパラメーターの値の種類は値フィールドのアイコンにより表示されます。以下の型が区別されます: ノードセット、ノードフラグメント、文字列、数値、Boolean。

9.1.5.3 XPath-Watch ウィンドウ

XPath-Watch ウィンドウは XSLT ならびに XQuery デバッグセッションにて利用可能です。

このウィンドウにより、コンテキスト上で評価を行ないたい XPath 条件式の入力を行うことができます。XSLT ドキュメントのステップを進めていくうちに、その時点のコンテキストで XPath 条件式が評価され、値列に結果が表示されます。



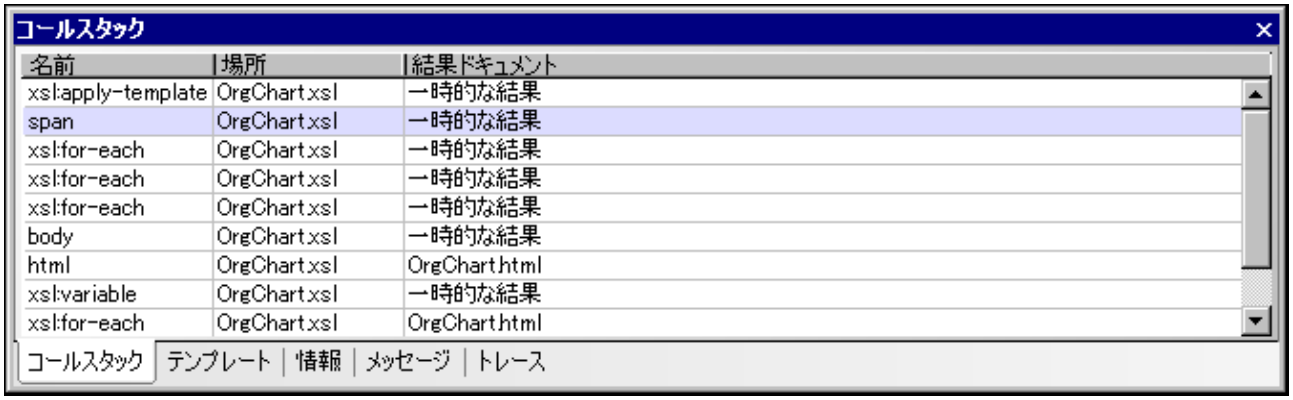
XPath 条件式を入力するには、名前列のテキストフィールドをダブルクリックして、XPath を入力します。他にも、ファイルから XPath 条件式をドラッグして、XPath-Watch ウィンドウにドロップすることもできます。XSLT スタイルシートの XSLT バージョンに合致する XPath のバージョンで使用される条件式を使用してください (XSLT 1.0 では XPath 1.0 が使用され、XSLT 2.0 では XPath 2.0 が使用され、XSLT 3.0 では XPath 3.0 が使用されます)。

注意: XML ファイルまたは XSLT ファイルにて名前空間が使用されている場合、XPath 条件式でも、適切な名前空間プレフィックスを使用する必要があります。

9.1.5.4 コールスタックウィンドウ

コールスタックウィンドウは XSLT ならびに XQuery 両方のデバッグセッションにて利用可能です。

コールスタックウィンドウには、それまで処理した XSLT テンプレートと命令が、リストの最上部にある現在のテンプレートと共に表示されます。



ウィンドウのエントリをクリックすることで、選択されたXSLT テンプレートや命令がXSLT ドキュメントウィンドウ内に表示されます。内蔵のテンプレートを参照するテンプレートや命令をクリックすると、全ての内蔵関数を表示する新たなウィンドウが表示され、選択されたテンプレートがハイライトされます。

9.1.5.5 メッセージウィンドウ

メッセージウィンドウはXSLT とXQuery 両方のデバッグセッションで利用することができます。

XSLT 1.0、XSLT 2.0、およびXSLT 3.0

XSLT デバッグセッションのメッセージタブには、エラー、`xsl:message` 命令、またはデバッグ中に発生したエラーメッセージが表示されます。

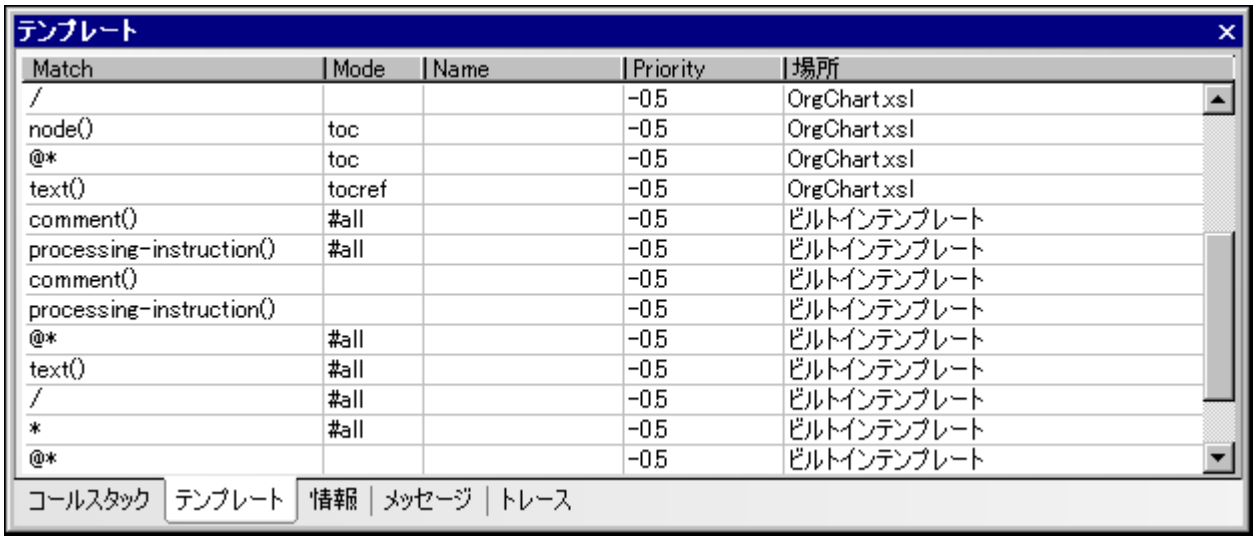
XQuery

XQuery デバッグセッションでは、メッセージウィンドウにエラーメッセージが表示されます。

9.1.5.6 テンプレートウィンドウ

(以下のスクリーンショットにある) テンプレートウィンドウはXSLT デバッグセッションでのみ表示され、XSLT デバッグセッションでは表示されません。

テンプレートウィンドウには、内蔵のテンプレートや名前付きテンプレートなどの、様々なXSLT スタイルシートが表示されます。マッチしたテンプレートは、ノードごとにまとめられて表示されます。名前付きテンプレートは名前順に表示されます。両方の種類のテンプレートにおいて、テンプレートの名前、モード、優先度、そして場所が表示されます。



上のスクリーンショットではXSLT スタイルシートにある3 つのテンプレートが表示されています: ドキュメントノード / にマッチするテンプレート、やテンプレート `n1:italic` と`n1:bold` ノードにマッチするテンプレートです。その他のテンプレートは(場所のフィールドにて示されている通り) 内蔵されている(ビルトインの) テンプレートです。

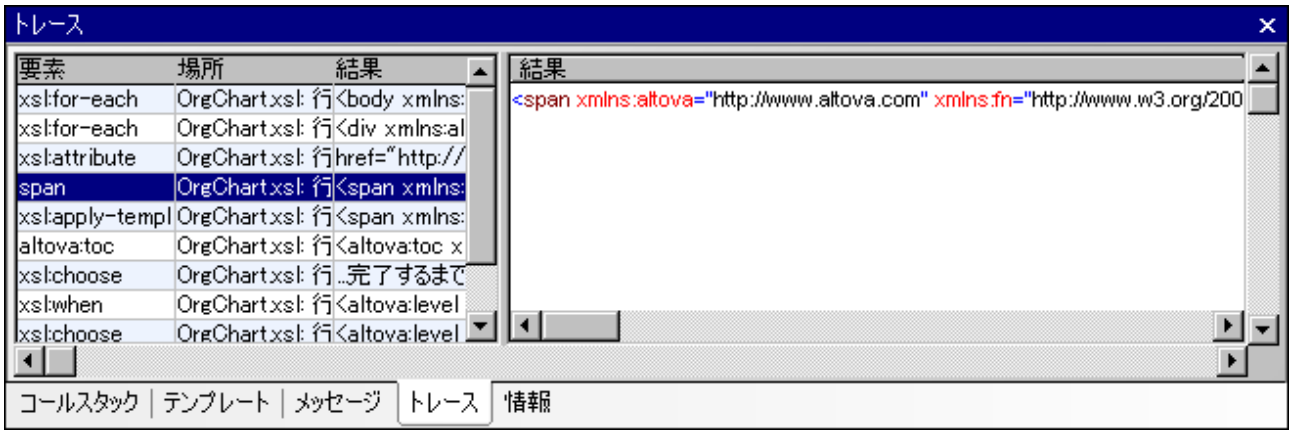
このウィンドウのエントリをクリックすることで、XSLT ドキュメントウィンドウにあるテンプレートがハイライトされます。内蔵のテンプレートをクリックすると、全ての内蔵テンプレートが表示されるウィンドウが新たに開かれ、選択したテンプレートがハイライトされます。

9.1.5.7 情報ウィンドウ

情報ウィンドウはXSLT とXQuery 両方のデバッグセッションで使用することができます。現在のデバッグセッションに関するメタ情報が表示されます。この情報の中には使用されているデバッガーや、ソースならびに出力ドキュメント、そしてデバッガーのステータスなどが表示されます。

9.1.5.8 トレースウィンドウ

トレースウィンドウはXSLT とXQuery 両方のデバッグセッションで利用することができます。



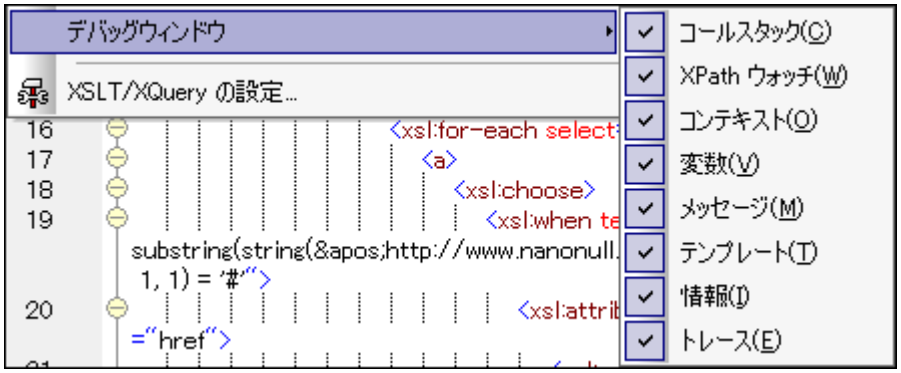
トレースウィンドウにはトレースポイントがセットされた要素、XSLT スタイルシート上の場所、そして要素が実行された際に生成される結果が表示されます。ウィンドウの左側にある行をクリックすることで、完全な結果が右側に表示されます。

9.1.5.9 情報ウィンドウの配置

情報ウィンドウをXSLT and XQuery Debuggerインターフェースの内部に配置することもできます。ウィンドウはインターフェースにドッキングさせることも、固定させないでおくことも、更に別のウィンドウにて1つのペインとして配置することも可能です。以下のメカニズムを利用して、ウィンドウの配置を行うことができます。

メニュー

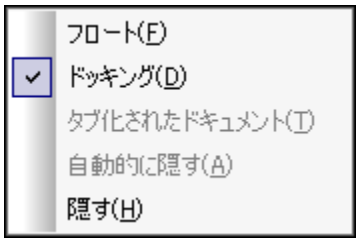
「XSL/XQuery」メニューにて、カーソルを「デバッグウィンドウ」に移動すると、情報ウィンドウのリストが表示されます。メニューアイテムをクリックすることでウィンドウの表示、非表示を切り替えることができます。



ウィンドウ表示のオン、オフを切り替えることができます。

コンテキストメニュー

ウィンドウタブ、または、タイトルバーを右クリックすることによりコンテキストメニューにアクセスすることができます。



目的のオプションをクリックすることで、ウィンドウを切り離し、ドッキングすることができます。

ドラッグアンドドロップ

タブ、または、タイトルバーからウィンドウをドラッグし、希望する場所に配置することができます。

更に、ウィンドウを他のウィンドウ、または、インターフェイスにウィンドウをドラッグすると表示される配置コントロールを使用してドックすることができます:

- ウィンドウを他のウィンドウ上にドラッグすると、配置コントロールが表示されます(*してのスクリーンショット参照*)。ウィンドウを別のウィンドウにドラッグすると、(以下のスクリーンショットにある)丸型の配置コントロールが表示されます。このコントロールは5 つの部分に分かれており、これらのうちのどれかでマウスボタンを放すと、ドラッグしていたウィンドウが「**ターゲットとなるウィンドウ**」の対応する場所に配置されます。端にある4 つのエリアにより、ターゲットとなるウィンドウの、対応する側への配置を行うことができます。中央のボタンは、ドラッグされたウィンドウをターゲットウィンドウのタブとしてドックします。タブバーにドラッグしてドロップすることにより、他のウィンドウ内のタブとしてウィンドウをドックすることもできます。

- ウィンドウをドラッグすると、4 つの矢印からなる配置コントロールが表れます。各矢印はエディター、または、**デバッガーインターフェイス**のサイドにそれぞれ対応しています。ドラッグされたウィンドウをこれらの矢印の一つの上でリリースすると、ドラッグされたウィンドウがエディター、または、デバッガーインターフェイスのサイドにドックされます。

ウィンドウのタイトルバーをダブルクリックすることでも、ウィンドウのドッキング、非ドッキング状態を切り替えることができます。

(C) 2015-2021 Altova GmbH

9.1.6 ブレークポイント

XSLT and XQuery Debugger ではXSLT、XQuery、そしてXMLドキュメントにてブレークポイントを定義することができます。ブレークポイントは(以下のスクリーンショットにあるように)赤色の点線で表示されます。

メモ: 同一の場所にブレークポイントとトレースポイントの両方をセットすることも可能です。(スクリーンショットにあるように)青と赤による線として表示されます。

```
<!– Local Variable reused –>
<xsl:variable name="OfficeName" select="n1:Name"/>
<!– Display the company name if the varaible is true ********* –>
<xsl:if test="$Show_Company_Name">
    <h3>
        <xsl:value-of select="$OfficeName"/>
    </h3>
    <xsl:message>
        <xsl:text>Company Named Displayed</xsl:text>
    </xsl:message>
</xsl:if>
```

デバッグセッションでデバッガーを開始すると、各ブレークポイントにてデバッギングが停止します。この方法を使うと、デバッグを行う場所を限定し、XSLT、XQuery、そしてXMLドキュメント内の特定の場所に注意を集中させることができます。セットできるブレークポイントの数に制限はありません。

注意:ドキュメント内にセットされたブレークポイントの情報は、そのドキュメントが閉じられるまで保持されたままになります。しかし(例えばXSD ドキュメントにて)スキーマビューへの切り替えが行われると、ブレークポイントは削除されます。その後テキストビューやグリッドビューに切り替え直しても、ブレークポイントは表示されません。

XMLドキュメント内のブレークポイント

XMLドキュメントでは任意のノードにブレークポイントをセットすることができます。ブレークポイントによる処理の停止は、そのノードの開始点で行われます。

XSLTドキュメント内のブレークポイント

XSLTドキュメントでは以下の箇所でブレークポイントをセットすることができます:

- テンプレートや(例えば`xsl:for-each`といった)テンプレート命令の開始箇所
- XPath 条件式(XPath 1.0 またはXPath 2.0)
- 文字列により構成されるXML フラグメント。ブレークポイントによる処理の停止は、そのノードの開始点で行われます。

XQueryドキュメント内のブレークポイント

XQueryドキュメントでは、以下の箇所でブレークポイントをセットすることができます:

- XQuery ステートメントの開始箇所
- XQuery 表現内部
- 文字列により構成されるXML フラグメント。ブレークポイントによる処理の停止は、そのノードの開始点で行われます。

ブレークポイントの挿入と削除

ブレークポイントを挿入するには:

1. ドキュメント内でブレークポイントを挿入したい箇所にカーソルを移動させます(上の記述を参照ください)。XSLT デバッグセッションでは、テキストビューならびにグリッドビューにてブレークポイントをセットすることができ、XQuery デバッグセッションでは、テキストビューでのみブレークポイントを使用することができます。
2. 以下のどれかの操作を行います:

   - 「XSL/XQuery | **ブレークポイントの挿入/削除**」を選択
   - **F9** を押下
   - 右クリックして、「**ブレークポイントの挿入/削除**」を選択

ブレークポイントを削除するには:

1. ドキュメント内のブレークポイントが存在する箇所にカーソルを移動させます。
2. 以下のどれかの操作を行ないます:

   - 「XSL/XQuery | **ブレークポイントの挿入/削除**」を選択
   - 「F9」を押下
   - 右クリックして「**ブレークポイントの挿入/削除**」を選択

その他にも、ブレークポイント/トレースポイントダイアログを使用することでブレークポイントを削除することができます:

1. メニューオプションから「XSL/XQuery | **ブレークポイント/トレースポイント...**」を選択
2. ダイアログボックスからブレークポイントを選択して、「**削除**」ボタンをクリック

「**全て削除**」ボタンにより、ダイアログボックス(そしてXSLT スタイルシートにある)全てのブレークポイントを削除することができます。

ブレークポイントの有効化/無効化

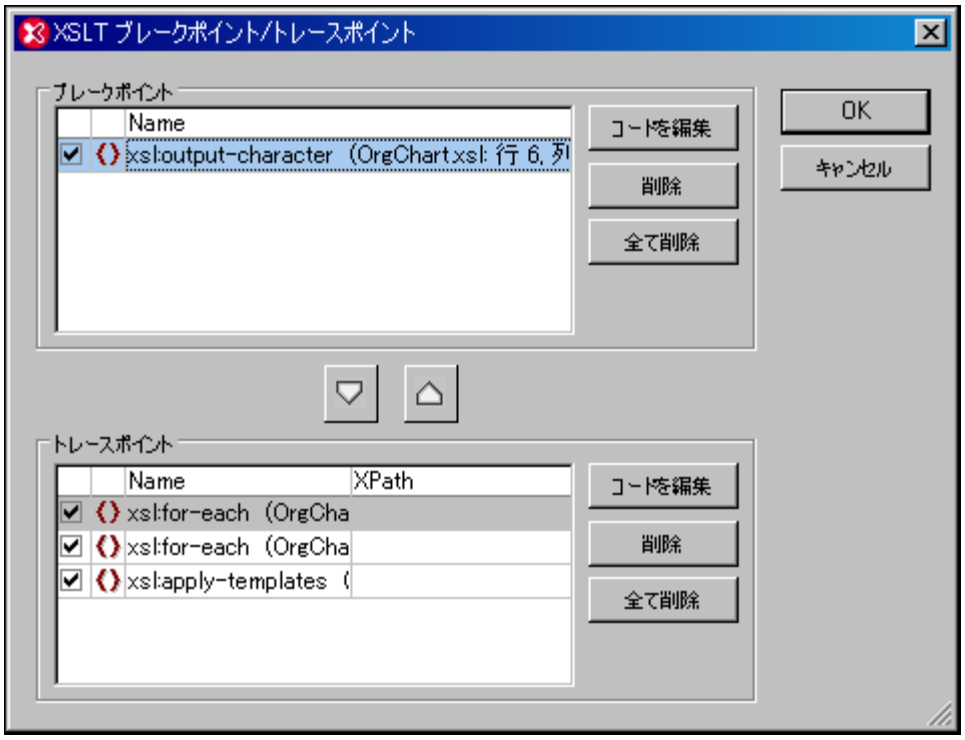
ブレークポイントを挿入した後に、そのブレークポイントを(削除すること無く)スキップしたい場合は、ブレークポイントの無効化を行うことができます。必要に応じて、ブレークポイントは再度有効化することができます。

ブレークポイントを無効化するには:

1. ブレークポイントが含まれているノードまたは表現にカーソルを移動します。
2. 「XSL/XQuery | **ブレークポイントの有効化/無効化**」を選択します(または「Ctrl」+「F9」を押下します)。ブレークポイントが赤色からグレーになり、無効化されます。

その他にも、ブレークポイント/トレースポイントダイアログを使用することでブレークポイントの無効化を行うことができます:

1. メニューオプションから「XSL/XQuery | **ブレークポイント/トレースポイント...**」を選択して、XSLT ブレークポイント/トレースポイントダイアログを表示します。ダイアログには開かれているXML ソースならびにXSLT スタイルシートにて現在定義されている全てのブレークポイントが表示されます。



2. 無効化したいブレークポイントのチェックマークを外し、「OK」ボタンにて確定します。ブレークポイントが赤色からグレーになり無効化されます。

ブレークポイントを有効化するには:

1. ブレークポイントを含んだコードや表現にカーソルを移動します。
2. 「XSL/XQuery | ブレークポイントの有効化/無効化」を選択します(または「Ctrl」+「F9」を押下します)。ブレークポイントがグレーから赤色になり、有効化されます。

特定のブレークポイントを見つける

特定のブレークポイントを見つけるには:

1. メニューオプションから「XSL/XQuery | ブレークポイント/トレースポイント...」を選択します。XSLT ブレークポイント/トレースポイントダイアログが表示されます。
2. ブレークポイントのリストから目的のブレークポイントをクリックします。
3. 「コードを編集」ボタンをクリックします。ブレークポイントダイアログが閉じられ、テキストビューにて選択したブレークポイントの直前にカーソルが配置されます。拡張グリッドビューの場合、ブレークポイントを含んだテーブルセルが赤色でハイライトされます。

ブレークポイントの後でデバッグを継続する

ブレークポイントの後にある処理を継続するには:

- 「XSL/XQuery | ステップイン」または「XSL/XQuery | デバッガーの開始/実行」コマンドを選択してください。

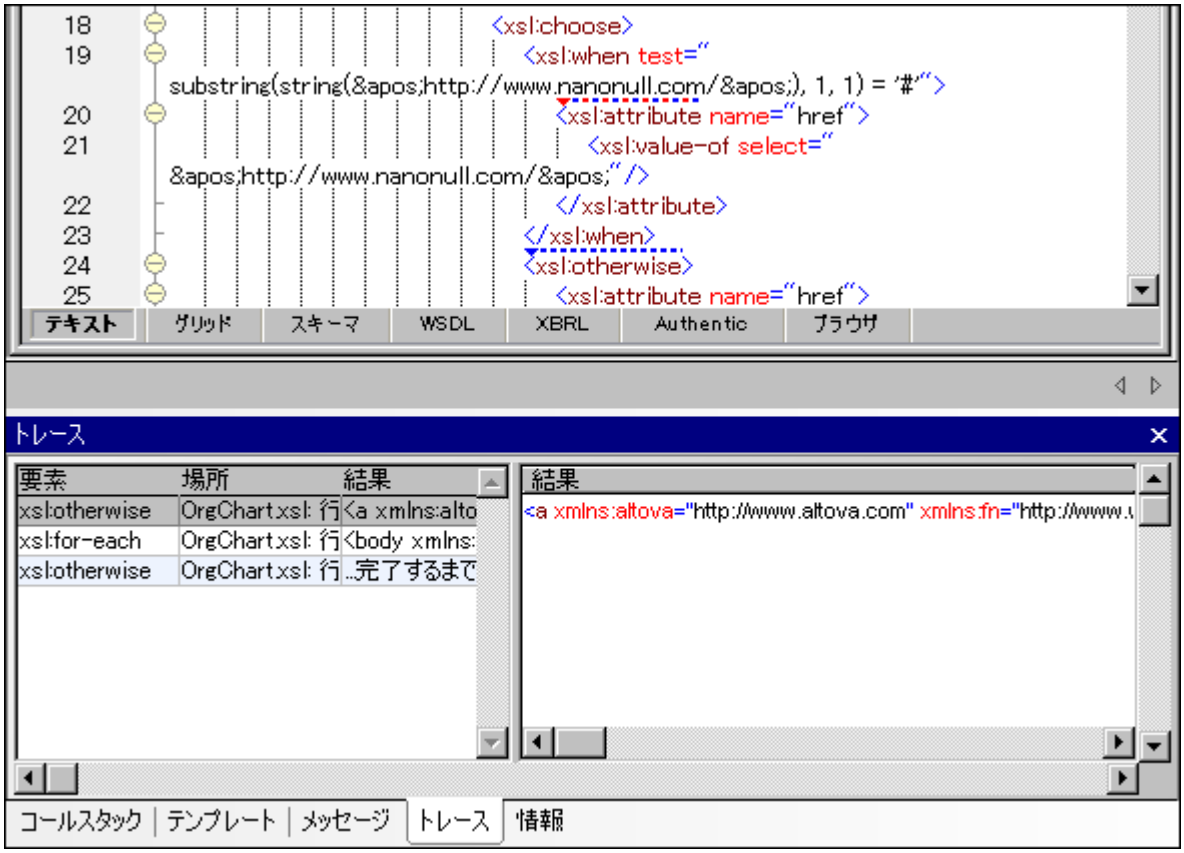
9.1.7 トレースポイント

XSLT and XQuery DebuggerではXSLT ドキュメントのトレースポイントを定義することができます。

トレースポイントを使うことで、命令により生成されたコンテンツをトレースし、(xsl:message 要素を使ってメッセージを表示させるなどの) XSLT スタイルシートの編集を行うこと無く、トレースポイントがセットされた箇所におけるXPath 条件式の結果を確認することができます。

トレースポイントは(下のスクリーンショットにあるように)XSLT スタイルシートにて青色の点線で表示されます。

メモ: 同一の場所にブレークポイントとトレースポイントの両方をセットすることも可能です。(スクリーンショットにあるように) 青と赤による線として表示されます。



トレースポイントがセットされた箇所では、各命令により生成された結果が出力されます。この出力はトレースウィンドウで確認することができます。セットすることのできるトレースポイントの数に制限はありません。

注意: セットされたトレースポイントはドキュメントが閉じられるまで保持されます。

XSLT ドキュメントのトレースポイント

トレースポイントはXSL 命令やXSLT スタイルシートのリテラル結果にセットすることができます。

XML とXQuery ドキュメントのトレースポイント

トレースポイントをXML やXQuery ドキュメントにセットすることはできますが、これらのトレースポイントは効果を持ちません。

トレースポイントの挿入/削除

トレースポイントを挿入するには:

1. XSLT ドキュメント内でトレースポイントをセットしたい箇所にカーソルを移動させます。デバッグセッションでは、テキストビューならびにグリッドビューでトレースポイントをセットすることができます。
2. 以下のどれかの操作を行ないます:

   - 「XSL/XQuery | トレースポイントの挿入/削除」を選択
   - 「Shift」+「F9」を押下
   - 右クリックして「トレースポイントの挿入/削除」を選択

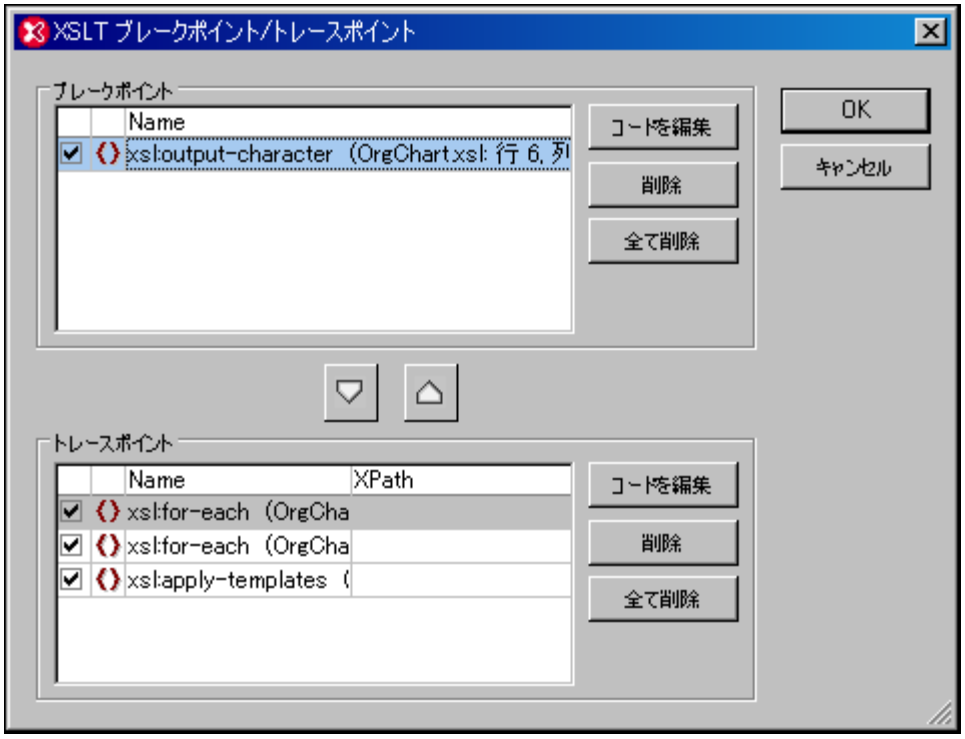
トレースポイントを削除するには:

1. XSLT ドキュメント内のトレースポイントがある場所にカーソルを移動させます。
2. 以下のどれかの操作を行ないます:

   - 「XSL/XQuery | トレースポイントの挿入/削除」を選択
   - 「Shift」+「F9」を押下
   - 右クリックして「トレースポイントの挿入/削除」を選択

他にも、XSLT ブレークポイント/トレースポイントダイアログからトレースポイントの削除を行うことができます:

1. メニューオプションから「XSL/XQuery | ブレークポイント/トレースポイント...」を選択します。
2. (スクリーンショットのように) ダイアログボックスのトレースポイントをクリックし、「削除」をクリックします。



トレースポイントペインにある「全て削除」ボタンにより、ダイアログボックスに表示されている(そして XSLT スタイルシートにある) 全てのトレースポイントを削除します。

トレースポイントにXPath をセット

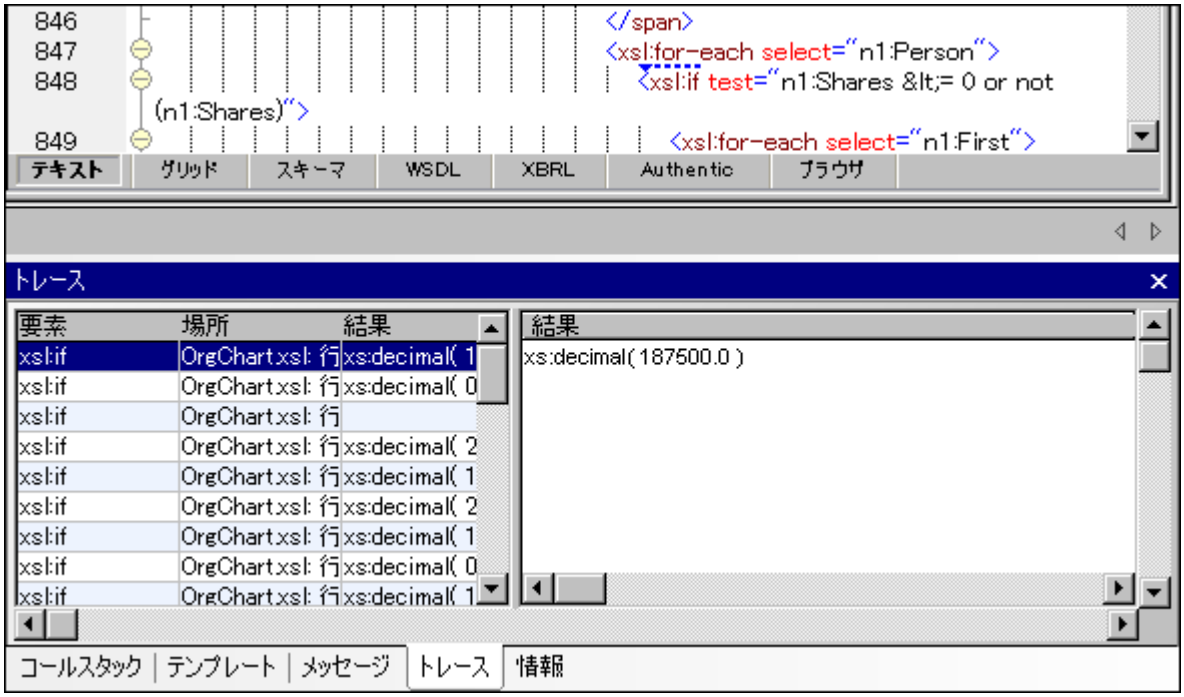
XPath に対してもトレースポイントをセットすることができます。XPath に対してトレースポイントをセットすると、ステートメントにより生成されるコンテンツの代わりに、XPath の評価結果がトレースウィンドウに表示されます。XPath の評価は、トレースポイントがセットされた箇所のコンテキストとは相対的に評価されます。

XPath にトレースポイントをセットするには:

1. メニューオプションから「XSL/XQuery | ブレークポイント/トレースポイント...」を選びます。XSLT ブレークポイント/トレースポイントダイアログには、現在開かれている全てのXSLT スタイルシートドキュメント内で定義されているトレースポイントが表示されます。
2. 目的のトレースポイントの行の「XPath」フィールドにて XPath を入力します。

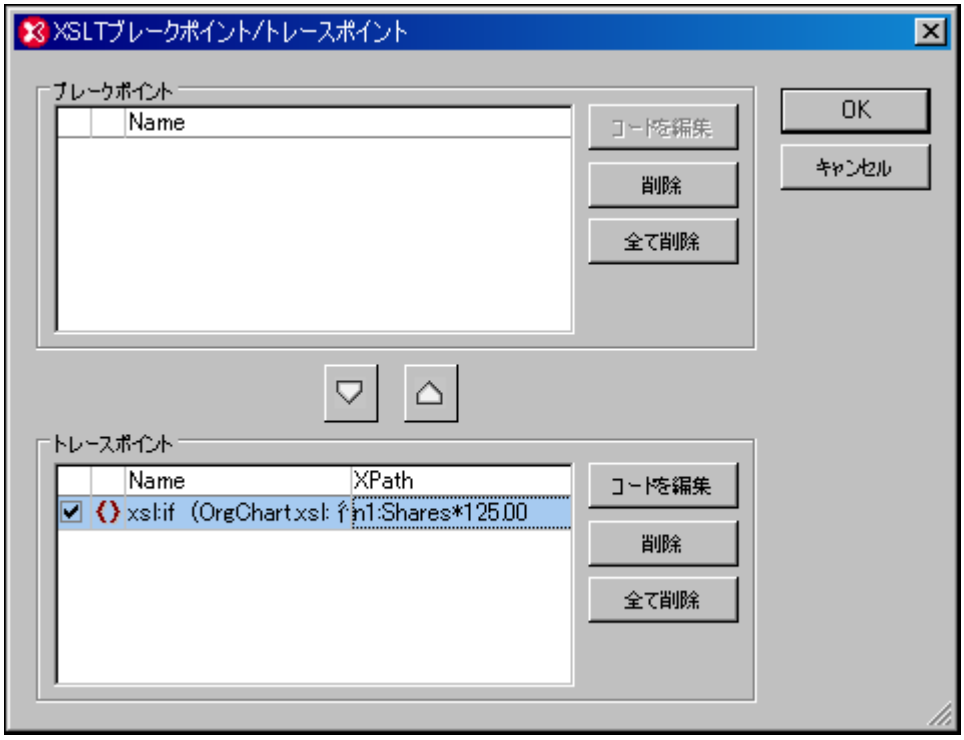
例:

以下のサンプルに示される通り、コンテキストノードが `Person` となるようにトレースポイントがセットされています。`Person` 要素には `Shares` 要素が含まれています。ここでは各 `Person` が持つ `Shares` の数を 125 でかけた値を表示します。



以下の操作を行ってください:

1. 555 行にてトレースポイントをセットします。
2. XSLT ブレークポイント/トレースポイントダイアログを開き、先程セットしたトレースポイントのXPath にて `n1:Shares*125.00` と入力します。

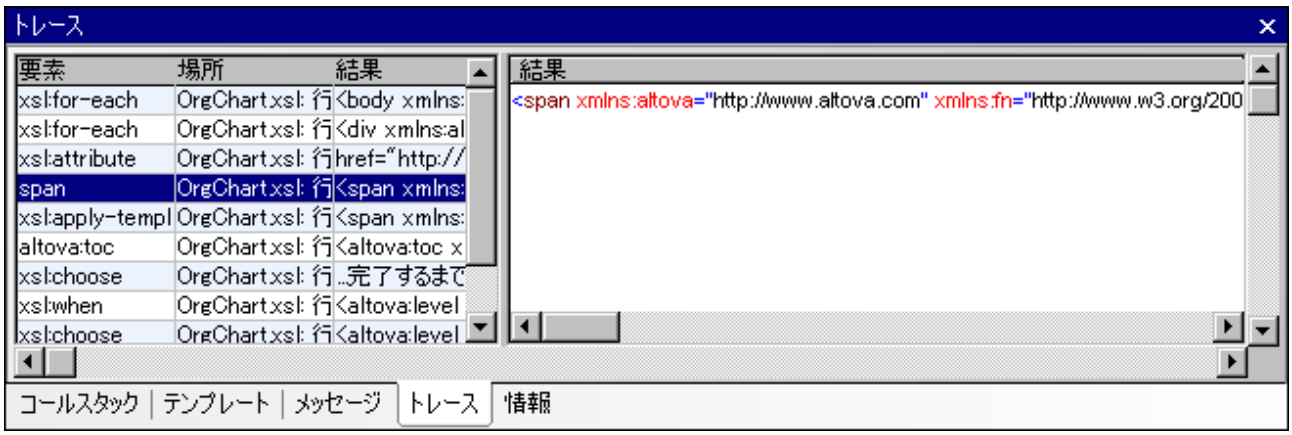


3. デバッガーを開始します。入力したXPath のいかがトレースポイントウィンドウに表示されます。

トレースウィンドウ

「XSL/XQuery | デバッガーの開始/実行」を選択して、デバッグをスタートさせます。トレースポイントがセットされた命令の出力がトレースウィンドウ[481]に表示されます(スクリーンショットを参照)。トレースウィンドウのアイテムをクリックすることで、その結果の詳細がウィンドウの右側に表示されます。

メモ:トレースウィンドウに表示される結果は、トレースされている命令が終了した後に表示されます。



トレースポイントの有効化/無効化

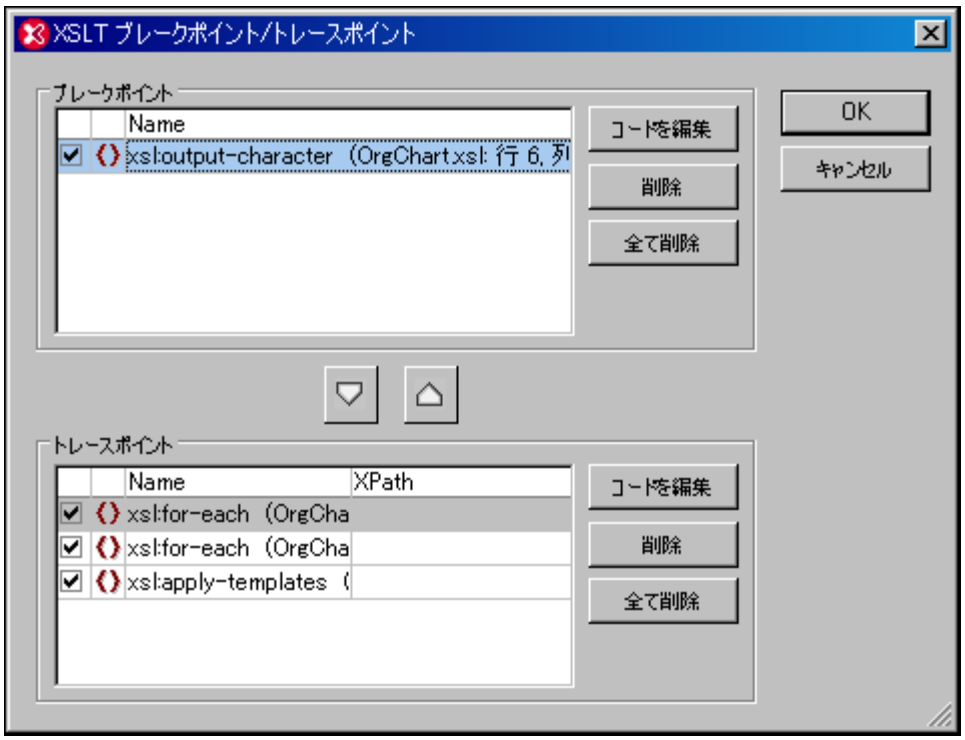
トレースポイントを挿入した後、それらを無効化することで、削除すること無くトレースポイントをスキップすることができます。必要になれば再度有効化することもできます。

トレースポイントを無効化するには:

1. XSLT スタイルシートにある無効化したいトレースポイントにカーソルを移動します。
2. 「XSL/XQuery | トレースポイントの有効化/無効化」を選択するか、「Ctrl」+「Shift」+「F9」を押下します。トレースポイントが青色からグレーに変わり、無効化されます。

その他にも、XSLT ブレークポイント/トレースポイントダイアログからトレースポイントの無効化を行うことができます:

1. メニューオプション「XSL/XQuery | ブレークポイント/トレースポイント...」を選択します。XSLT ブレークポイント/トレースポイントダイアログボックスが開かれ、開かれている XSLT スタイルシートドキュメント内で現在定義されている全てのトレースポイントが表示されます。



2. 各トレースポイントにあるチェックを外すことで、トレースポイントを無効化して、「OK」で確定します。トレースポイントが青色からグレーに変わり、無効化されます。

トレースポイントを有効化するには:

1. XSLT スタイルシートにある有効化したいトレースポイントにカーソルを移動します。
2. 「XSL/XQuery | トレースポイントの有効化/無効化」を選択するか、「Ctrl」+「Shift」+「F9」を押下します。トレースポイントが青色からグレーに変わり、無効化されます。

特定のトレースポイントを見つける

特定のトレースポイントを見つけるには:

1. メニューオプションから「XSL/XQuery | ブレークポイント/トレースポイント...」を選択します。XSLT ブレークポイント/トレースポイントダイアログが表示されます。
2. リストにある目的のトレースポイントをクリックします。
3. 「コードを編集」ボタンをクリックします。XSLT ブレークポイント/トレースポイントダイアログが閉じられ、テキストビューにて選択したトレースポイントの直前にカーソルが配置されます。拡張グリッドビューの場合、トレースポイントを含んだテーブルセルが青色でハイライトされます。

9.1.8 デバッガーショートカット

デフォルトのデバッガーショートカットは以下にリストされているとおりです

- *デバッガーコマンド*

F9	ブレークポイントの挿入/削除
F9 + Shift	トレースポイントの挿入/削除
F9 + CTRL	ブレークポイントの有効化/無効化
F9 + Shift + CTRL	トレースポイントの有効化/無効化
F11	ステップイン
F11 + Shift	ステップアウト
F11 + CTRL	ステップオーバー
F11 + Alt	デバッガーを開始する

9.2 XSLT と XQuery プロファイラー

Altova ウェブサイト : XSLT プロファイラー、XQuery プロファイラー

XSLT/XQuery プロファイラーを利用することで、XMLSpy 内部からXSLT(1.0 ならびに2.0)スタイルシートやXQuery ドキュメントの実行時間を分析することができます。XSLT スタイルシートやXQuery ドキュメントにある各命令の実行に必要な時間が表示され、これらファイルの実行時間を最適化するための材料として使うことができます。

プロファイラーを使用することで、実行時間が最も長い命令を発見し、全体の最適化を行うことができます。
命令の合計実行時間が長くなる原因として、以下の2つが考えられます。

- 長い実行時間を必要とする命令である
- 命令の呼び出し回数が高い(高いヒットカウント)

ヒットカウントとコールグラフプロファイリング

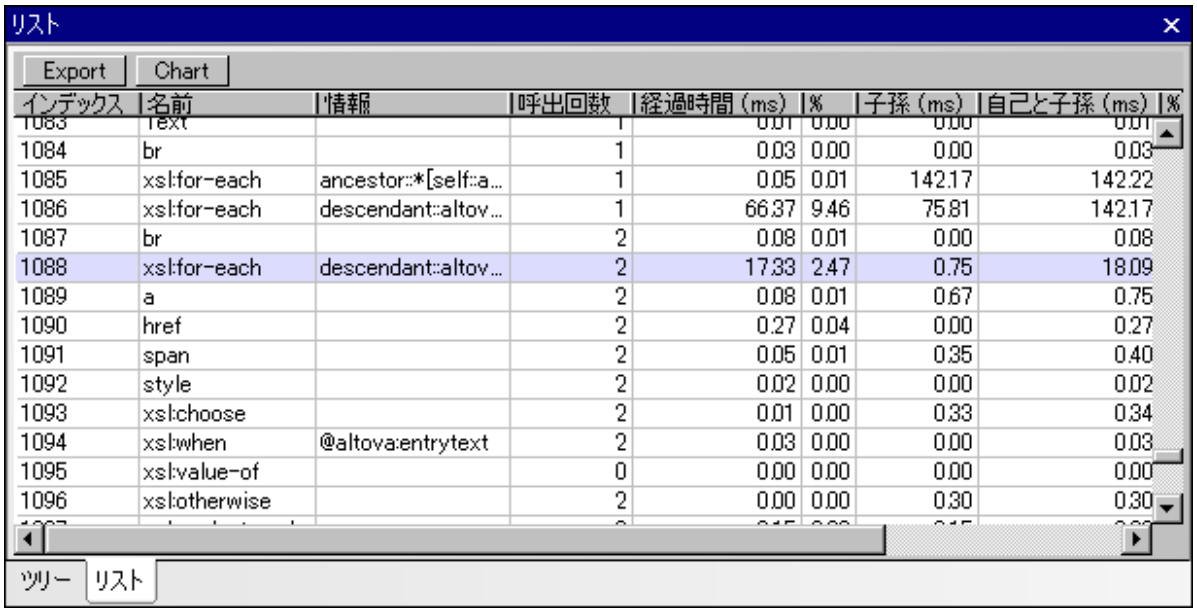
プロファイラーではヒットカウントとコールグラフプロファイリングからの選択を行うことができます。各命令の実行結果の表は両方のプロファイリングで表示されます。
最適化の為には、各命令を1行で表示するヒットカウントプロファイリングが通常使用されます。
コールグラフプロファイリングではXSLT 変換またはXQuery 実行全体の履歴が表示され、例えば変換において呼び出されたテンプレートや関数が表示されます。コールグラフプロファイリングでは、命令毎に1行使用するのではなく、命令の呼び出し毎の表示が行われます。

XSLT/XQuery プロファイラーを使用するには、XSLT プロファイリング[497]とXQuery プロファイリング[500]を参照ください。

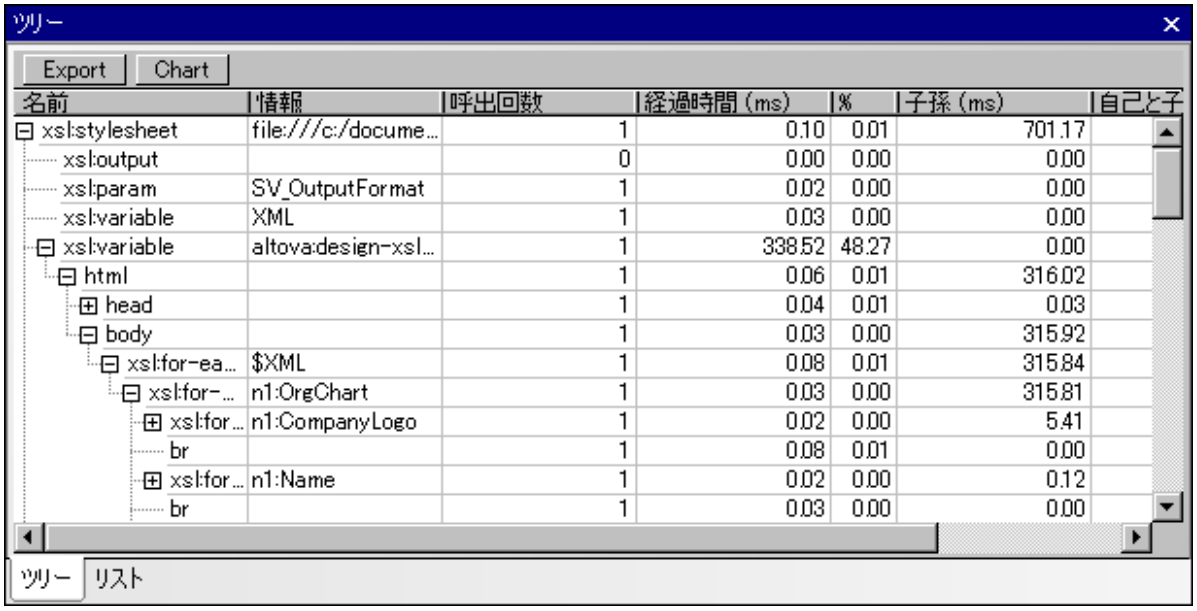
プロファイラービュー

分析の結果は、タブにより切り替えることのできる以下のビューのどれかで確認することができます:

- **リスト表示**:プロファイリングの結果がリスト形式で表示され、例えば命令の実行に要した経過時間により並べ替えを行うことができます。



- **ツリー表示**：結果がツリー構造に似た形式で表示されます。例えば関数の実行に要した時間を確認し、その関数のツリーを展開して、関数内にある各命令が実行に要した時間と実行回数を参照することができます。



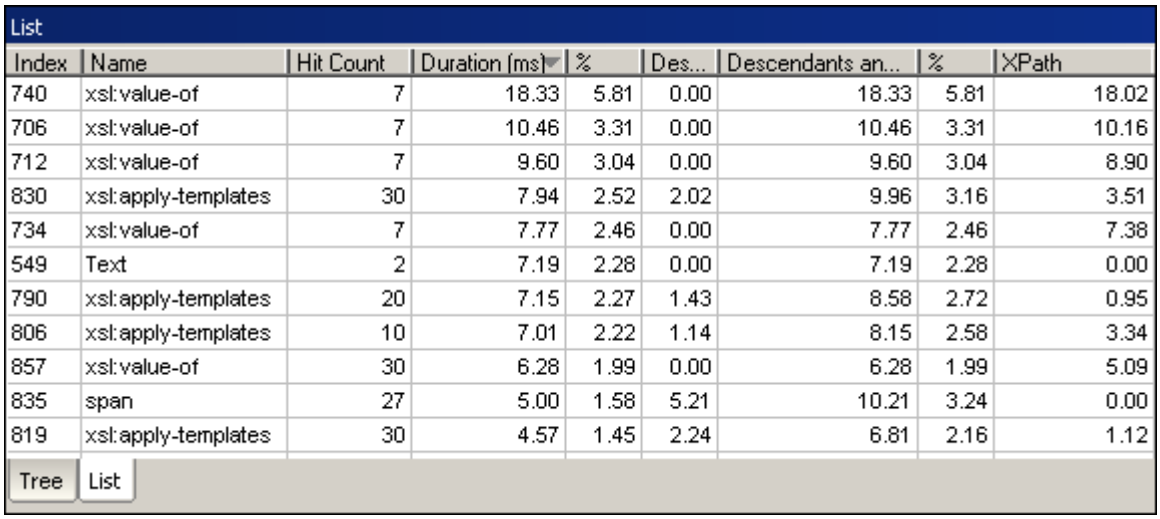
結果の並び替え

プロファイリングを行った後、各命令が実行に要した時間や、各命令が呼び出された回数により並べ替えを行うことができます。

プロファイラー内の情報を並べ替えるには:

1. 「**リスト**」タブをクリックします。
2. 並べ替えを行いたい列のヘッダーをクリックします。例えば**呼び出し回数**では、命令が呼び出された回数で、**経過時間**では実行に要した時間により並べ替えが行われます。

このスクリーンショットでは命令の経過時間を降順で並べ替えた結果を示します。



XSLT スタイルシートとXQuery ドキュメントの最適化

XSLT スタイルシートならびにXQuery ドキュメントにある命令の最適化を行う際には、以下の点に注意してください:

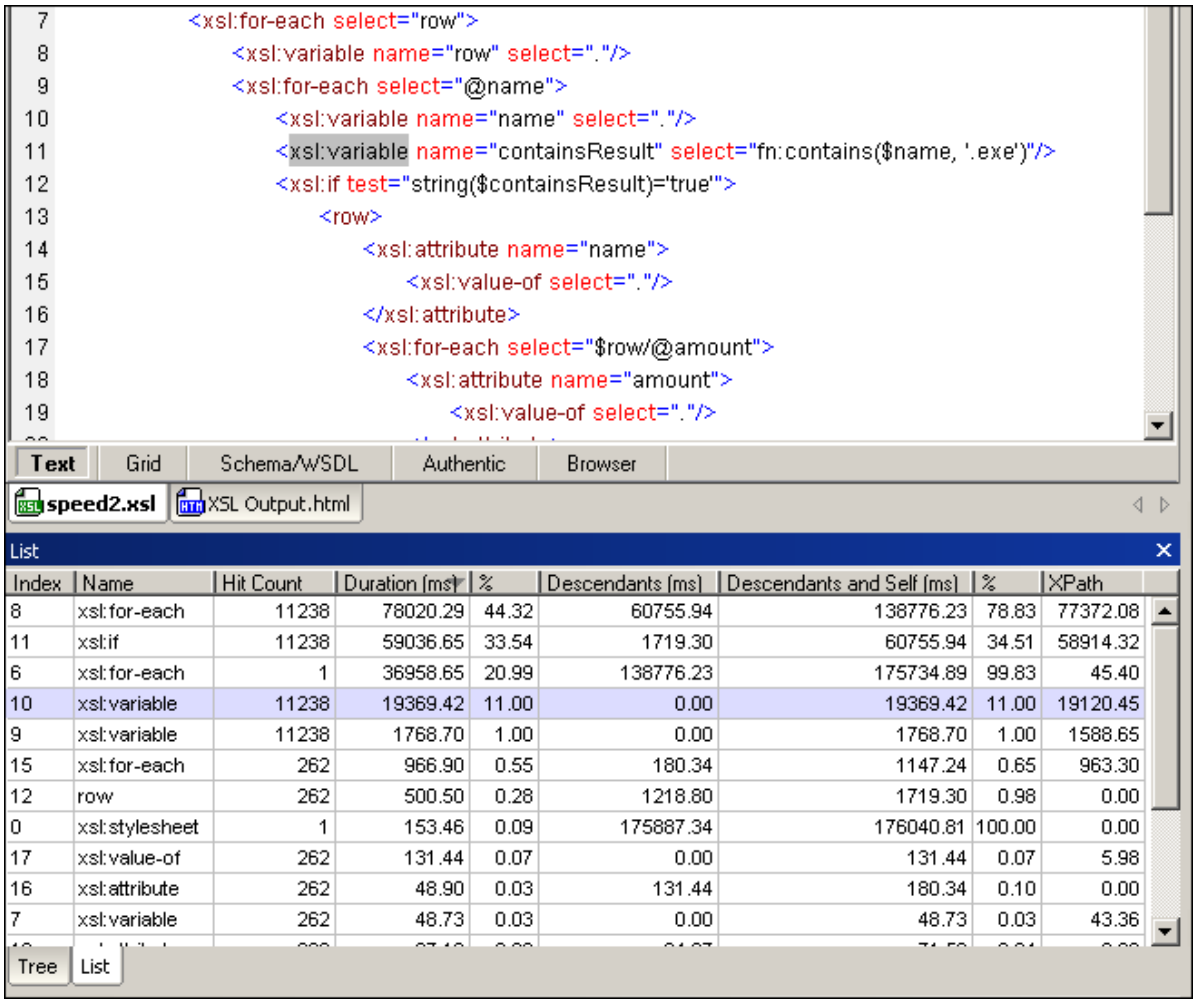
- 変数が1度だけしか使用されないのであれば、命令内での変数の使用を避ける(変数の宣言には時間がかかるため)。

  以下の2つのXSLT コードフラグメントにて、必要ない変数を削除することでコードの最適化を図る例を示します。両者は同じことを行っていますが、2つめの例では変数 `name` と`containsResult` を使用すること無く同じ処理を行っています:

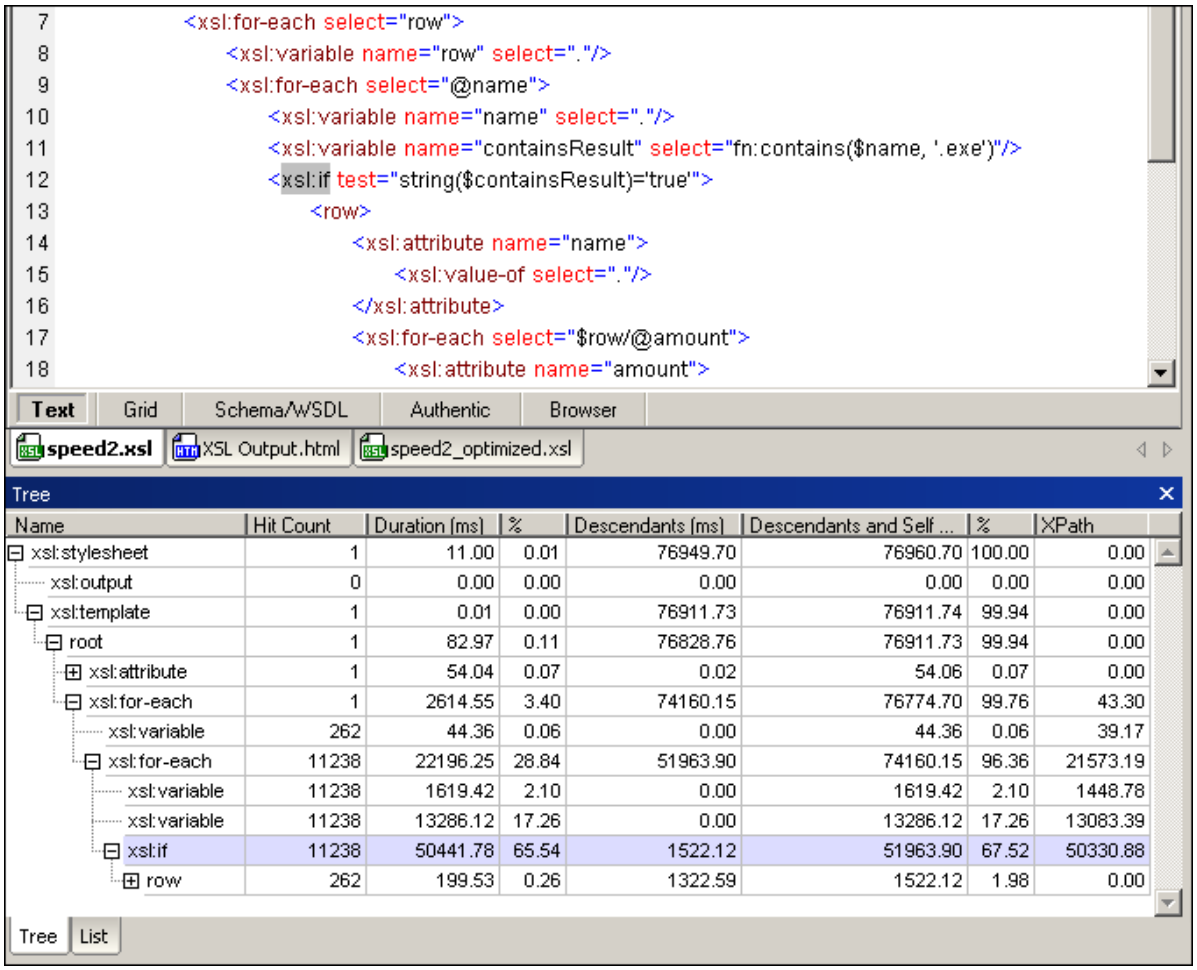
  コードフラグメント1:

```
<xsl:for-each select="row">
	<xsl:variable name="row" select="."/>
	<xsl:for-each select="@name">
		<xsl:variable name="name" select="."/>
		<xsl:variable name="containsResult" select="fn:contains($name,
'.exe')"/>
		<xsl:if test="string($containsResult)='true'">

		...

		</xsl:if>
	</xsl:for-each>
</xsl:for-each>
```

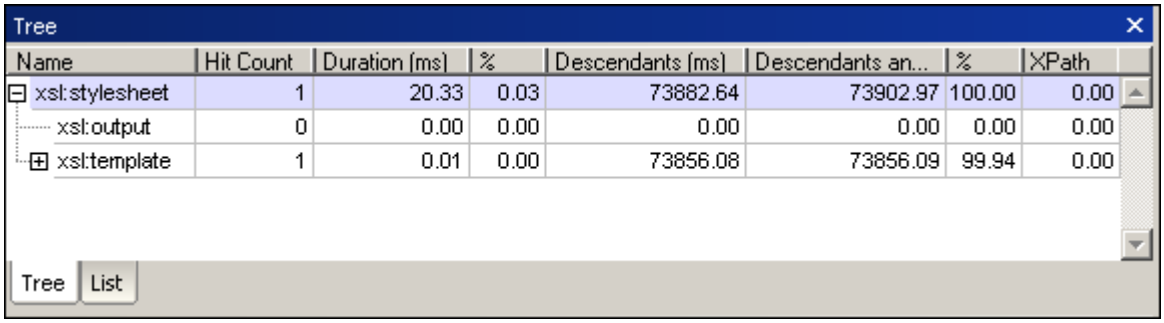
  以下のスクリーンショットでは、このコードフラグメントを含むファイルの分析結果を命令の長さにより並べ替えた結果を示しています。変数 `containsResult` が初期化するための命令に、合計で19 秒の実行時間が必要になっています。



以下のスクリーンショットでは分析結果がツリー表示で示されています。ここでは、変数 containsResult を使用するIF ステートメントの実行に合計で約 50 秒かかることが確認できます:



XSLT 変換には合計で約74秒必要なことが確認できます:

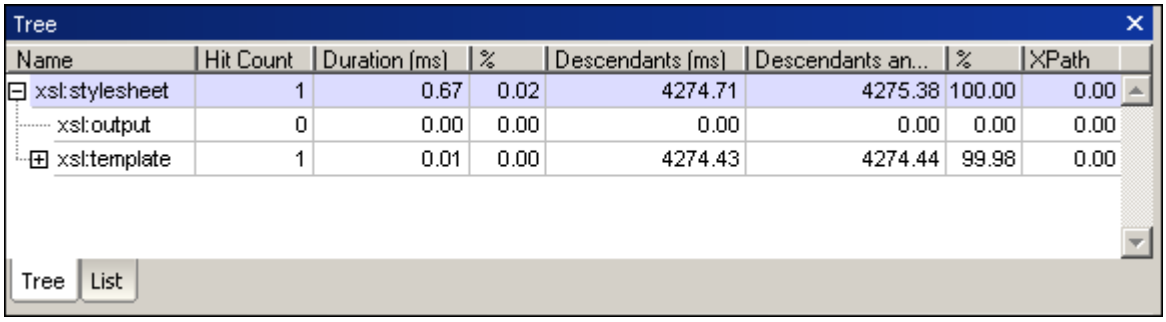


コードフラグメント2:

```
<xsl:for-each select="row">
       <xsl:variable name="row" select="."/>
       <xsl:for-each select="@name">
              <xsl:if test="fn:contains(., '.exe')">

              ...

              </xsl:if>
```

```
        </xsl:for-each>
</xsl:for-each>
```

これらの変数を使用すること無くスタイルシートを書き直すと、合計実行時間は約 4.3 秒まで短縮できることが確認できます。



- 値や条件式が繰り返し使用される場合には変数を使用する。
- 関数内ではローカル定数の作成を避ける。代わりにグローバル変数を使用する。
- 関数内では静的なツリーフラグメントの作成を避ける。代わりにグローバルで作成する。
- かっこを使ったフィルタリングの評価は各ノードに対して行われるため、かっこの使用を制限する。例えば、名前を使ったフィルタリングを行うことで、かっこの使用数を減らすことができます。以下の例では、＊は2つのかっこと共に使用されています:

  ```
  //*[node-name()=Book][author="Steve"]
  ```

  等価の処理を行う以下のステートメントでは、Book という名前と1つのかっこが使用されています:

  ```
  //Book[@Author="Steve"]
  ```

- 1度しか実行されなくても良い命令は1度しか呼び出されないように命令を分割する。グローバルな文脈でしか使用されない部分からは、グローバル変数を作成します。

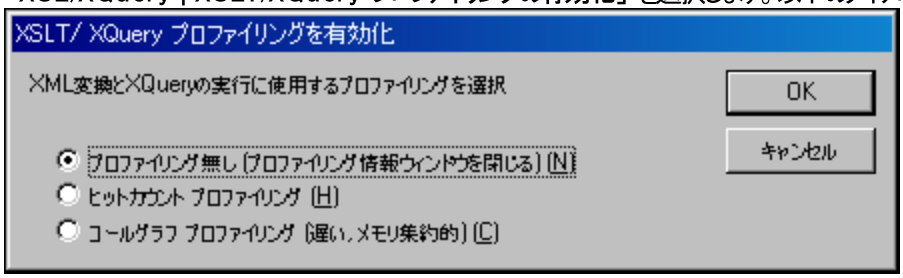
9.2.1 XSLT プロファイリング

プロファイラーの開始

注意：プロファイラーに表示される実行時間は、お使いのコンピューターで同時に動作している他のアプリケーションにより影響を受ける可能性もあります。プロファイラーを使ってファイルの分析を行うには、XMLSpy アプリケーションのみを動作させることが理想的です。

XSLT スタイルシートの分析を行うには:

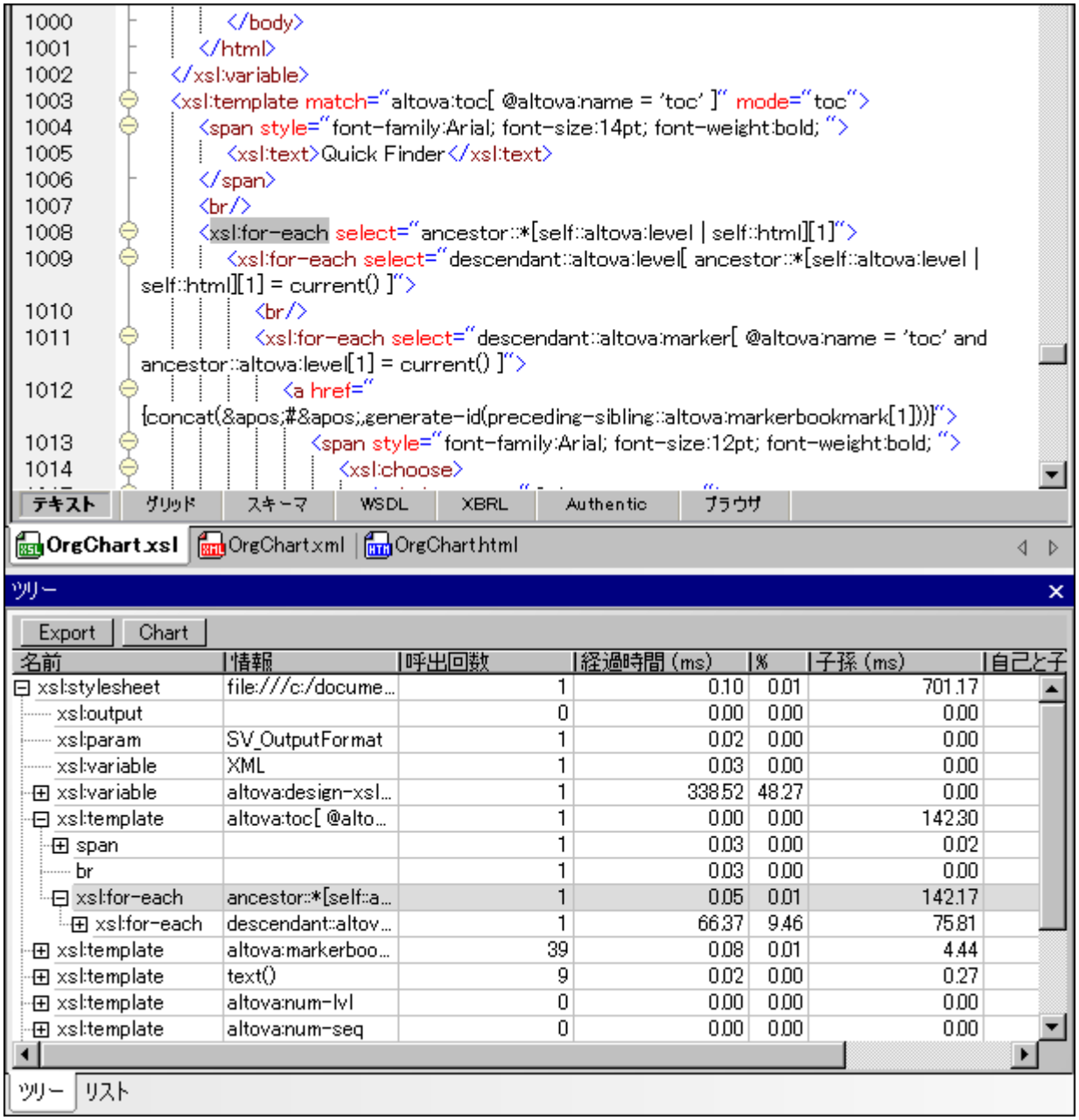
1. XMLSpy にて XSLT 変換の入力データとして使用される XML ファイルを開きます。
2. 「XSL/XQuery | XSLT/XQuery プロファイリングの有効化」を選択します。以下のダイアログが開きます。



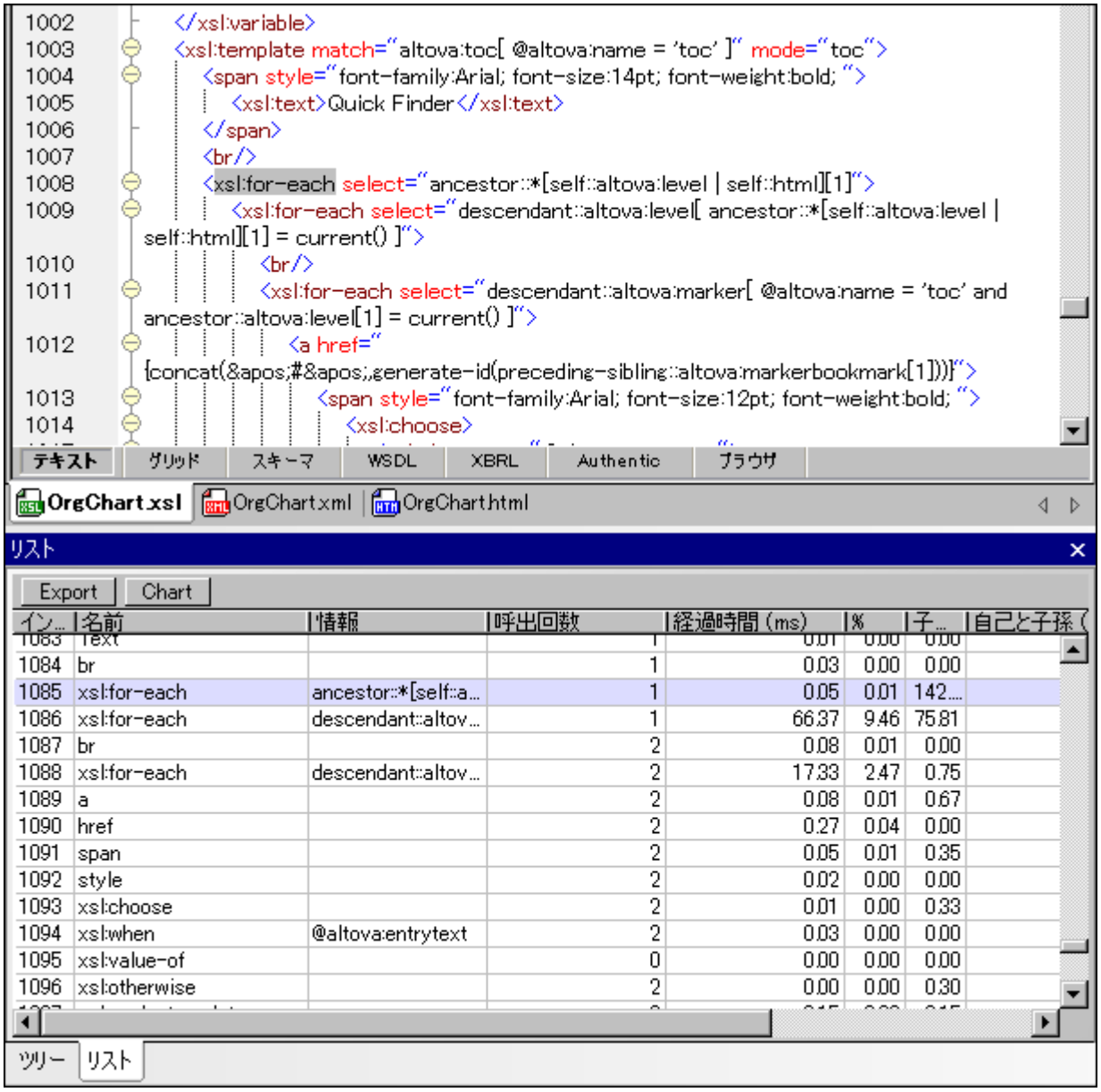
3. 「ヒット カウント プロファイリング」または「コールグラフプロファイリング」を選択します。「OK」により確定すると、空のプロファイラーウィンドウが開きます。
4. XSL 変換を実行します(「XSL/XQuery | XSL 変換」)。分析を行ないたい XSLT スタイルシートへのパスを選択するダイアログが表れます。変換の終了後、プロファイラーウィンドウに実行の結果が表示されます。
5. プロファイラーにある "+" アイコンをクリックして、XSLT スタイルシートの実行時間を表示します(以下のスクリーンショットを参照)。これらのスクリーンショットにはヒットカウントプロファイリングが使用されています。

プロファイラーにある行をクリックすると分析が行われたファイルの対応する命令がハイライトされます。

以下のスクリーンショットではプロファイラーのツリービューが示されます:



以下のスクリーンショットでは同じ XSLT スタイルシートに対するプロファイラーのリストビューが示されます:



プロファイラーの情報を使用する

XSLT スタイルシートにある各命令に関する以下の情報がプロファイラーには表示されます:

- **インデックス**: 各命令に割り当てられている番号。呼び出された順に番号は振り付けられます。
- **名前**: XSLT 命令の名前。
- **呼び出し回数（ヒットカウント）**: 変換においてその命令が呼び出された回数の合計。
- **経過時間（ms）と%** : （子孫を計算に入れず）その命令を実行するのに要した時間をミリ秒単位で表し、全体に占める実行時間の割合が一緒に表示されます。

- **子孫 (ms)**: その命令の子孫を実行するのに要した時間をミリ秒単位で表します。
- **自己と子孫 (ms) と %** : その命令と、その命令の子孫を実行するのに要した時間がミリ秒単位で表示され、全体に占める実行時間の割合が一緒に表示されます。
- **XPath**: 命令に XPath ステートメントが含まれている場合、そのステートメントを実行するのに要した時間が表示されます。

メモ: ヒットカウントプロファイリングを使用している場合、プロファイラーウィンドウに表示される時間は、その命令に対する全ての呼び出しに要した時間の合計になります。コールグラフプロファイリングを使用している場合、命令の各呼び出しが別々に表示されるため、プロファイラーウィンドウに表示される時間は各実行の個別に呼び出すための時間となります。

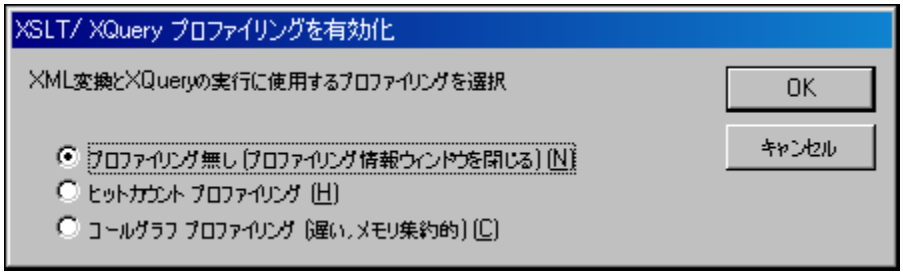
9.2.2 XQuery プロファイリング

プロファイラーの開始

注意: プロファイラーに表示される実行時間は、お使いのコンピューターで同時に動作している他のアプリケーションにより影響を受ける可能性もあります。プロファイラーを使ってファイルの分析を行うには、XMLSpy アプリケーションのみを動作させることが理想的です。

XQuery ドキュメントの分析を行うには:

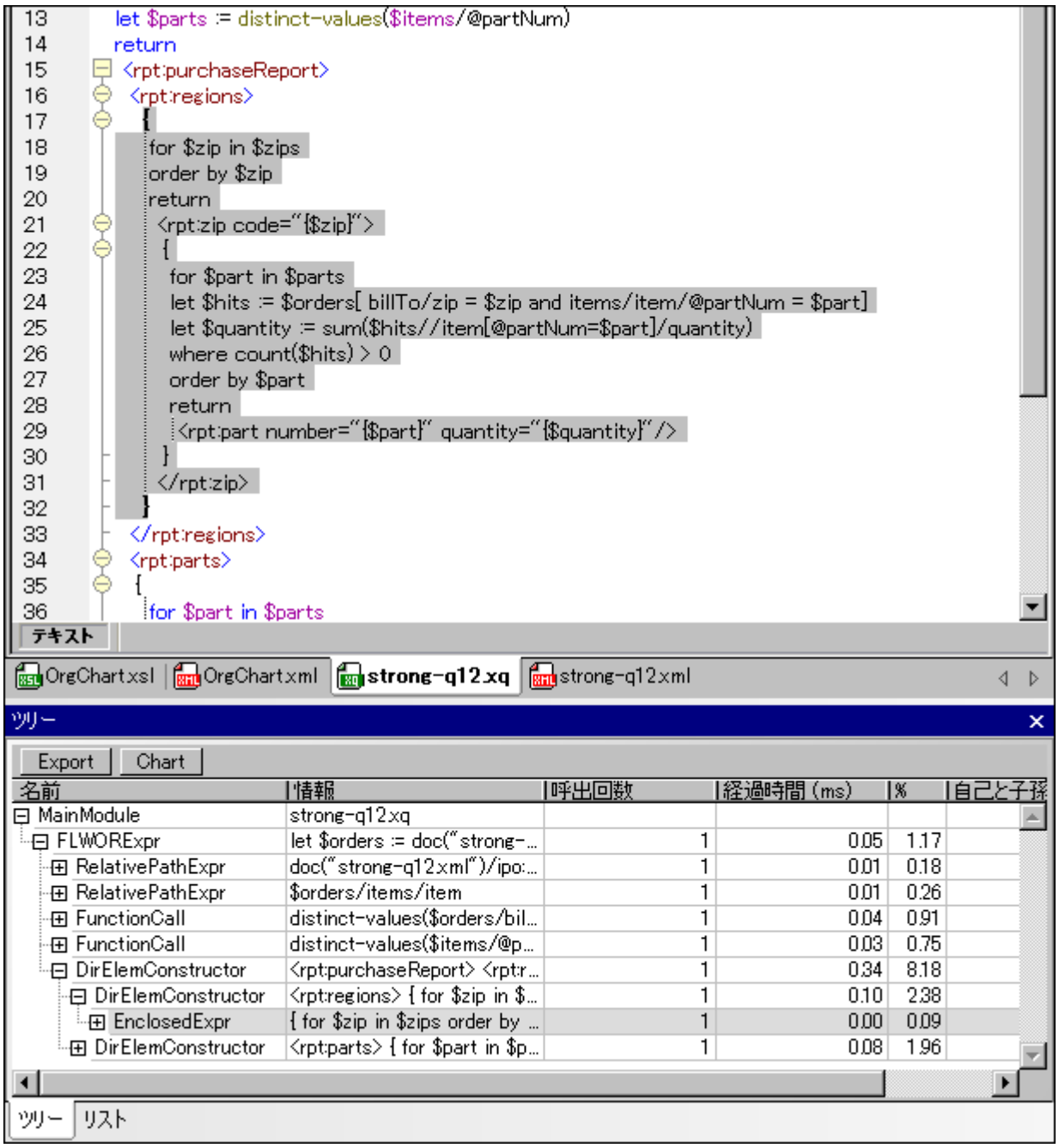
1. XMLSpy にて分析を行う XQuery ドキュメントを開きます。
2. 「XSL/XQuery | XSLT/XQuery プロファイリングの有効化」を選択します。以下のダイアログが開きます。



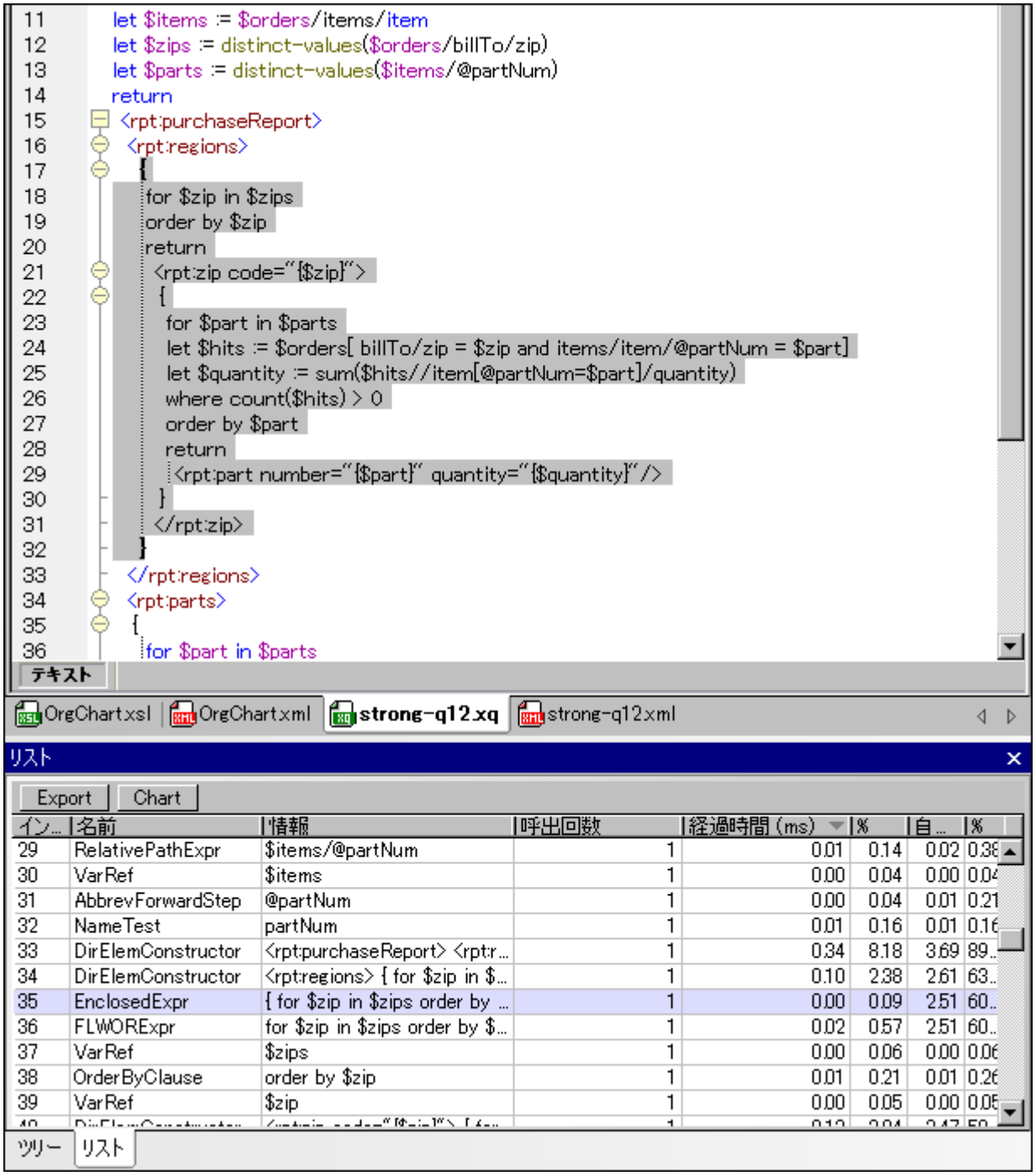
3. 「ヒットカウントプロファイリング」または「コールグラフプロファイリング」を選択します。「OK」により確定すると、空のプロファイラーウィンドウが開きます。
4. XQuery を実行します(「XSL/XQuery | XQuery を実行」)。実行が終了した後、プロファイラーウィンドウに実行の結果が表示されます。
5. プロファイラーにある "+" アイコンをクリックして、XQuery ドキュメントにある命令の実行時間を表示します(以下のスクリーンショットを参照)。これらのスクリーンショットには「ヒットカウントプロファイリング」が使用されています。

プロファイラーの行をクリックすると、分析されたファイルの対応する箇所がハイライトされます。

以下のスクリーンショットではプロファイラーのツリービューが示されます:



以下のスクリーンショットでは同じ XQuery ドキュメントに対するプロファイラーのリストビューが示されます:



プロファイラーの情報を使用する

XQuery ドキュメント内の各命令に関する以下の情報がプロファイラーには表示されます:

- **インデックス**: 各命令に割り当てられている番号。呼び出された順に番号は振り付けられます。

- **名前**:XQuery 命令の名前。
- **情報**:命令に関する情報が表示されます。例えば、命令が変数の宣言であれば、この列にはその関数の名前と値が表示されます。命令が関数であれば、列にはその関数の名前とパラメーターが表示されます。
- **呼び出し回数(ヒットカウント)**:実行中にその命令が呼び出された合計回数を表示します。
- **経過時間(ms)と%**:(子孫を計算に入れず)その命令を実行するのに要した時間をミリ秒単位で表し、全体に占める実行時間の割合が一緒に表示されます。
- **自己と子孫(ms)と%**:その命令と、その命令の子孫を実行するのに要した時間がミリ秒単位で表示され、全体に占める実行時間の割合が一緒に表示されます。

メモ:ヒットカウントプロファイリングを使用している場合、プロファイラーウィンドウに表示される時間は、その命令の全ての呼び出しに要した時間の合計になります。コールグラフプロファイリングを使用している場合、命令の各呼び出しが別々に表示されるため、プロファイラーウィンドウに表示される時間は各実行の個別に呼び出すための時間となります。

9.2.3 プロファイリングの結果: エクスポートとチャート

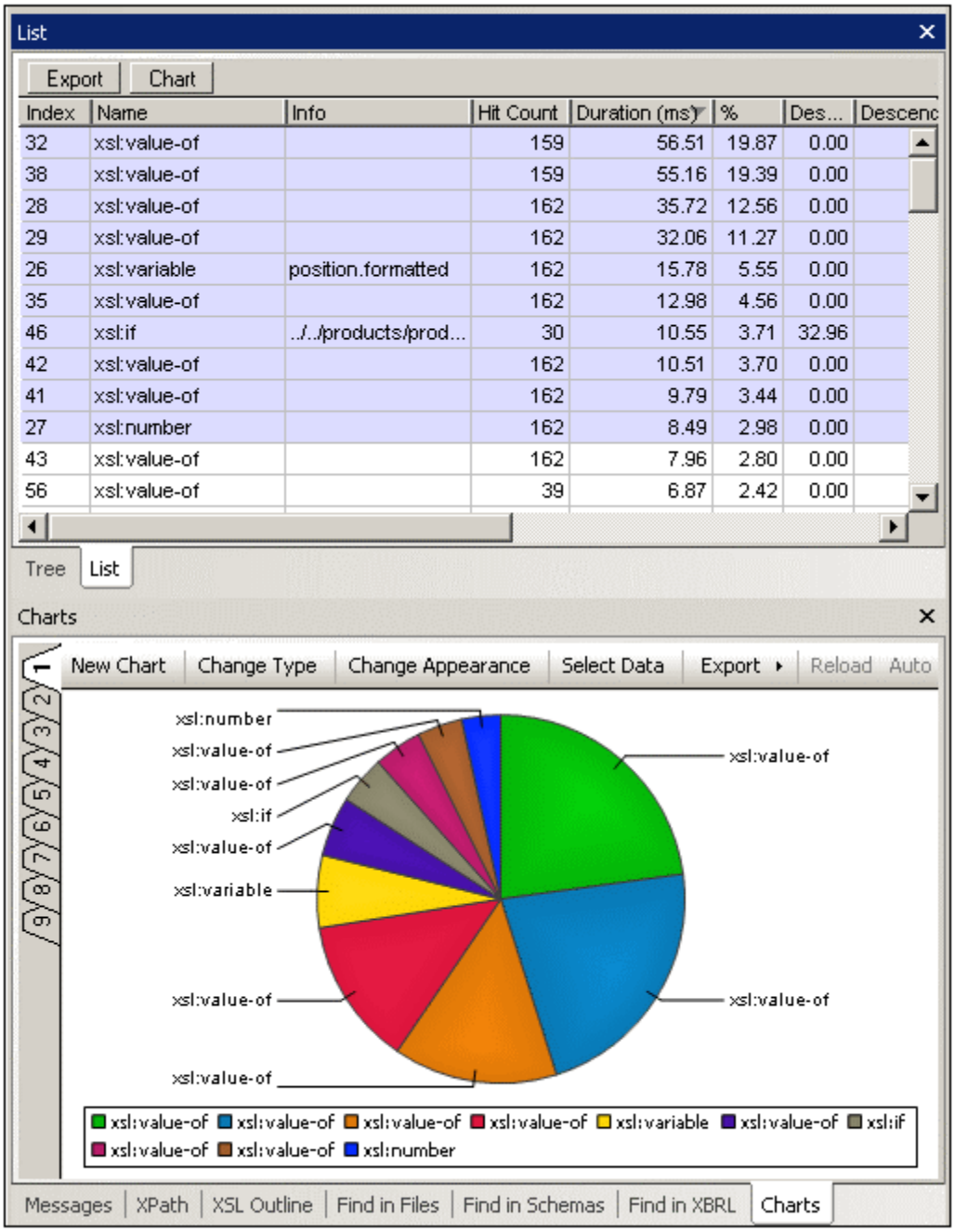
XSLT/XQuery プロファイラーを実行すると、その結果を XML へエクスポートし、作成されたチャートをイメージファイルとして保存することができます。

エクスポート

エクスポートボタンをクリックすると、プロファイリングの結果をエクスポートする XML ファイルの場所とファイル名を指定するよう促されます。グリッドビューを使用することで、XML ファイルの構造をよりクリアに理解することができます。例えばXSLT プロファイリングの結果をグリッドビューで標示すると、XML ドキュメントの構造が`node` 要素により分類される3つの階層レベルから構成されていることを確認することができます。最初の`node` 要素によりドキュメントルートが、2番目の`node` 要素により`xsl:stylesheet` 要素が、3番目の`node` 要素により(`xsl:output` や`xsl:template` といった)グローバル要素が表されます。プロファイリングのデータは、`node` 要素の各属性に保管されます。

チャート

XSLT/XQuery プロファイラを実行後、実行結果やそのサブセットをグラフの形で生成することができます。(以下のスクリーンショットにあるように)プロファイラウィンドウにて、「**チャート**」ボタンをクリックし、グラフをチャート出力ウィンドウに生成します。

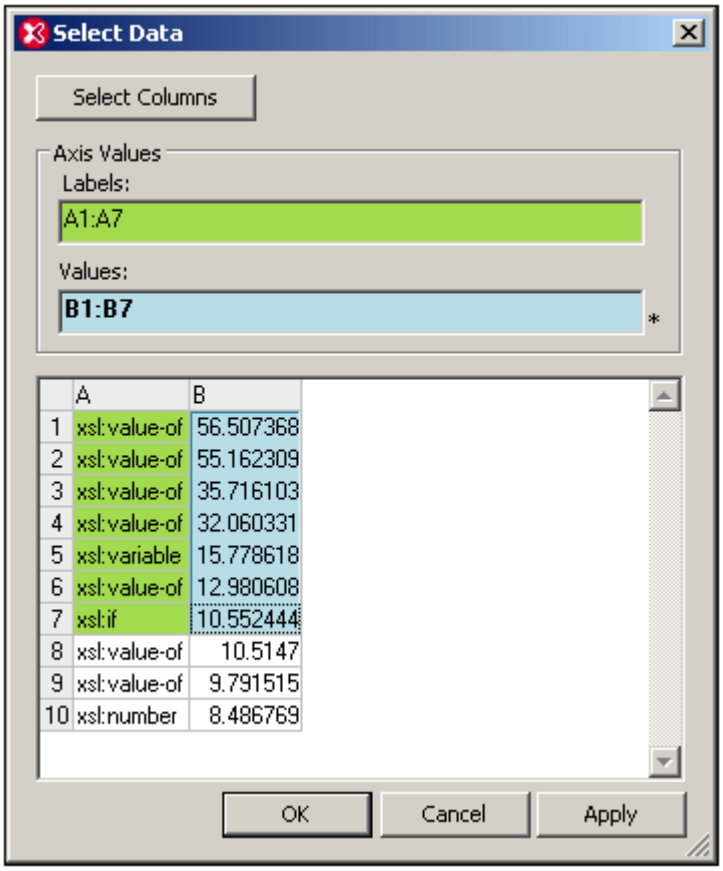


以下の点に注意してください:

- プロファイラーウィンドウ(ツリーならびにリストビュー)にて、項目のマーキングをすることで、実行結果のサブセットを選択することができます。「Ctrl」や「Shift」キーを使って複数のアイテムを選択することができます。リストビューでは、列のヘッダーをクリックすることで、列にある値をベースに並べ替えを行うことができます。例えばこの機能を使うと、最も処理に時間を要するアイテム順に並べ替えを行った後、変換処理に最も時間を要しているアイテムだけを選択し、必要でないアイテムを結果の表示から取り除くことができます。上のスクリーンショットでは、ハイライトされたアイテムの結果だけが表示されています。

(C) 2015-2021 Altova GmbH

- チャートを作成した後は、チャート出力ウィンドウにある種類の変更[332]ボタンをクリックして、そのチャートの種類(円グラフ、棒グラフ、折れ線グラフなど)を変更することができます。利用可能なチャートの種類に関する情報は、このドキュメントのチャート[310]セクションを参照ください。
- チャート出力ウィンドウの「データの選択」ボタンをクリックすることで、データの選択ダイアログが表示されます(以下のスクリーンショットを参照)。このダイアログでは、列の選択プロセスで得られたデータテーブルから、X-軸ならびにY-軸にて使用されるデータの選択を行うことができます。軸のデータ選択を行うには、その軸のテキストフィールドをクリックし、テーブルの範囲(例: `A1:A7`)を入力するか、カーソルをドラッグすることで、範囲を指定します。



「列の選択」ボタンをクリックすることで、データテーブルに表示されるデータの選択を変更することができます。列の選択に関しては、ソースXPath[317]、X-軸の選択[320]、Y-軸の選択[325]を参照ください。

チャートに関するより詳細な情報については、ドキュメントのチャート[310]セクションを参照ください。

10 XPath/XQuery 条件式

XPath とXQuery 式はXML ツリーをナビゲートするために使用されます。バージョン3.1 内のマップと配列のためのサポートに加え、バージョン3.1 のXPath とXQuery 式もJSON 構造をナビゲートするために使用することができます。XPath はXQuery のサブセットで、両言語で有効な式は両方の言語で同じ結果を返します。2つの言語に関する詳細に関してはXPath 3.1 勧告 とXQuery 3.1 勧告 を参照してください。

XMLSpy 内でXPath/XQuery 式を評価する方法

XMLSpy はXPath/XQuery 式を作成し、式をXML とJSON ドキュメントに対して評価しデバッグするためのパワフルな機能を提供します。これにより、使用するXML またはJSON ドキュメントに対して式を素早くビルドしテストすることができます。

出力ウィンドウが他の出力ウィンドウ[108] と共にアプリケーションのインターフェイスの下に表示されている「XPath/XQuery ウィンドウ」内でこれらの分析機能を使用することができます。XPath/XQuery ウィンドウのこれらの機能はこのセクションのサブセクションで説明されています。

一般的なユーザーシナリオでは、以下を行います:

1. 式をビルドまたは評価するXML またはJSON ドキュメントを開きます。
2. XPath/XQuery ウィンドウ内にXPath/XQuery 式を入力します。
3. 結果を確認するためにエバリュエーターまたはデバッガーを実行します。エバリュエーターは最終的な結果を表示しますが、デバッガーは各ステップの結果アスペクトを表示し評価プロセスをステップバイステップに確認します。

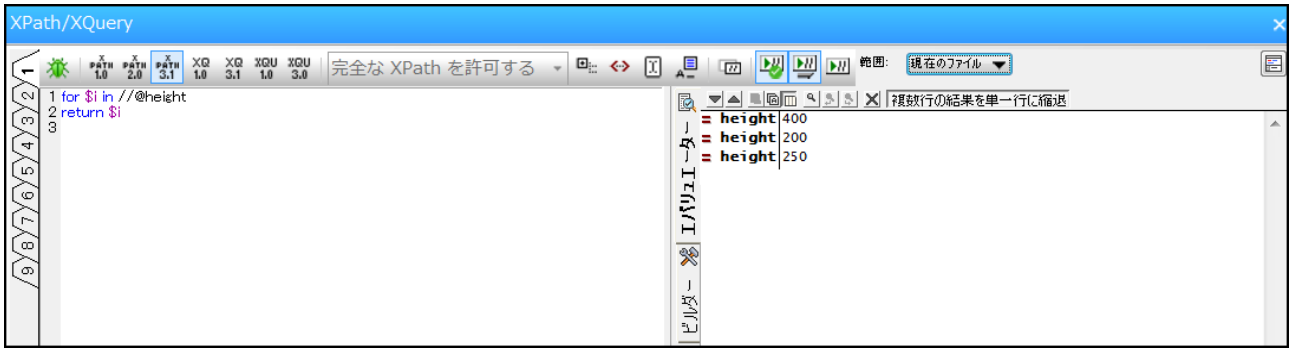
このセクション

このセクションでは以下のように整理されたXPath/XQuery ウィンドウの機能について説明されています:

- Path/XQuery ウィンドウについて[507]
- 評価モード[509]: XPath/XQuery 式の評価のためのモード
- デバッグモード[514]: XPath/XQuery 式のデバッグのためのモード
- 式ビルダー[520]: XPath/XQuery 式の作成のためのビルダー
- JSON のためのXQuery 式[522]
- 注意点[524]

10.1 XPath/XQuery ウィンドウについて

XPath/XQuery ウィンドウ[113] (*下のスクリーンショット*) によりXPath とXQuery 式をXML またはJSON* ドキュメントに対して作成、評価、およびデバッグできるようになりました。



* JSON クエリを有効化する機能はXPath/XQuery 3.1 で導入されました。*XSLT/XQuery を使用した JSON の変換*[666] *を参照してください*。

水平的および垂直的なレイアウト

ツールバーの右角に垂直および水平レイアウトを切り替えるボタンがあります (*上と下のスクリーンショットを参照してください*)。どのモードでもレイアウトを切り替えることができます (*評価モードとデバッグモード*[507] *を参照してください*)。上のスクリーンショットは XPath/XQuery 式が複数のラインで垂直的なレイアウトを表示しています (上のスクリーンショット内の左側のペイン)。

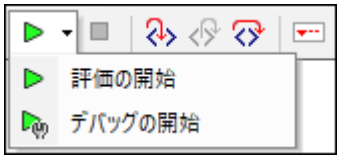
水平的なレイアウト (*下のスクリーンショット*) は結果が水平的な方向に広がるラインを持つ場合役に立ちます。

評価モードとデバッグモード

XPath/XQuery ウィンドウ[113] は2つのモードで使用することができます:

- 1つまたは複数のXML/JSON ドキュメントに対してXPath またはXQuery 式が評価される評価モード[509]。式ペインに式は入力され、結果は横の「結果ペイン」に表示されます。結果内のノードをクリックしてXML またはJSON ドキュメント内のそのノードに移動することができます。
- XPath/XQuery 式を現在アクティブなXML ドキュメントに適用しつつデバッグするデバッグモード[514]。ブレークポイントとトレースポイントを設定することができ、評価をステップバイステップで行うことができます。各ステップで変数のコンテンツを確認することができ、評価の追加アスペクトをチェックするカスタムウォッチ式を設定することができます。

2つのモードを切り替えるには、ウィンドウのツールバーの左角の「**評価/デバッグの開始**」ドロップダウンメニュー内で適切なコマンドを選択します (*下のスクリーンショットを参照*)。



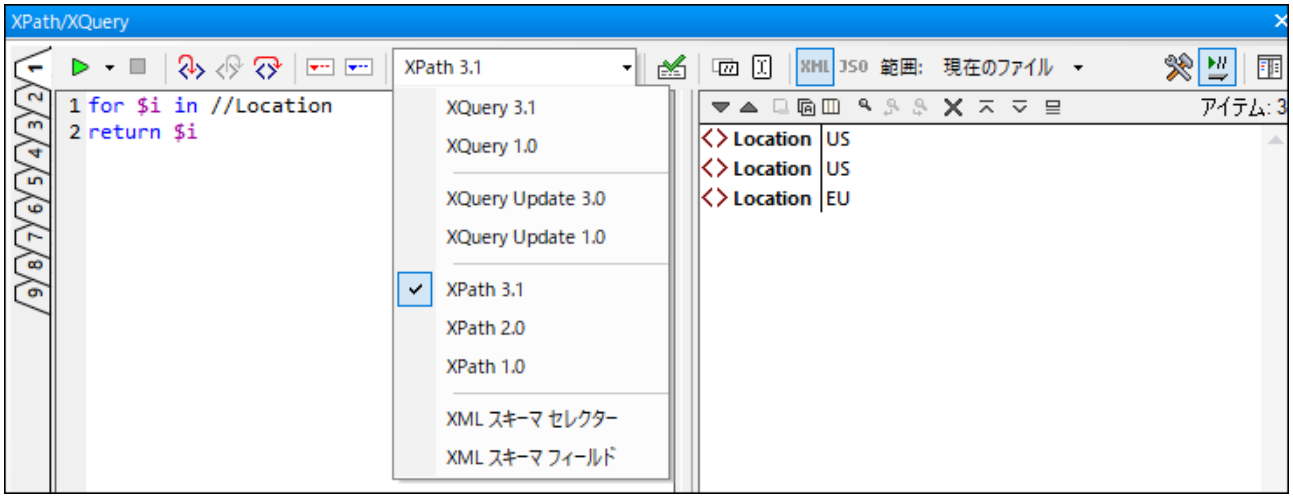
2つのモードの使用方法はこのセクションのサブセクションで説明されています。

XPath/XQuery 式ビルダー

両方のモード[507] 内で、式ビルダー[520] を使用してシステム的に正確な式を作成する手助けを行います。Switch 式ビルダー[520] をメインツールバー の「**ビルダーモード**」ボタンを使用してオンとオフに切り替えます。

10.2 式の評価

XPath/XQuery ウィンドウによりXPath またはXQuery 式を選択する言語で 作成することができ、式を選択範囲オプションで指定する範囲で評価することができます (*下のスクリーンショットを参照*)。式は現在のファイル上、および複数のファイルの組み合わせで評価することができます: (i) 現在開かれているドキュメント (ii) 現在アクティブな XMLSpy プロジェクト 950 のファイル (iii) 選択されたフォルダーのファイル。式を単純にテストするには、スコープとして1 つの適切なファイルのみが必要とされます。XPath/XQuery ウィンドウは結果ペイン内の特定の1 つまたは複数のファイルとレポートを検索するために使用することができます。この場合スコープ内で適切なファイルセットを選択してください。

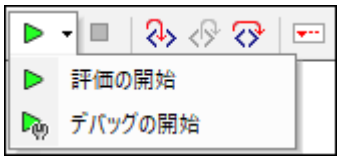


XPath/XQuery ウィンドウはツールバーと式ペイン (*上のスクリーンショットの左側*) と結果ペイン (*上のスクリーンショットの右側*) 2つのペインにより構成されています。

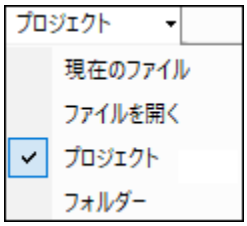
プロシージャの評価

XPath/XQuery 式を評価するには、以下を行います (*上のスクリーンショットを参照してください*):

1. *評価モードの選択*: ツールバーの左上にある「**評価/デバッグの開始** (F5)」コマンドのプルダウンメニューから「**評価の開始**」を選択します (*下のスクリーンショットを参照*).

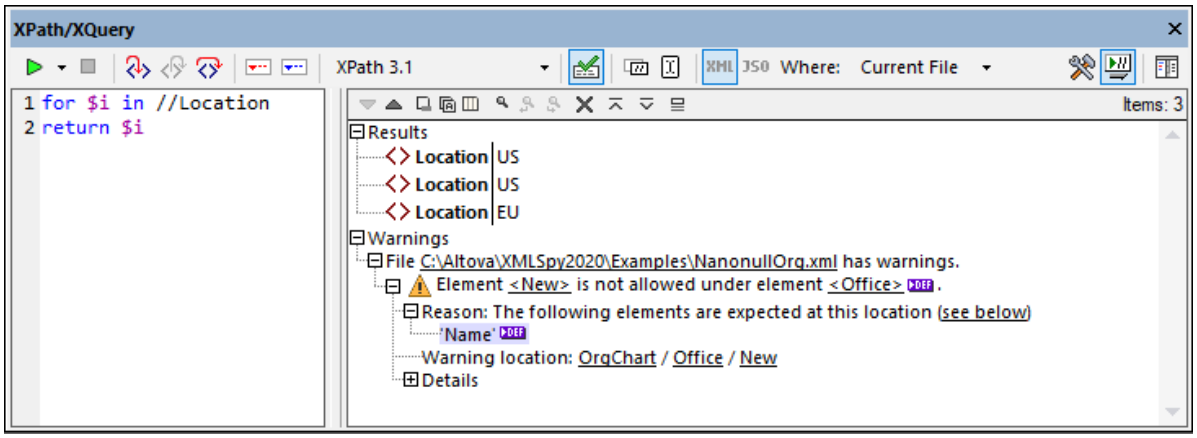
   

2. *水平的な/垂直的なレイアウト*: レイアウトを切り替えるには、(ツールバーの右側にある)「**水平的な/垂直的なレイアウト**」ボタンをクリックします。デフォルトは垂直的なレイアウトです。
3. *言語バージョンの選択*: ツールバー内で、評価する式のための言語バージョンを選択します。例えば、XPath 3.1 の場合、デフォルトはXPath 3.1 です。以下の *XML スキーマセレクタとXML スキーマフィールド* 511 も参照してください。
4. *式の入力*: 式ペイン内に、評価する式を入力します。式の作成を手助けするために、式ビルダー 520 の入力ヘルパーを使用します。式ペインの編集機能に関する詳細は、下記の*式ペイン内での編集* 511 を参照してください。
5. *評価スコープの選択*: ツールバー内の選択範囲オプション内で、式が評価されるファイルを選択します。オプション: *現在のファイル*、*ファイルを開く*、*(XMLSpy) プロジェクト*、または*フォルダー*。デフォルトは *現在のファイル*です。



現在のファイル が選択されている場合、現在アクティブなファイルが必要になります。*開かれているファイル* を選択すると現在 XMLSpy で開かれているすべてのファイルに対して式が評価されます。プロジェクトは現在アクティブな XMLSpy プロジェクト 950 を指しています。(XMLSpy プロジェクト内の外部フォルダーは *外部フォルダーをスキップ* にチェックすることにより除外することができます)。*フォルダー* オプションにより必要なフォルダーを参照することができ、XPath 式はこのフォルダー内の XML または JSON ファイルに対して評価されます。

6. *XML または JSON 評価の選択:* 現在のファイルが評価のスコープの場合、評価モード (XML または JSON) は自動的に (JSON-適合ファイルのためには JSON モードが、非-JSON ファイルのモードためには XML など) ドキュメントのファイル型の適合型 1458 により決定されます。両方のボタンは有効化され必要な評価モードを選択することができます。デフォルトは以前に選択されていたモードです。
7. *コンテキストノードの設定:* コンテキストノードを以下により設定することができます: (i) ルートノード (ii) アクティブなドキュメント内の現在の選択。 ツールバーボタン「XPath/XQuery のために現在の選択を起点とする」を介して2つの設定間を切り替えることができます。デフォルトの設定はルートノードです。
8. *XML 検証:* XML の評価 ツールバーボタンがオンに切り替えられている場合(デフォルトの設定)、評価されている XML ファイルが評価されます。エラーは結果ペイン内でレポートされる警告として扱われますが評価は継続されます (*下のスクリーンショット*)。



9. *式の評価:* ツールバーの切り替えオプション「入力中に評価」が選択されている場合、結果ペイン内に入力中の評価の結果が表示されます。このオプションが選択されていない場合、評価はツールバーの左上にある「評価/デバッグの開始 (F5)」をクリックして明示的に開始される必要があります。

評価プロシージャ内で使用されるツールバーボタン

	評価/デバッグの開始 (F5)	評価モードの選択を有効化し、評価を開始します。
	評価/デバッグの停止 (Shift+F5)	評価中に有効化されます。評価が長くかかる場合、または永続的なループで行われる場合に評価を中断する際に役に立ちます。
	水平的な/垂直的なレイアウト	水平的および垂直的レイアウト間の切り替えを行います。

	ビルダーに切り替える	式の作成を助けるコンテキストを考慮した入力ヘルパーを提供する式ビルダーに切り替えます。
	現在の選択範囲のXPath をコピーする	XMLドキュメント内のノードのロケーターパスを式ペイン内のカーソルの最後の場所にコピーします。
	XML/JSON 評価モード (XMLとJSON 評価モード間を切り替えます)	ペアのハイライトされているアイコンはアクティブなオプションです。評価スコープが複数のファイルである場合、両方のアイコンは有効化され、1つを選択することができます。それ以外の場合、評価モードはファイル型に従い自動的に検知されます。他のアイコンは無効化されます。
	現在の選択範囲をコンテキストとして設定する	ルートノードと現在の選択範囲の式コンテキストを切り替えます。
	入力中に評価	式の入力中に式の評価がオンに切り替えられます。
	XML の評価	オンに切り替えられるとターゲット XML ドキュメントが検証されます。

XML スキーマセレクタとXML スキーマフィールド
XML スキーマセレクタとXML スキーマフィールド オプションは特定のXPath 1.0 ケースの範囲の狭いサブセットのために使用され、XML スキーマ内で一意な識別制約が定義されている場合に役に立ちます。どちらかのオプションが選択されていると、名前テスト (とワイルドカード *) のみがXPath 式内で許可され、述語とXPath 式は使用できない場合があります。更に、*XML スキーマオプションセレクタ* オプションのために子軸の式のみが許可されます。, *XML スキーマフィールド* オプションに関しては子軸上の式が許可されます。詳細に関しては、W 3C's XML スキーマ: 構造と勧告 を参照してください。

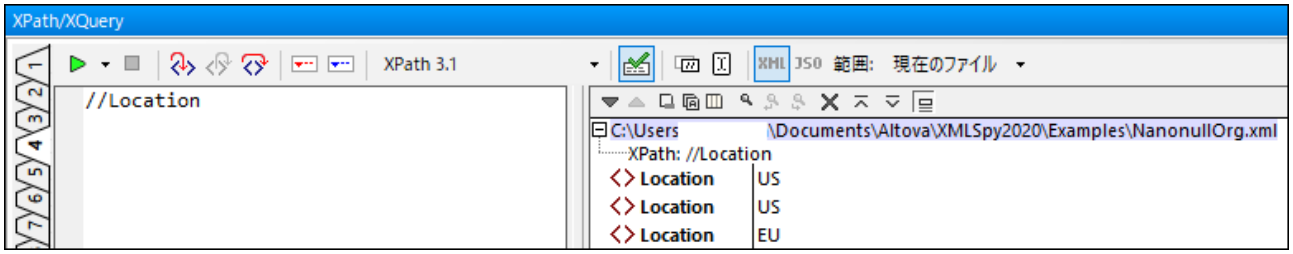
式ペイン内での編集
式ペイン内の式の編集について以下の点に注意してください:

- (わかりやすいように) 複数行で式を作成するには、**Return** キーを使用してください。
- 式フィールド内のテキストのサイズを大きくする/小さくするには、式フィールドをクリックし、**Ctrl** を押して、スクロールホイールを回します。*結果ペイン内でこの作用は適用されることに注意してください*。
- ノードのロケータパスを手動で入力する代わりにXMLSpy がパスを入力することができます。これを以下のように行います: (i) XPath 式内でロケータパスを挿入する箇所でカーソルをポイントします。(ii) ターゲットするノードの内部にカーソルをポイントします。(iii) ツールバー内で、ボタン「**現在の選択範囲のXPath をコピーする**」をクリックします。これにより、選択されたノードのロケーターパスは式内に挿入されます。ロケーターパスはドキュメントのルートノードから開始する絶対パスになります。

XQuery とJSON 評価

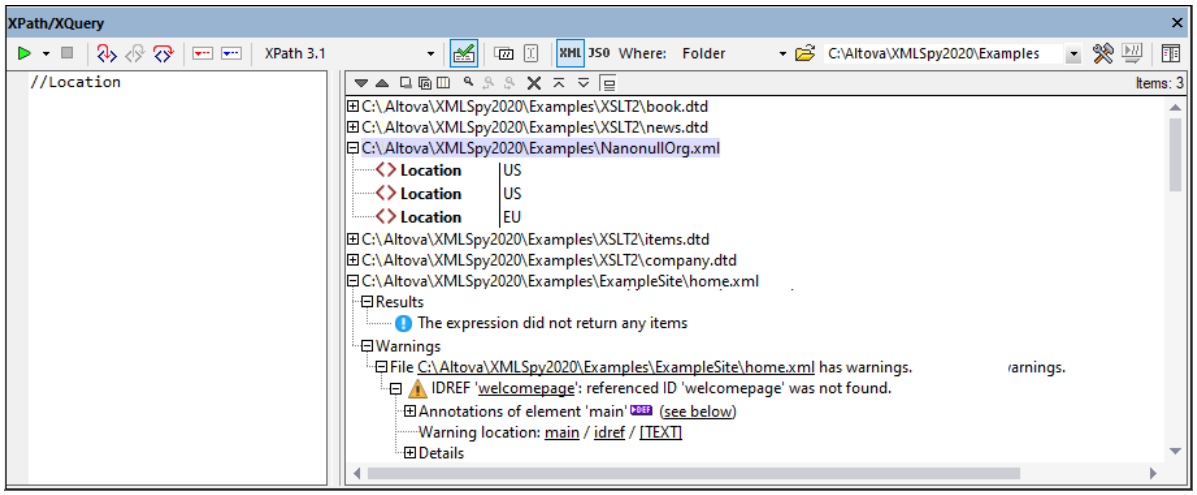
- XQuery 評価に関する詳細は、セクション XQuery 評価[451] を参照してください。(xq アイコンは XQuery 評価[451] のために、xqu アイコンは XQuery Update 式[456] のために使用されます。)
- JSON ドキュメントのクエリの説明に関しては JSON のためのXQuery 式[522] とXSLT/XQuery を使用したJSON 変換[625] .を参照してください。

結果ペイン
結果ペインは下のスクリーンショット内の右側で表示されています。結果ペインには自身のツールバーが存在することに注意してください。



結果ペインには以下の機能が搭載されています:

- (*下のスクリーンショットで表示されているとおり*) 出力内でXML ファイルとXPath 式の場所を表示する場合 *出力内にヘッダーを表示するアイコンをオン*に切り替えます。
- 結果リストは2 つの列により構成されています: (i) ノード名、またはデータ型 (ii) ノードのコンテンツ。
- XPath 式が(要素または属性などの) ノードを返す場合、ノードのコンテンツ全体がノードの値として表示されるかを選択することができます。これを行うには、*完全な結果の表示* をオンに切り替えます。
- 式生成リテラルとは異なり結果が(テキストノードを含む)ノードを含む場合、結果ペイン内でそのノードをクリックするとメインウィンドウ内でXML ドキュメント内の対応するノードがハイライトされます。
- (*選択個所オプションで指定されている*) 複数のファイルに対して評価が行われる場合、各ファイルの結果はそのファイルのパスの下に個別にリストされます(*下のスクリーンショットを参照*)。評価モードがXML の場合、XML-適合ファイルは評価され、他の型はスキップされます。評価モードがJSON の場合、JSON-適合ファイルが評価され、他の型はスキップされます。



- 結果が複数のファイルを含む場合、結果リスト内でファイル名をクリックすると、XMLSpy 内でファイルが開かれ、このファイルがアクティブなファイルになります。
- 結果サブラインの両方の列をコピーまたは、値列のみをコピーすることができます。全ての列をコピーするには、サブラインを右クリックして、「**全ての列を含みコピーする**」をオンに切り替えます(または、コマンドを結果ペインのツールバー内のアイコンをオンとオフに切り替えコマンドを切り替えることができます。) コピーするサブラインを右クリックして(サブラインのための) 「**サブラインのコピー**」または(全てのサブラインのための)「**全てをコピー**」を選択します。

結果ペインのツールバー
結果ペインのツールバーにはナビゲート、検索、およびコピー機能を提供するアイコンが含まれています。これらのアイコンは左から以下のテーブルで説明されています。対応するコマンドは結果リストアイテムのコンテキストメニューで使用することができます。

アイコン	動作

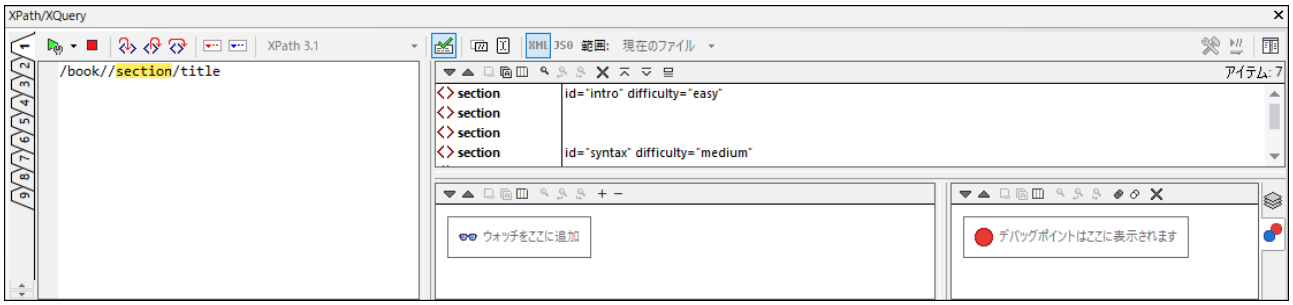
次、前	結果リストからそれぞれ次のおよび前のアイテムを選択します。
クリップボードに選択されたテキストラインをコピーする	クリップボードに選択された結果アイテムの値カラムをコピーします。全てのカラムをコピーするには、全てのカラムを含みコピーするコマンドに切り替えてください（*下を参照*）。
全てのメッセージをクリップボードにコピーする	空の値を含む値カラムのすべての結果アイテムをクリップボードにコピーします。各アイテムは異なるラインにコピーされます。
全てのカラムを含みコピーする	(i) 全てのカラム、または(ii) 値カラムのコピー間を切り替えます。カラムの区切りは単一のスペースです。
検索する	結果リスト内で特別文字を含む文字列を検索する検索ダイアログを開きます。
前を検索する	検索ダイアログ内に最後に入力された用語の前の発生を検索します。
次を検索する	検索ダイアログ内に最後に入力された用語の次の発生を検索します。
クリア	結果のリストをクリアする
複数のラインの結果を単一のラインに折りたたむ	結果アイテムの値カラムが複数ラインのテキストを含む場合、（新しいラインまたは文字を含む場合）、複数ラインと単一ラインの表示を切り替えることができます。
完全な結果を表示する	ノードの値としてノードのコンテンツ全体を表示します。
出力内にヘッダーを表示する	XML ファイルとXPath 式のロケーションを表示します。

10.3 式のデバッグ

XPath/XQuery ウィンドウのデバッグモードによりアクティブなファイルのコンテキスト内で XPath/XQuery 式をデバッグすることが可能になります (*下のスクリーンショット*)。

デバッグモード内では結果ペインに2つのペインが追加されました (*下のスクリーンショットを参照*):

- ペイン内で個別のタブを有するコールスタックとデバッグポイントペイン。
- ウォッチ式と変数が同じペインに表示されている変数とウォッチペイン。



式と結果ペインは水平的、または垂直的に隣接することができます。これらのレイアウトを切り替えるには、(ウィンドウのツールバーの右上の) **「水平的な/垂直的なレイアウト」** ボタンをクリックします。

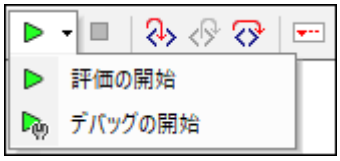
デバッガーモードには以下の機能が搭載されています:

- XPath 式がどのように評価されているか確認するために XPath 評価プロセスにステップインすることを可能にします。**ステップイン (F11)** ツールバーボタンをこの目的にために使用します。各評価ステップでは、式の一部が現在評価されている場合は黄色でハイライトされますがそのステップの評価の結果は結果ペイン内で表示されます (*上のスクリーンショットを参照*)。例えば、上のスクリーンショットでは、`book` 要素のすべての`section` 子孫要素が選択されています。
- 評価を一時停止する箇所にブレークポイントを設定し、この箇所の結果を確認します。評価をブレークポイントでのみ一時停止してステップスルーすることができます。このために **デバッグの開始 (F5)** ツールバーボタンを使用します。これは **ステップイン (F11)** を使用して全てのステップで一時停止するよりも素早く評価を行うことができます。を使用してください。
- トレースポイントを設定してトレースポイントとしてマークされたステップでの結果のレポートを確認することができます。評価は (ブレークポイント以外) 一時停止せず、トレースポイントの結果は結果ペイン内のリストに表示されます。
- ウォッチ式は (評価のドキュメントデータまたはアスペクトなど) 情報を確認するために使用することができます。これはブレークポイントに対して特に役に立ちます。
- 自身の値を含むスコープ内の変数は変数とウォッチ式ペイン内に表示されます。
- 評価ステップのプロセッサーの呼び出しはコールスタックとデバッグペインのコールスタックタブ内で表示されています。
- ブレークポイントとトレースポイントが設定されている場合、これらはコールスタックとデバッグペインのデバッグポイントタブに表示されます。

これらの機能の詳細に関しては、下の説明を参照してください。

デバッグモードのセットアップ

デバッグモードを切り替えるには、ツールバーの左上にある **「評価/デバッグの開始 (F5)」** コマンドのドロップダウンメニュー内の **「デバッグの開始」** をクリックします (*下のスクリーンショットを参照*)。デバッグモード内では、ウォッチ式、コールスタック、およびデバッグポイントペインが表示されます。現在の式のデバッグを開始するには、**「デバッグの開始」** または **「F5」** をクリックします。



現在のファイルでのみデバッグモードは作動することに注意してください。複数のファイルとは使用することができません。この結果、選択個所が*現在のファイル*に設定されていると、このモードに切り替えられます。

- *デバッグモードをセットアップするためのボタン*

	評価/デバッグの開始 (F5)	デバッガーを開始します。
	水平的な/垂直的なレイアウト	水平的および垂直的レイアウト間の切り替えを行います。
	ビルダーに切り替える	式の作成を助けるコンテキストを考慮した入力ヘルパーを提供する式ビルダーに切り替えます。
	現在の選択範囲の XPath をコピーする	XML ドキュメント内のノードのロケーターパスを式ペイン内のカーソルの最後の場所にコピーします。
XML JSON	**XML/JSON 評価モード**	ペアのハイライトされているアイコンはアクティブなオプションです。評価モードはファイル型に従い自動的に検知されます。他のアイコンは無効化されます。
	現在の選択範囲をコンテキストとして設定する	ルートノードと現在の選択範囲の式コンテキストを切り替えます。
	入力中に評価	式の入力中に式の評価がオンに切り替えられます。
	XML の評価	オンに切り替えられるとターゲット XML ドキュメントが検証されます。

デバッガーの実行

XPath/XQuery 式のデバッグのためのステップは通常以下で説明される通りです:

1. 式を実行する XML/JSON ファイルをアクティブなファイルにします。
2. XPath/XQuery 式を式ペインに入力します。
3. ブレークポイントまたはトレースポイントを設定します。ブレークポイントは評価が一時停止されるポイントです。トレースポイントは評価が記録されるポイントです。ですからトレースポイントは評価結果をトレースすることが可能なパスを提供します。
4. 「**デバッガーの開始**」をクリックすると、式内でブレークポイントがマークされていない限り、評価が終わりまで1つのステップで実行さ「**デバッガーの開始**」を繰り返しクリックして、各ブレークポイントを前進し評価の終わりまで到達します。
5. ステップイン/アウト/オーバー機能を使用してステップバイステップに評価を進めることができます。

- *デバッグのためのボタン*

	デバッガーの開始 (F5)	デバッガーを開始します。ブレークポイントで停止する以外、評価は直接最後まで行われます。
	デバッガーの停止 (Shift+F5)	評価を終了しデバッガーを停止します。

	ステップイン（F11）	評価をステップごとに進めます。
	ステップアウト（Shift+F11）	現在の評価ステップをステップアウトし、親ステップに移動します。
	ステップオーバー（Ctrl+F11）	子孫ステップにステップオーバーします。
	ブレークポイントの挿入/削除（F9）	カーソルが置かれている場所の式ステップでブレークポイントを挿入/削除します。
	トレースポイントの挿入/削除 (Shift+F9)	カーソルが置かれている場所の式ステップでトレースポイントを挿入/削除します。

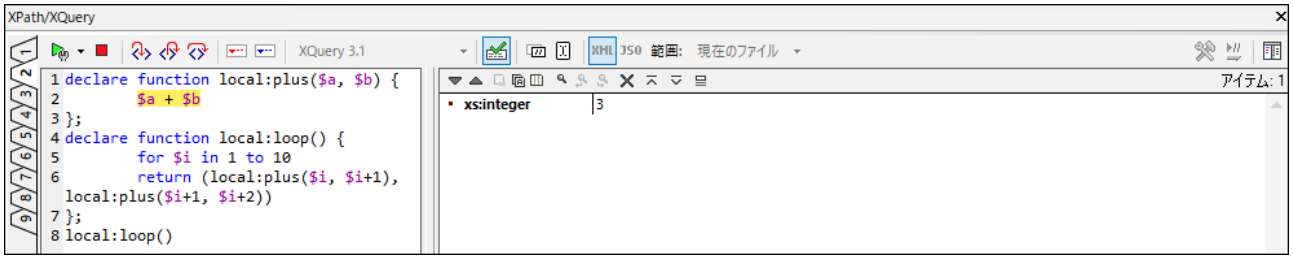
評価ステップのステップイン、アウト、オーバー

ステップイン 機能は評価をステップバイステップで行うことができます。このコマンドは評価の次のステップに移動します。現在のステップは式内でハイライトされて表示されています（*下のスクリーンショットを参照*）。*ステップアウト* 機能は現在のステップを高いレベルに移動しますが、*ステップオーバー* 機能は同じレベルより1つ低いレベルに移動します。下のXQuery 3.1 式を式ペインに張り付け *ステップ* ボタンをクリックしてステップ機能をテストすることができます。

ステップイン、ステップアウト、およびステップオーバー機能を試す XQuery 3.1 式

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

下のスクリーンショットは `$i=1` である最初のパススルーループ中に追加ステップに到達した際に処理が一時停止された評価を表示しています。この追加ステップでは、結果は（`1+2` の結果である）`3` を表示します。



ブレークポイント

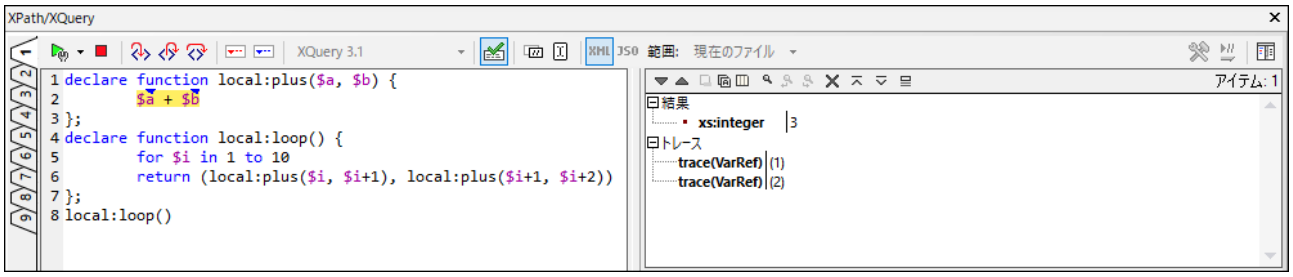
ブレークポイントは「**デバッガーの開始**」を使用して開始された後デバッガーを停止するポイントです。デバッガーがブレークポイントで停止すると、結果をチェックして「**ステップイン**」機能を使用し評価の次のステップの結果を表示することができます。ブレークポイントを設定するには、ブレークポイントを設定する箇所でカーソルを式内にポイントし、「**ブレークポイントの挿入/削除**（F9）」ツールバーボタンをクリックします。ブレークポイントは赤い点線のオーバーラインで表示されます。ブレークポイントを削除するには、選択し「**ブレークポイントの挿入/削除**（F9）」をクリックします。

デバッグポイント 518 を参照してください。

トレースポイント

トレースポイントは結果が記録されるポイントです。これらの結果は*結果タブのトレースツリー*内で表示されます（*下のスクリーンショットを参照*）。これにより式の特定の箇所のすべての評価結果を確認することができます。例えば、下のスクリーンショット内では、トレースポイントは `$a, $b, とlocal:plus($i, $i+1);` に設定されており、ループの最初の反復中のこれらのトレースポイントでの結果は以下の*トレースツリー*に表示されます: `$a=1, $b=2, とlocal:plus($i, $i+1)=3.`

トレースポイントを設定するには、トレースポイントを設定する箇所でカーソルを式内にポイントし、ツールバーボタン「**トレースポイントの挿入/削除 (Shift+F9)**」をクリックします。トレースポイントは赤い点線のオーバーラインで表示されます。（*下のスクリーンショットを参照*）. トレースポイントを削除するには、選択し「**トレースポイントの挿入/削除 (F9)**」をクリックします。

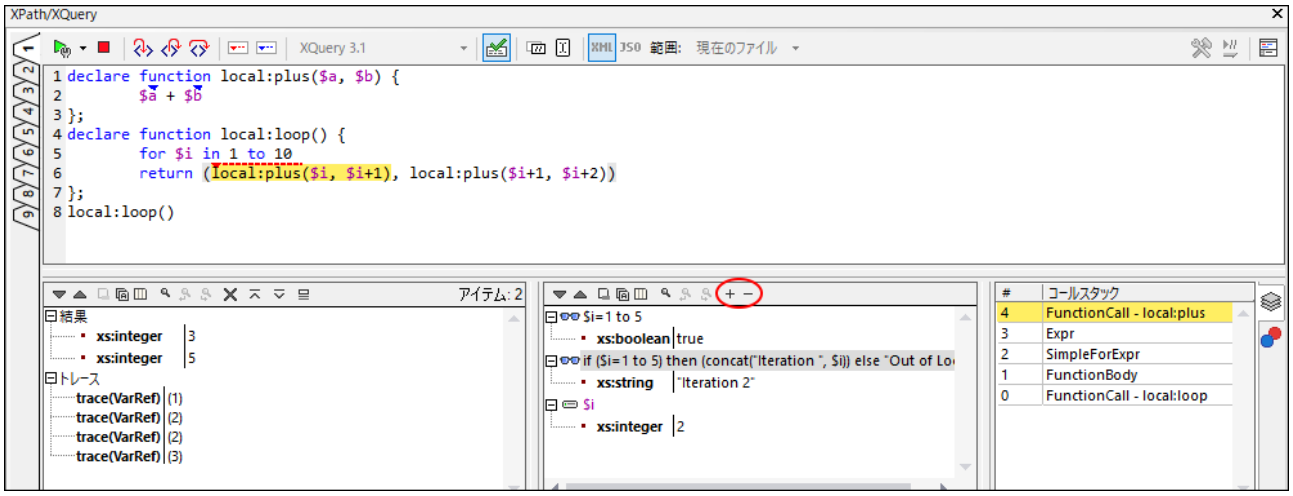


メモ　ブレークポンとトレースポイントがセットで式の一部である場合、オーバーラインは赤と青の点線で示されています。

デバッグポイント 518 を参照してください。

変数、ウォッチ式、およびコールスタック

変数とウォッチ式は変数とウォッチ式ペインに表示されます（*下のスクリーンショットの下の中央のペイン*）。



変数

式内で宣言されており、現在の評価ステップ内のスコープにある変数は対応する現在の値と共に表示されます。例えば、上のスクリーンショットでは、処理は`local:plus` 関数への呼び出しに達成しました。`$i` 変数は`local:loop` 関数のスコープ内にあり、`2` に増やされました。ですから、`$i` は現在の値と共に表示されています。 `$a` と`$b` 変数は`local:loop` 内にはなく、このためスコープ内にはありません。この結果これらは表示されません。

ウォッチ式
評価の開始前、または、評価の一時中止中にウォッチ式を入力することができます。次の目的のために使用することができます：

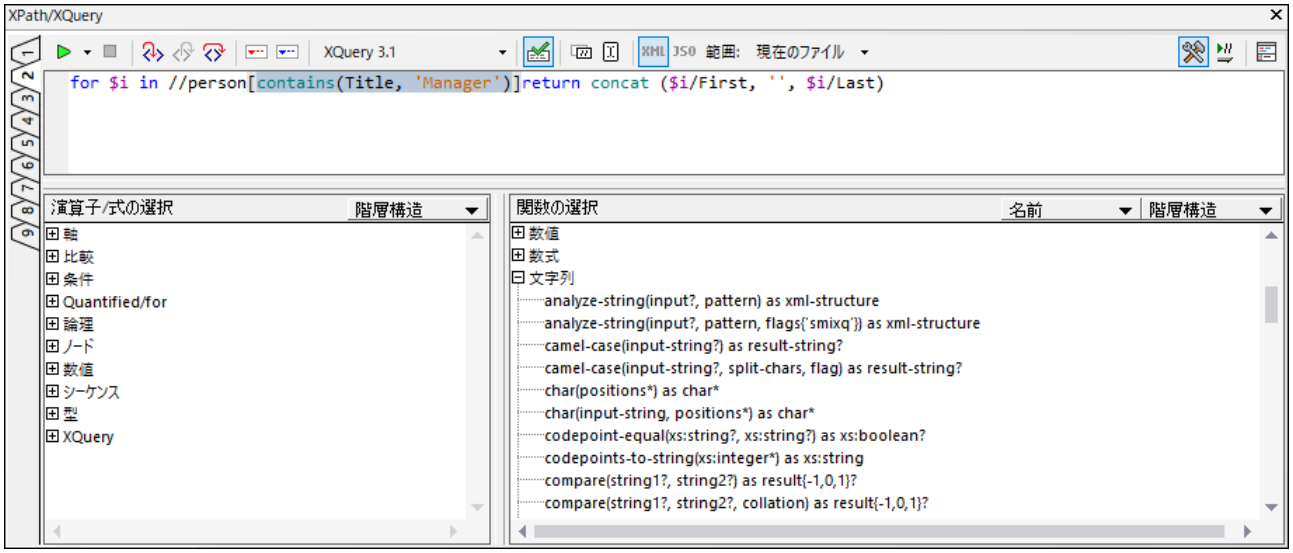
- 特定の条件をテストするため。例えば、上のスクリーンショットでは、ウォッチ式 `$i=1 to 5` は `$i` 変数が指定された範囲に値を持つかをテストするために使用されています。結果 `true` は条件が満たされたことを示しています。
- 特定のコンテキスト内のデータを検索するため。例えば、`Company` 要素のコンテキストでウォッチ式 `@id` を入力して、会社の顧客コードを XML ターゲットドキュメント内でルックアップすることができます。
- 追加データを生成するため。追加データを生成するため。例えば、上のスクリーンショットでは、ウォッチ式 `if ($i=1 to 5) then (concat("Iteration ", $i)) else "Out of Loop"` は現在の評価ループを反復する適切な文字列を生成することができます。

ウォッチ式を入力するには、ペインのルールバー内で(*上のスクリーンショットで赤い枠で囲まれてる*)「**ウォッチエントリの追加**」をクリックし、式を入力し、完了すると **Enter** をクリックします。ウォッチ式を削除するには、ツールバー内で選択し「**選択されたウォッチエントリを削除する**」をクリックします。デバッグ中に式が(例えば、変数の一つがスコープ外にあるなど)何らかの理由で正確に評価されない場合ウォッチ式は赤い色で表示されます。

コールスタック
コールスタックとデバッグポイントペインの*コールスタック*タブはその時点までのデバッグ内のプロセッサーの呼び出しを表示しています。*(上のスクリーンショット内の右下のペイン)*。現在のプロセッサーの呼び出しは黄色でハイライトされています。現在の評価ステップに直接導いた呼び出しが表示されます。例えば、上のスクリーンショットでは、現在の評価ステップは `local:plus` への関数呼び出しです。これは `local:loop` の2番目の反復ですが、最初の反復のプロセッサー呼び出しは表示されません。これはこれらの呼び出しは現在の関数呼び出しの並列レベルにあり、呼び出しを導いたわけではありません。

デバッグポイント

コールスタックとデバッグポイント ペインのデバッグポイントタブは式に設定することのできるブレークポイント（赤丸）とトレースポイント（青丸）を表示します*(下のスクリーンショット内の右下のペイン)*。各デバッグポイントはラインと文字番号と共にリストされています。例えば、`FunctionCall@7:13` はライン7、文字13にデバッグポイントが存在することを示しています。



以下の機能にご注意してください：

- ブレークポイントに関しては、*ブレーク条件の入力*をダブルクリックして、「**ブレークの条件**」を入力し、「Enter」を押して入力します。ブレークポイントは条件が `true` を評価すると有効化されます。例えば、上のスクリーンショットでは、ブレークの条件 `$i=3` は `$i` の値 3 がの場合のみこの関数呼び出し上でブレークポイントを有効化します。スクリーンショットは子のブレークポイントで一時停止された評価を表示しています。

(C) 2015-2021 Altova GmbH

- 全てのデバッグポイントの有効化/無効化を対応するツールボタンをクリックすることで行うことができます: **全てのデバッグポイントの有効化** と **全てのデバッグポイントの無効化** *(ボタンは上のスクリーンショット内でボタンは緑色の枠で囲まれています)*。デバッグポイントが無効化されると、有効化されるまで全て無効化されます。
- 対応するコンテキストメニュー内で個別のブレークポイントを有効化/無効化することができます。

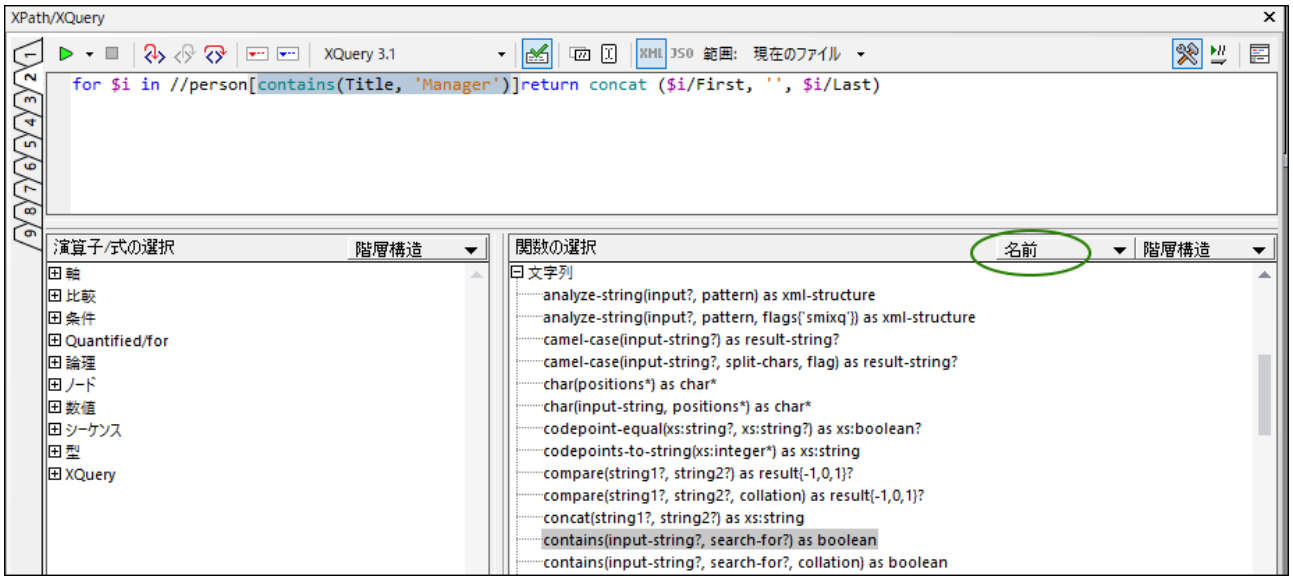
ペイン内のツールバーコマンド

デバッグモード内のXPath/XQuery ウィンドウにはナビゲート、検索、およびコピー機能を含むボタンが搭載されています(*上のスクリーンショットを参照*)。これらのアイコンは左から以下のテーブルで説明されています。対応するコマンドは結果リストアイテムのコンテキストメニューで使用することができます。

アイコン	動作
次、前	結果リストからそれぞれ次のおよび前のアイテムを選択します。
クリップボードに選択されたテキストラインをコピーする	クリップボードに選択された結果アイテムの値カラムをコピーします。全てのカラムをコピーするには、全てのカラムを含みコピーするコマンドに切り替えてください(*下を参照*)。
全てのメッセージをクリップボードにコピーする	空の値を含む値カラムのすべての結果アイテムをクリップボードにコピーします。各アイテムは異なるラインにコピーされます。
全てのカラムを含みコピーする	(i) 全てのカラム、または(ii) 値カラムのコピー間を切り替えます。カラムの区切りは単一のスペースです。
検索する	結果リスト内で特別文字を含む文字列を検索する検索ダイアログを開きます。
前を検索する	検索ダイアログ内に最後に入力された用語の前の発生を検索します。
次を検索する	検索ダイアログ内に最後に入力された用語の次の発生を検索します。
クリア	結果のリストをクリアする
複数のラインの結果を単一のラインに折りたたむ	結果アイテムの値カラムが複数ラインのテキストを含む場合、(新しいラインまたは文字を含む場合)、複数ラインと単一ラインの表示を切り替えることができます。
完全な結果を表示する	ノードの値としてノードのコンテンツ全体を表示します。
出力内にヘッダーを表示する	XML ファイルとXPath 式のロケーションを表示します。

10.4 式ビルダー

式ビルダー（または、ビルダーモード）はメインツールバー の「ビルダーモード」アイコンをクリックしてオンとオフに切り替えます（*下のスクリーンショットを参照*）。式ビルダーは両方のモードでオンに切り替えることができます（評価モードとデバッグモード[507]）。2つの入力ヘルパーペインが存在します: (i) 演算子と式、および(ii) 関数（*下のスクリーンショットを参照*）。両方のペイン内のアイテムは階層的にまたはフラットリストとして表示されます。各ペインの右上にあるドロップダウンリスト内からオプションを選択します（*下のスクリーンショットを参照*）。スクリーンショット内では、アイテムは階層的なグループにより表示されています。



式ビルダーの機能:

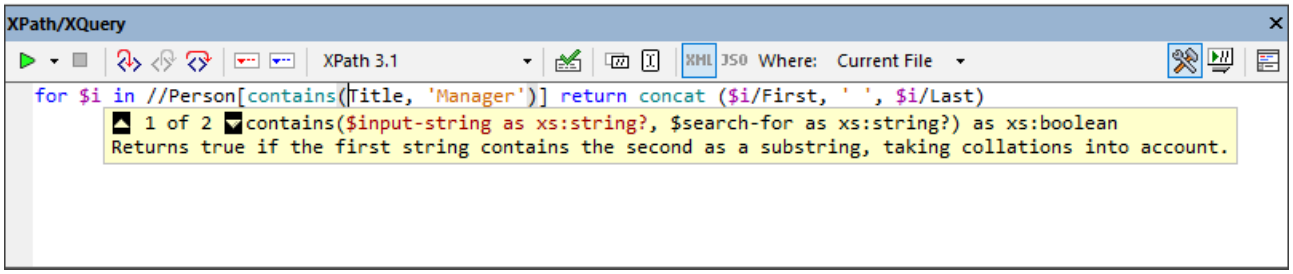
- 入力ヘルパーペイン内のアイテムのテキスト詳細を確認するためには、マウスをアイテムにポイントしてください。
- 各関数は署名（すなわち、引数、引数のデータ型、および関数の出力のデータ型）と共にリストされています。
- 1つ以上の署名が単一の関数名のために存在する場合、各署名は個別の関数としてリストされています。（これらのバリアントはその関数名のオーバーロードとして知られています。）上のスクリーンショットでは、例えば、`contains` 関数は2度表示されています: 2つの署名のためにそれぞれ表示されています。
- 引数は名前により、またはデータ型によりリストされています。関数ペインのタイトルバー内ののドロップダウンリストからオプションを選択します（*上のスクリーンショット内では緑色の枠で囲まれています。*）
- ペイン（演算子、式、または関数）内のアイテムをダブルクリックして、式内のカーソルの位置にそのアイテムを挿入します。式はプレースホルダ（`#` シンボルにより示される引数と共に挿入されます。
- (i) テキストが式の編集フィールドで選択されており(ii) プレースホルダを含む演算子、式または関数はダブルクリックして挿入することができます。選択済みのテキストはプレースホルダの代わりに挿入されます。これは（パス式などの）長いテキストを演算子、式、または関数に挿入する素早い方法です。
- ノードを選択しウィンドウのメインツールバーのボタン「現在の選択範囲のXPathをコピーする」をクリックしてアクティブドキュメント内のノードにパスを挿入することができます。

式ビルダー内で使用されるツールバーボタン

	水平的な/垂直的なレイアウト	水平的および垂直的なレイアウト間を切り替える

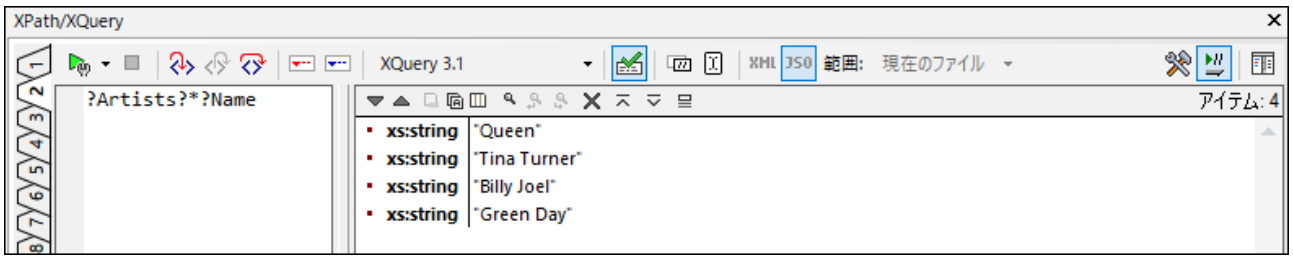
	ビルダーに切り替える	式を作成するためのコンテキストを考慮した入力ヘルパーを提供する式ビルダーモードに切り替える。
	現在の選択の XPath をコピーする	XML ドキュメント内のノードのロケーターパスを式ペイン内の最後のカーソルの位置にコピーします。
	コンテキストとして現在の選択範囲をセットする	ルートノードと現在の選択範囲間の式コンテキストを切り替えることができます。

式内に関数を入力した後、式ペイン内の関数名にカーソルをポイントすると関数の署名と関数のテキストの詳細が表示されます。関数のために 2 つ以上の署名が存在する場合、これらは表示の下のオーバーロード要素を共に示されます。関数の括弧内にカーソルをポイントし「Ctrl+Shift+Spacebar」を押すと、その関数名の多種のオーバーロードの署名を確認することができます *(下のスクリーンショットを参照)*。



10.5 JSONX のための Query 式

JSON ドキュメントは XPath/XQuery 出力ウィンドウ[113] 内に XPath/XQuery 3.1 クエリ式を入力するとクエリすることができます（*下のスクリーンショット参照*）。



JSON ドキュメントを評価するには以下を行います:

1. **XPath 3.1** アイコンまたは **XQuery 3.1** アイコンを選択します。
2. ウィンドウが JSON 評価モードであることを確認します。
3. XPath 3.1 または XQuery 3.1 式を入力します。
4. （ツールバーの左の）**「評価の開始」** をクリックします。

JSON のための XQuery 3.1 式

JSON データ構造は通常オブジェクトと配列を使用するため、 XQuery 3.1 lookup operator `?` が（基本的に XQuery パースペクティブからマップする） JSON オブジェクトと JSON 配列内のノードを検索するために使用されます。ノードをロケートするこの方法は XML ドキュメント内でノードをロケートするために パス式が作成される方法とは異なります。これらの中では、スラッシュ演算子 `/` が パス式内でステップを接続するために使用されています（例: `items/*`）。JSON のための式 XQuery 内では、スラッシュ演算子はノードをロケートするために使用されません。

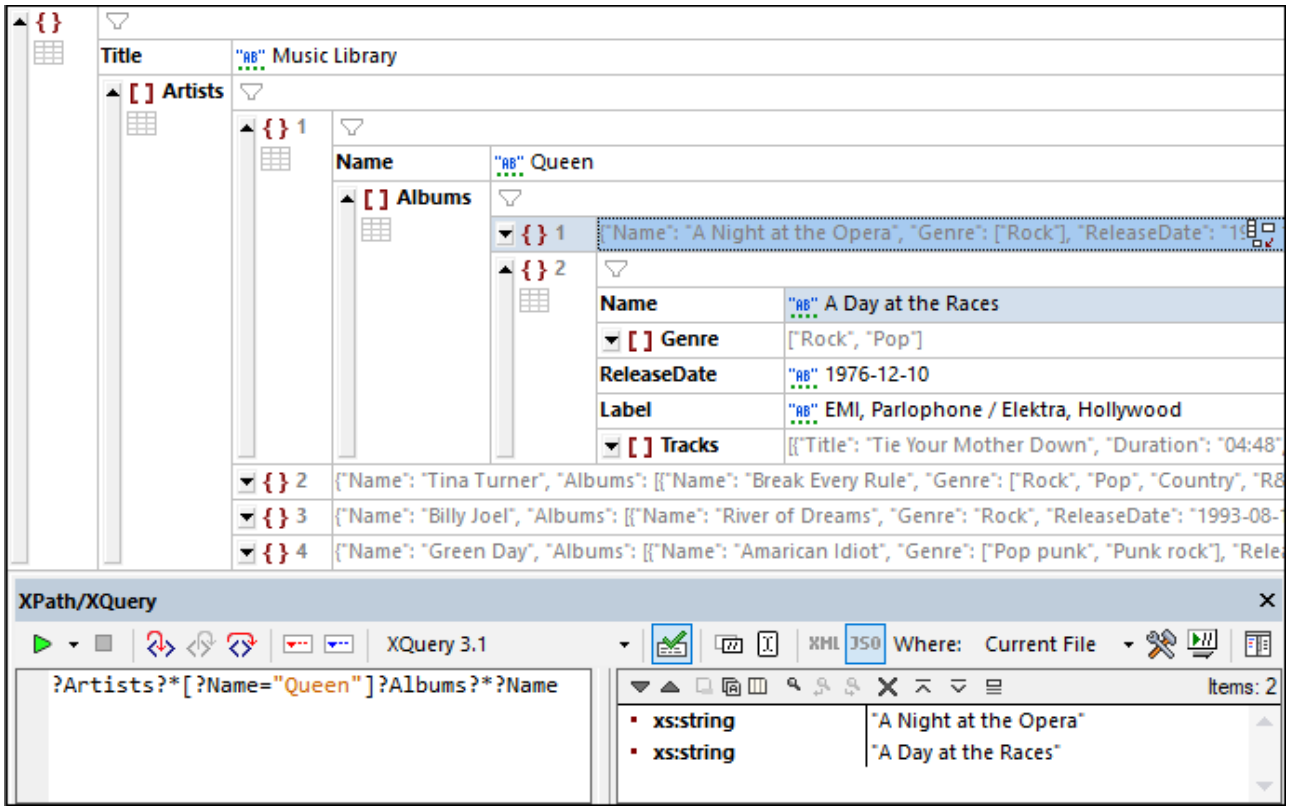
JSON のための XQuery 式のサンプル

`?items?*`
意味: 子ノード `items` をルックアップし、自信の子ノード全てをルックアップします。`items` はコンテキストノードの子ノードであることが期待されることにご注意してください。

`?Artists?1?Albums?2?Name`
意味: 子ノード `Artists` をルックアップし、最初の子ノードをルックアップします。そのノードの内部で子ノード `Albums` をルックアップし、2 番目の子ノードをルックアップします。2 番目の `Name` ノードが返されます。

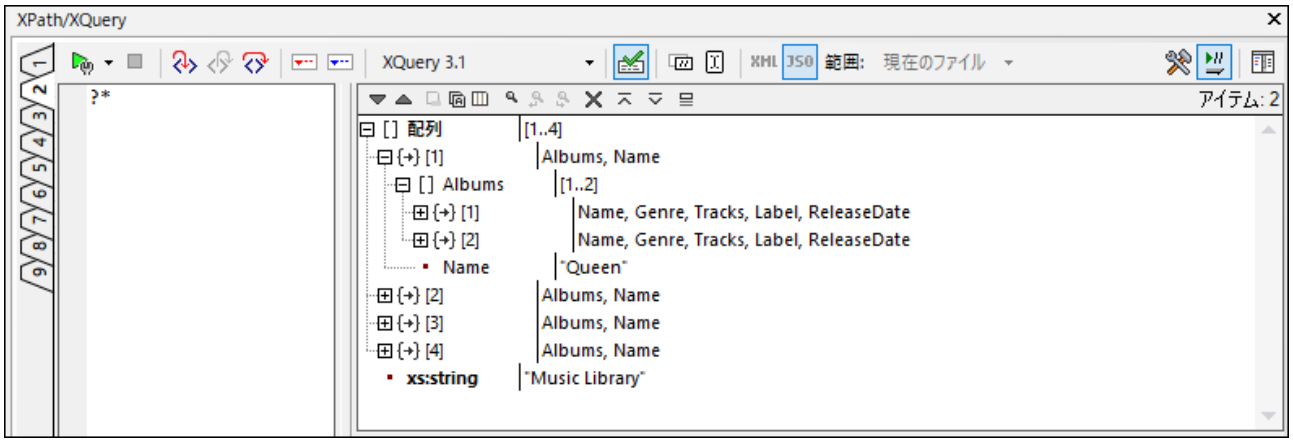
`?Tracks?*[contains(?Writer, 'Brian')]`
意味: 子ノード `Tracks` をルックアップし、自信のすべての子をルックアップします。子をルックアップ中に、各 `Writer` 子ノードの子をルックアップし、文字列 `'Brian'` を含むものを選択します。この式内には 3 つのルックアップ演算子が存在します。ノードセットがルックアップされる個所でそれぞれが新規ステップ内で使用されています。

`?Artists?*[?Name="Queen"]?Albums?*?Name`
意味: ルートオブジェクト内で lookup 子ノード `Artists` をルックアップし、`"Queen"` の値を持つ `Name` ノードを持つすべての子をルックアップします。コレアのノード内で、全ての子 Albums ノードをルックアップし、その子もルックアップします。これらの子内で、対応する `Name` ノードをルックアップ（して返）します。下のスクリーンショットでは、この式は XPath/XQuery ウィンドウ[506] 内でターゲット JSON ドキュメントの JSON グリッドビュー[600] 表示と共に表示されています。



JSON 評価のための結果ペイン

結果は選択された JSON コンポーネントを*結果* ペインの左側に太字で表示し、コンポーネントの値をペインの右側に表示します。下のスクリーンショットでは、結果は太字で表示されます。配列はメンバーを表示するために展開されています。



完全な結果の表示 アイコンがツールバー内でオンまたはオフに切り替えられているかに従い、マップと配列は短い、または詳細フォーマットで表示されています。ペインの左側のマップと配列はそれぞれのプラスアイコンをクリックして展開することができます。

10.6 注意点

XPath 1.0 条件式

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

XPath 2.0 と3.1 条件式

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

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

XPath 2.0 と3.1 内のデータ型

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

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

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

文字とエンティティ参照の文字列長

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

(C) 2015-2021 Altova GmbH

XPath 2.0 および 3.1 関数サポート

付録[1980]を参照ください。

11 Authentic

(以下のスクリーンショットにある) Authentic View はお使いの XML ドキュメントをグラフィカルな形式で表示したものです。XML ドキュメントのマークアップの表示を隠し、入力フィールド、コンボボックス、ラジオボタンといったフォーマットやデータ入力機能を適切な形で表示します。Authentic View を使ってユーザーが入力したデータは XML ファイルにも反映されます。

Authentic View でデータの閲覧と入力を行うためには、Altova StyleVision により作成された **StyleVision Power Stylesheet (SPS)** に XML ドキュメントを関連付ける必要があります。SPS (.sps ファイル) は、実質的には XSLT スタイルシートで、データ入力を行うことができる XML ファイルの出力フォーマットを指定することができます。Authentic View ユーザーは、XML ファイルや DB へのデータ入力をグラフィカルに行うことができます。SPS はスキーマをベースにしており、スキーマに特化したファイルです。Authentic View ビューで SPS を使った XML ファイルの編集を行うには、XML ファイルがベースにしているスキーマファイルを使用しなければなりません。

Authentic View を使う

- XML ファイルが開かれている場合、メインウィンドウの下部にある **Authentic** ボタンをクリックすることで Authentic View に切り替えることができます。SPS が XML ファイルにまだ割り当てられていない場合、割り当てを行うよう促されます。使用できる SPS は、XML ファイルがベースにしているスキーマをベースにしたもので無ければなりません。
- 「**ファイル | 新規作成**」コマンドを選択し、「StyleVision スタイルシートを選択」ボタンを選択すると、新規 XML ファイルが作成され、Authentic View で表示されます。この新しいファイルは、選択した SPS ファイルに割り当てられたテンプレートファイルになります。このファイルは、既に表示されている様に様々な数の開始データを持つことができます。この開始データは (テンプレート) XML ファイルに含まれ、SPS ファイルと関連付けることもできます。XML ファイルの Authentic View が表示されたら、データの入力とファイルの保存を行うことができます。

(C) 2015-2021 Altova GmbH

- 「Authentic | 新規作成」 コマンドを使用してSPS を開くことができます。テンプレート XML ファイルがSPS に割り当てられている場合、テンプレート XML ファイル内のデータは、Authentic View.により作成されたXML ドキュメントテンプレートのデータを開始するために使用されます。

このセクションでは...

このセクションはAuthentic ビューのチュートリアルを含んでおり、Authentic View の使い方を説明した後、Authentic ビューでの編集や個々の編集機能の詳細について説明します。

Authentic View について更に詳しく

Authentic View に関する更に詳しい情報については、(i) Authentic View 編集ウィンドウについて解説された、このドキュメントの **Authentic** 526 | Authentic ビューインターフェイス 540 を参照するか、(ii) このドキュメントのユーザーリファレンスにある Authentic メニュー 1294 セクションを参照ください。

11.1 Authentic View チュートリアル

Authentic View 内の視覚的な WYSIWYG インターフェイス内で Microsoft Word などのワードプロセッサーアプリケーション内で作業するように、XML ドキュメントを編集することができます（*下のスクリーンショット*）。XML ドキュメントの Authentic View を管理するスタイルシート内で既に定義されているため、ドキュメントの書式設定を心配する必要はありません（StyleVision Power Stylesheet はこのチュートリアル内では SPS と省略されています）。*スタイルシート*は Altova の StyleVision 製品の*スタイルシート*デザイナーを使用して作成されます。

Authentic View 内で XML ドキュメントを編集するには2つのユーザーアクションが含まれます: (i) ドキュメントの構造の編集（例えば、段落とヘッドラインなどのドキュメントの一部を追加、または、削除するなど）、および (ii) データの挿入（ドキュメントの一部のコンテンツ）.

このチュートリアルでは次のステップについて説明されています:

- Authentic View 内で XML ドキュメントを開く。Authentic View 編集の主要な必要条件は、XML ドキュメントが SPS ファイルに関連付けられていることです。
- Authentic View インターフェイスの主要な編集メカニズムに関する概要を確認してください。
- ノードを挿入または削除することによりドキュメントの構造を編集します。
- XML ドキュメント内にデータ挿入します。
- (i) 属性入力ヘルパーを介した属性の値と (ii) エンティティの値を入力します。
- ドキュメントを印刷します。

このチュートリアルは使用開始の準備をするためであり、簡単にまとめられています。Authentic View インターフェイス[540] セクション内で追加の参照マテリアルと機能の説明を確認することができます。

チュートリアルの必要条件

チュートリアルで必要な全てのファイルはAltova アプリケーションの Examples フォルダー内にあります。これらのファイルは、以下のとおりです:

- NanonullOrg.xml (XML ドキュメント)
- NanonullOrg.sps (XML ドキュメントがリンクされている StyleVision Power Stylesheet)
- NanonullOrg.xsd (XML ドキュメントと StyleVision Power Stylesheet がベースにされ、リンクされている XML スキーマ)
- nanonull.gif and Altova_right_300.gif (チュートリアルで使用されるイメージ)

メモ このチュートリアルのある時点で、(ドキュメントの Authentic View とは異なり) XML ドキュメントの XML テキストを確認するように促されます。(Authentic Desktop と Authentic Browser 同様) Altova 製品エディションにテキストビューが搭載されていない場合、Wordpad または Notepad のようなテキスト エディターを使用して XML ドキュメントのテキストを確認してください。

注意点: 問題が発生した際に元のファイルにアクセスできるように、NanonullOrg.xml をチュートリアルのためにコピーすることが奨励されます。

11.1.1 AUTH-VIEW 内で XML ドキュメントを開く

Authentic View 内で、既存の XML ドキュメントを編集、または、新規の XML ドキュメントを作成して編集することができます。(このセクションで説明されているとおり) このチュートリアルでは、Authentic View 内で既存の XML ドキュメントを開き、(次のセクションで) 編集の方法を学びます。更に、Authentic View 内での編集のために新規 XML ドキュメント作成の方法について説明されています。

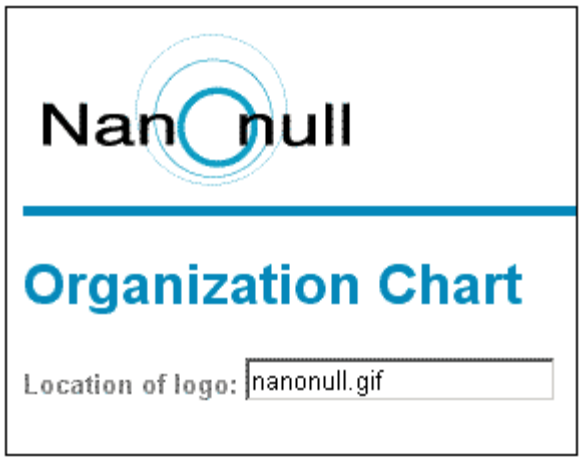
既存の XML ドキュメントを開く

NanonullOrg.xml というファイルを開きます。Altova アプリケーションの Examples フォルダー内にあり、以下の2つの方法により NanonullOrg.xml を開くことができます:

- Altova 製品内の「**ファイル | 開く**」をクリックし、表示されるダイアログ内から NanonullOrg.xml を選択し、「**開く**」をクリックします。
- ウィンドウ Explorer を開き、ファイルを検索し、右クリックして、ファイルを開くためのアプリケーションとして Altova 製品を選択します。

Authentic View 内でファイル NanonullOrg.xml を直接開くことができます(*下のスクリーンショット*)。理由:

1. StyleVision Power Stylesheet (SPS) が既にファイルに割り当てられており、
2. SPS ファイルの割り当てが既に割り当てられている場合、表示タブのオプションダイアログ(「**ツール | オプション**」)で、Authentic View 内で XML ファイルを開くオプションを選択します(それ以外の場合、ファイルはテキストビュー内で開かれます)。



留意点: SPS はAuthentic View 内でXMLドキュメントがどのように表示されるかを表示し管理します。SPS が存在しない場合、ドキュメントのAuthentic View は存在しません。

SPS をベースにした新規のXMLドキュメント

SPS をベースにした新規のXMLドキュメントを作成することができます。これを2つの方法で行うことができます: **「ファイル | 新規作成」**メニューコマンドと**「Authentic | 新規のドキュメント」**メニューコマンドを使用する。SPS が両方の方法で選択されます。

「ファイル | 新規作成」を使用する方法

1. **「ファイル | 新規作成」**を選択します。
2. 新規のドキュメントの作成ダイアログ内で、希望するSPS を参照します。

テンプレート XML ファイルがSPS に割り当てられている場合、Authentic View 内で作成されたドキュメントテンプレートの開始データとしてテンプレート XML ファイル内のデータが使用されます。

「Authentic | 新規のドキュメント」を使用する方法

1. **「Authentic | 新規のドキュメント」**を選択します。
2. 「新規のドキュメントの作成」ダイアログ内で希望するSPS を参照します。

テンプレート XML ファイルがSPS に割り当てられている場合、テンプレート XML ファイル内のデータは、Authentic View 内で作成されるXMLドキュメントテンプレートの開始データとして使用されます。

11.1.2 Authentic View インターフェイス

Authentic View 編集インターフェイスはドキュメントデータを入力し編集するメインウィンドウと2つの入力ヘルパーにより構成されています。ドキュメントの編集は簡単です。ドキュメントのマークアップを確認する場合、マークアップタグをオンに切り替え、ドキュメントのコンテンツの入力を開始します。ドキュメントの構造を変更するには、コンテキストメニュー、または、要素入力ヘルパーを使用します。

XML ノードタグ(ドキュメントマークアップ)の表示

XMLドキュメントは、基本的には、ノードの階層構造です。例:

<ドキュメントRoot>

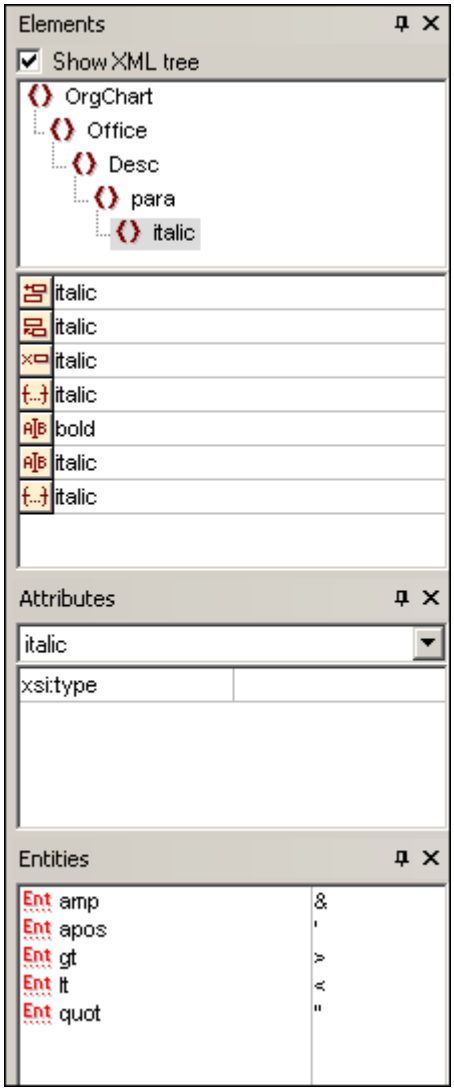
```
  <Person id="ABC001">
        <Name>Alpha Beta</Name>
        <Address>Some Address</Address>
        <Tel>1234567</Tel>
    </Person>
</ドキュメントRoot >
```

デフォルトでは、ノードタグ Authentic View 内では表示されていません。メニューアイテム「Authentic | 大きなマークアップを表示する」（または、 ツールバーアイコン）を選択してノードタグをオンに切り替えることができます。大きなマークアップタグにはそれぞれのノードの名前が含まれています。更に、（タグ内にノード名の無い）小さなマークアップと複合型マークアップを選択することができます（スタイルシートのデザイナーにより大きい、小さい、または、マークアップタグの無い複合型マークアップが定義されます。ドキュメントのためのデフォルトの複合型マークアップにはマークアップが存在しません）。

Altova 製品 またはテキストエディターのテキストビュー内で XML ドキュメントのテキストを確認することができます。

入力ヘルパー

インターフェイス内には、デフォルトでアプリケーションウィンドウの右端にある3つの入力ヘルパーが存在します（下のスクリーンショット）。これらは、要素、属性、および、エンティティ入力ヘルパーです。



要素入力ヘルパー: 要素入力ヘルパーは、カーソルの現在の場所、または、メインウィンドウ内の選択された場所に挿入、または、から削除することができる要素を表示します。入力ヘルパーは状況依存であることに注意してください。コンテキストはカーソルの位置、または、選択範囲により異なります。入力ヘルパーのコンテンツは以下の方法で変更することができます: 要素入力ヘルパーのXML ツリー内で他のノードが選択された場合、そのノードに関連する要素が入力ヘルパー内で表示されます。入力ヘルパーの上のXML ツリーの表示チェックボックスをチェックして、XML ツリーを表示するために要素入力ヘルパーを展開することができます（*上のスクリーンショットを参照*してください）。XML ツリーは、トップレベル要素ノードからマインウィンドウ内で選択されたノードまでのノードの階層構造を表示します。

属性入力ヘルパー: メインウィンドウ内で選択されている要素の属性、および、属性の値を表示します。属性の値は、属性入力ヘルパー内に入力し編集することができます。属性入力ヘルパーのコンボボックス内の選択のために トップレベルの要素選択された要素までの要素ノードを使用することができます。

コンボボックスのドロップダウンリストから要素を選択すると、要素の属性が入力ヘルパー内に表示されます。入力ヘルパー内で要素の属性を編集することができます。

エンティティの入力ヘルパー: エンティティの入力ヘルパーは状況依存ではなく、ドキュメントのために選択された全てのエンティティを表示します。エンティティをダブルクリックすると、カーソルの位置にエンティティを入力することができます。ドキュメントへのエンティティの追加方法はセクション Authentic View インターフェイス[540] で説明されています。

コンテキストメニュー

Authentic View ドキュメント内を右クリックすると、その場所（ノード）に関連したコンテキストメニューがポップアップされます。コンテキストメニューは以下を行うことのできるコマンドを提供します:

- 選択されたノードの前、または、あとにノードを挿入します。サブメニューはそれぞれの挿入場所で使用を許可されたノードのリストを表示しています。
- （スキーマにより許可されている場合）選択されたノード、または、削除することのできる祖先要素を削除します。（スキーマに従い）削除される可能性のあるノードはサブメニューにリストされています。
- エンティティとCDATA セクションを挿入します。ドキュメントのために宣言されているエンティティはサブメニュー内にリストされています。CDATA セクションはテキスト内にのみ挿入することができます。
- ドキュメントのコンテンツの切り取り、コピー、（XML 、または、テキストとしての貼り付けを含む）貼り付け、および削除。

メモ インターフェイスに関する詳細は、次を参照してください: Authentic View インターフェイス[540]

11.1.3 ノードを操作する

Authentic View XML ドキュメント内で発生するノードには2 つの主要な種類があります: **要素ノード** と **属性ノード**。これらのノード はオンに切り替えることのできる[530] タグによりマークアップされます。（マークアップされていない）テキストノードと 周りのテキストから区別するためにマークアップされている）CDATA セクションノード などドキュメント内に他のノードも存在します。

このセクションで説明されたノードの操作は要素ノードと属性ノードのみを参照します。このセクション内で説明されたオペレーションを試す場合、大きなマークアップがオンに切り替えられていることが[530] 勧められます。

メモ **同じレベル、または、ハイレベルの要素** のみが選択された要素の前、または、後に挿入されることに注意してください。同じレベルの 要素は **兄弟** です。段落要素の兄弟は、他の段落要素ですが、リスト、テーブル、イメージであることができます。兄弟は要素の前、または、後に発生することができます。 ハイレベルの要素は **祖先** 要素と祖先の兄弟です。段落要素に関しては、祖先要素はセクション、チャプター、アーティクル、などである場合があります。有効な XML ファイル内の段落には既に祖先が存在します。結果として、Authentic View 内に高いレベルの要素を追加すると、関連する祖先の兄弟として新規の要素が作成されます。例えば、セクション 要素が段落の後に挿入されると、現在の段落要素を含むセクションの兄弟として作成されます。

ノードの操作の実行

ノードの操作は、コンテキストメニュー内でコマンドを選択、または、要素入力ヘルパー[531]内でノード操作エントリをクリックすることで行うことができます。一部の場合、要素、または、属性は、ドキュメントのAuthentic View 内のノードリンクの追加[533]をクリックして追加することができます。段落またはリストアイテムとして定義されている要素の特別な場合は、このような要素内を作成する際、Enter キー[534]を押すと、この種類の新規の兄弟要素が作成されます。このセクションは要素の適用[534]、ノードの削除[534]、および要素をクリアする[534]メカニズムを使用して、ノードを作成、または、削除するかについて説明されています。

要素の挿入

要素は次の場所に挿入することができます:

- 要素ノード内のカーソルの場所に挿入することができます。コンテキストメニューの「挿入」コマンドのサブメニュー内にその場所に挿入することのできる要素がリストされます。要素入力ヘルパー内では、その場所に挿入することのできる要素は アイコンにより表示されます。NanonullOrg.xml ドキュメント内では、 para 要素内にカーソルを置き、コンテキストメニューと要素入力ヘルパーを使用して bold とitalic 要素を作成します。
- スキーマにより許可されている場合、選択された要素、または、その祖先の前、または、後に挿入することができます。必要とされる要素をロールアウトされるサブメニューから選択します。要素入力ヘルパー内では、選択された要素の前、または、後に挿入することのできる要素は および アイコンにより表示されます。要素入力ヘルパー内では、選択された要素の前後のみに要素を挿入することができます。祖先要素の前後に要素を挿入することはできません。para 要素と社員をリストするテーブル内にカーソルを置いてこのコマンドを試してください。

ノードリンクの追加

要素、または、属性がドキュメントデザイン内に含まれ、XML ドキュメント内で存在しない場合、ノードが指定されているドキュメント内の場所でノードリンクの追加が表示されます。このリンクを確認するために、テキスト、ロゴの場所を持つライン内で、CompanyLogo 要素内で @href ノードを選択し、(「削除」キーを押して) 削除します。編集された CompanyLogo 要素内で add @href リンクが表示されます(下のスクリーンショット)。リンクをクリックすると、@href ノードがXML ドキュメントに追加されます。@href ノードがこのように表示されるようにデザインを指定しているため、@href タグ内のテキストボックスは、はこのように表示されます。@href ノード(または、コンテンツ)の値を入力する必要があるので、テキスト nanonull.gif を入力します。

要素のコンテンツモデルが曖昧な場合、例えば、子要素のシーケンスを指定する場合、順序に決まりはありませんので add... が表示されます。ノード名が指定されていないことに注意してください。リンクをクリックすると、有効に挿入することのできる要素のリストがポップアップします。

メモ ノードリンクの追加はドキュメントテンプレート内に直接表示されます。コンテキストメニュー、または、要素入力ヘルパーに対応するエントリはありません。

Enter キーを使用して新規要素を作成する

要素がスタイルシート デザイナーにより段落、または、リストアイテムとして書式設定されている場合、このようなノード内で「Enter」キーを押すと、点在のノードの後にこのような種類の新規のノードが挿入されます。NanonullOrg.xml ドキュメント内でこのメカニズムを試すことができます。(終了タグの前の) para ノードの終わりに移動し、「Enter」を押すことで実行することができます。

要素の適用

(テキストと子要素の両方を含む) 複合型 コンテンツの要素内では、テキスト コンテンツの一部を選択し、許可されている子要素を適用することができます。選択されたテキストは適用された要素のコンテンツになります。要素を適用するには、コンテキスト メニュー内で、「適用」を選択し、適用することのできる要素から選択します (選択されたテキストに要素が適用されない場合、「適用」コマンドはコンテキスト メニュー内に表示されません)。要素入力ヘルパー内では、選択のために適用することのできる要素は、 アイコンで表示されます。NanonullOrg.xml ドキュメント内で、複合型 コンテンツ para 要素内のテキストを選択し、bold と italic 要素の適用を試してください。

スタイルシート デザイナーが要素を適用するツール バーアイコンを作成する場合があります。NanonullOrg.xml ドキュメント内では、bold と italic 要素をアプリケーションの Authentic ツール バー内の太字と斜体アイコンをクリックすることで適用できます。

要素の削除

ドキュメントを無効にしない場合、ノードを削除することができます。ノードの削除はその全てのコンテンツを削除します。コンテキスト メニュー内の「削除」コマンドを使用してノードを削除することができます。「削除」コマンドがハイライトされると、選択されたノードからドキュメントのトップ レベルのノードまで削除する全てのノードを含むサブメニューがポップアップされます。削除するためにノード選択するには、カーソルをノード内に置く、または、ノード (または、ノードの一部) をハイライトします。要素入力ヘルパー内では、削除することのできるノードは アイコンで表されます。削除可能なノードは選択後に「削除」キーを押すことにより削除することができます。NanonullOrg.xml ドキュメント内で、上記のメカニズムを使用していくつかのノードを削除してください。**Ctrl+Z** を使用して変更を元に戻すことができます。

要素の作成

複合型 コンテンツ (テキストと要素の子) を持つ要素の子である要素ノードをクリアすることができます。ノードが選択されると、または、入力ポイントとしてノードにカーソルがノード内に置かれると、要素の全体をクリアすることができます。要素内のテキスト フラグメントはテキスト フラグメントをハイライトすることにより要素マークアップをクリアすることができます。選択を決定し、コンテキスト メニュー内の「クリア」を選択して、要素をクリアします。要素入力ヘルパー内では、特定の選択のためにクリアする要素は アイコン (挿入ポイント 選択) と アイコン (範囲の選択) により示されます。NanonullOrg.xml ドキュメント内で、(複合型 コンテンツを持つ) para の bold と italic 子要素を使用して、クリア メカニズムを試してください。

テーブルとテーブルの構造

Authentic View テーブルには2つの型が存在します:

- *SPS テーブル (静的と動的)*。スタイルシート デザイナーにより SPS テーブルの大まかな構造は決定されます。この広義な構造では、許可されている唯一の構造的な変更はコンテンツ優先のみです。例えば、新規の行を動的な SPS テーブルに追加することができます。
- *XML テーブル* は特定のノードのコンテンツを表示します。(個人の特定の情報などを含んだ) テーブルなど。スタイルシート デザイナーが XML テーブルとしてこのノードの作成を有効化している場合、テーブルの構造を決定し、コンテンツを編集することができます。XML テーブルはAuthentic View 内のテーブル 557 セクションで詳細に説明されています。

11.1.4 Authentic View 内にデータを挿入する

Authentic View メインウィンドウ内のXMLドキュメントに直接データが入力されます。追加して、属性のためにデータ（属性の値）を属性入力ヘルパーに挿入[536] することができます。データは（i）直接テキストとして入力されます。または、（ii）定義済みのテキストエントリにマップされる データ入力デバイス内のオプションを選択することにより入力されます。

テキストコンテンツの追加

Authentic View のメインウィンドウ内にテキストとして直接入力することができます。コンテンツを挿入するために、カーソルをテキストを入力する箇所にポイントし、入力を開始します。クリップボードからテキストをコピーして、ドキュメントに貼り付けることもできます。コンテンツは、「Caps」と「削除」キーなどの通常の編集機能を使用して編集することができます。例えば、編集するテキストをハイライトして、「Caps」キーがオンのままに変わりのテキストを入力することができます。

例えば、企業の名前を変更するには、`Office` の`Name` フィールド内でカーソルを Nanonull の後にポイントし、Nanonull, Inc. から Nanonull USA, Inc. に名前を変更するために`USA` 入力します。

テキストを編集することができる場合、カーソルをポイントしてハイライトします。それ以外の場合ハイライトすることができません。アドレスブロック内の「Street」、「City」、または「State/Zip」などの（フィールドの値ではなく）フィールド名 を変更しようと試みてください。StyleVision Power Stylesheet から生成されたテキストはXML コンテンツではないため、このテキスト内にはカーソルをポイントすることができません。

特別文字とエンティティの挿入

データの挿入する場合、次のコンテンツの型は特定の方法で扱われます:

- *特殊文字は、XML マークアップ*（アンパサンド、アポストロフィ、より大記号（>）、より小記号（<）、および引用符）のために使用されます。これらの文字は内蔵のエンティティ[537] として使用することができ、エンティティの入力ヘルパー内の対応するエンティティをダブルクリックすることによりドキュメント内に入力することができます。（例えば、プログラムコードリスティング内では） CDATA セクション内に挿入することができます。CDATA セクションを入力するには、CDATA セクションを入力する場所を右クリックして、コンテキストメニューから「**CDATA セクションの挿入**」を選択します。XML プロセッサーはCDATA セクション内の全てのマークアップ文字を無視します。CDATA セクション内で特殊文字を使用する場合、エンティティの参照ではなくその文字を入力します。
- *キーボードを使用して入力できない特殊文字* はシステムの文字マップからコピーういてドキュメント内の必要とする場所に貼り付けます。
- *頻繁に使用される文字列* は、エンティティの入力ヘルパー内に表示される をエンティティとして定義[573] することができます。エンティティは[537] 必要な場所にカーソルをポイントとして、入力ヘルパーとして挿入することができます。テキスト文字列の値は1つの場所に保管されるため、これは管理のために役にたちます。値を変更する場合、エンティティの定義を変更するだけで行うことができます。

メモ Authentic View 内でマークアップが非表示の場合、空の要素を簡単に見逃す可能性があります。空の要素を見逃すことを回避するため、大きなまたは小さなマークアップをオンに切り替える[530]。

上記のテキストコンテンツの各種類の使用法を試してください。

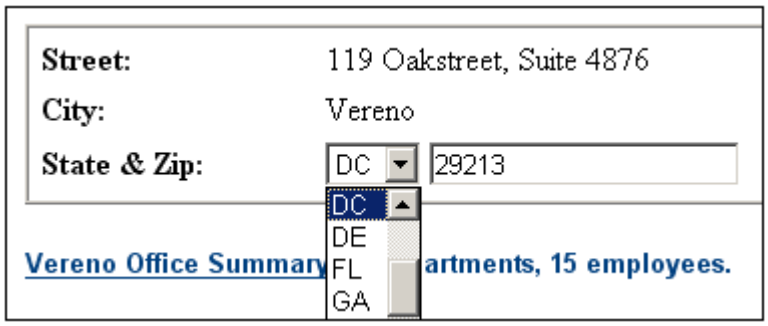
データ入力デバイスを使用してコンテンツを追加する方法

上記で学んだコンテンツの編集内で、テキスト内にコンテンツとして直接入力することにより追加することができます。要素のコンテンツ（または、属性の値）を以下の方法以外で Authentic View 内に入力することはできません。データ入力デバイスを使用して入力する

Authentic View 内のデータ入力デバイスのリストと各デバイスのためのXML ファイルにどのようにデータが入力されるかの説明と共に下に示されています。

データ入力デバイス	XML ファイル内のデータ
入力フィールド (テキストボックス)	ユーザーにより入力されたテキスト
複数の入力フィールド	ユーザーにより入力されたテキスト
コンボボックス	値にマップされているユーザーの選択
チェックボックス	値にマップされているユーザーの選択
ラジオボタン	値にマップされているユーザーの選択
ボタン	値にマップされているユーザーの選択

アドレスフィールドを含む静的なテーブル内で、2つのデータ入力デバイスがあります (*下に表示される*): `Zip` フィールドの入力フィールドと州のフィールドのコンボボックス。テキストフィールド内に入力する値は、対応する要素のXML コンテンツとして直接入力することができます。他のデータ入力デバイスに関しては、選択された内容は値にマップされます。



上に示される Authentic View のための対応するXML テキストは以下のとおりです:

```
<Address>
  <ipo:street>119 Oakstreet, Suite 4876</ipo:street>
  <ipo:city>Vereno</ipo:city>
  <ipo:state>DC</ipo:state>
  <ipo:zip>29213</ipo:zip>
</Address>
```

コンボボックスの選択 `DC` は `DC` の値にマップされていることに注意してください。`Zip` フィールドの値は、`ipo:zip` 要素のコンテンツとして直接入力されます。

11.1.5 属性の値を入力する

属性は要素のプロパティで、要素は無制限の数の属性を持つことができます。属性には値があります。XML データを属性の値として入力する場合があります。Authentic View 内では、属性の値を2つの方法で入力することができます:

- 属性が値を受け入れるように作成されている場合、メインウィンドウ内のコンテンツとして入力する
- 属性入力ヘルパー内で入力する

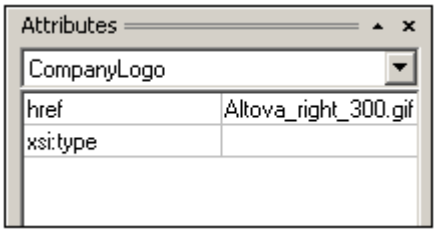
メインウィンドウ内の属性の値

属性の値をノーマルなテキスト、または、入力フィールド内にテキストとして、または、XML 値にマップされるユーザー選択として入力することができます。これらは要素のエンテンツが入力される方法と同様に入力されます: 次を参照してください: Authentic View 内のデータの挿入 535。このような場合、要素のエンテンツと属性の値がStyleVision Power Stylesheet により区別され、データは適切に処理されます。

属性入力ヘルパー内の属性の値

属性の値を入力、または、変更する場合、属性入力ヘルパー内で行うこともできます。最初に、属性ノードをAuthentic View 内で選択すると、属性の値が属性ヘルパー内に入力、または、属性ヘルパー内で編集されます。`NanonullOrg.xml` ドキュメント内では、ロゴの場所は、`CompanyLogo` 要素の`href` 属性の値として保管されます。使用されるロゴを変更するには、以下を行います:

1. CompanyLogo タグをクリックして、CompanyLogo 要素を選択します。CompanyLogo 要素の属性は、属性入力ヘルパー内に表示されます。
2. 属性入力ヘルパー内で、href 属性 をnanonull.gif からAltova_right_300.gif に変更します(`Examples` フォルダー内のイメージ)。



Nanonull ロゴがAltova ロゴにより置き換えられます。

メモ 属性入力ヘルパー内にエンティティを入力することはできません。

11.1.6 エンティティを入力する

Authentic View 内のエンティティは必ずしもそうではありませんが通常、単一のテキスト文字列、およびXML ドキュメントのフラグメントのような XML データです。エンティティはイメージファイルのようなバイナリファイルであることができます。すべてのエンティティは特別のドキュメントのために使用することができ、エンティティの入力ヘルパー内で表示されます(*下のスクリーンショット*)。エンティティを入力するには、ドキュメント内の入力する箇所にカーソルをポイントし、エンティティ内のエンティティをダブルクリックします。属性入力ヘルパー内にエンティティを入力することはできないことにご注意してください。

(アポストロフィ、より大記号 (>)、より小記号 (<) シンボル、および二重引用符と同様)アンパサンド文字 (&) はXML 内で特別な意味を有します。XML 特有の文字と区別するために、これらの文字を挿入するために、エンティティは使用されます。これらの文字はAuthentic View 内のエンティティとして使用することができます。

NanonullOrg.xml 内で、Joe Martin (Marketing) のタイトルを Marketing Manager Europe & Asia に変更してください。以下を行ってください:

1. アンパサンドが入力される箇所にカーソルをポイントしてください。
2. 「amp」としてリストされているエンティティをダブルクリックします。アンパサンドが入力されます (*下のスクリーンショット*)。

メモ エンティティの入力ヘルパーは状況に依存しません。全ての使用することのできるエンティティは、カーソルの位置に関係なく表示されます。これは、ドキュメント内の全ての場所にエンティティを入力できることを意味しません。わからない場合は、エンティティを入力した後、ドキュメントを検証してください:「XML | XML検証」(**F8**)。

自身のエンティティの定義

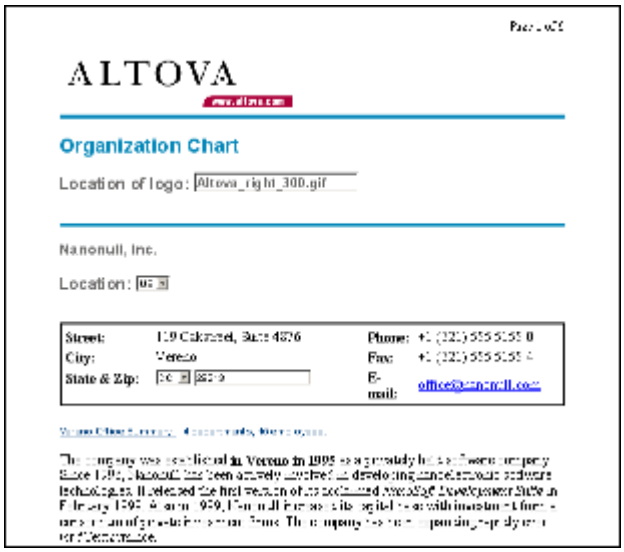
ドキュメントエディターとして、ドキュメントエンティティを定義することができます。この点についてセクション Authentic View 内でエンティティを定義する 573 内で詳しく説明されています。

11.1.7 ドキュメントを印刷する

XML ドキュメントの Authentic View からの印刷は、Authentic View 内の書式設定を保持します。

NanonullOrg.xml を印刷するには以下を行います:

1. まだ行っていない場合は、マークアップモードを非表示にするに切り替えます。マークアップを印刷しない場合は、これを行う必要があります。
2. 「ファイル | 印刷をプレビュー」を選択して、全てのページをプレビューします。50% に縮小された印刷のプレビューページが下に表示されています。ページの書式設定は Authentic View と同じであることに注意してください。



3. ファイルを印刷するには、「ファイル | 印刷」をクリックします。

マークアップを表示するドキュメントのバージョンを印刷することもできます。これを行うには、Authentic View を小さなマークアップモードの表示 、または、大きなマークアップモードの表示に切り替えて印刷します。

11.2 Authentic View インターフェイス

Authentic View はアクティブなドキュメントのAuthentic タブをクリックすると有効化されます。SPS がXML ドキュメントに割り当てられていない場合、割り当てるようにプロンプトされます。「Authentic | Stylevision スタイルシートを割り当てる」コマンドを使用していつでもSPS を割り当てることができます。

このセクションでは以下について説明されています:

- インターフェイスの概要
- Authentic View に特有のツールバーの説明
- メインのAuthentic View ウィンドウで使用することのできるモードの表示に関する説明
- 入力ヘルパーの使用方法の説明
- コンテキストメニューの説明は、XML ドキュメント内のAuthentic View で見つけることができます。

Authentic View 情報の追加ソースは以下のとおりです:

- Authentic View チュートリアルはAuthentic View インターフェイスの使用方法を表示します。このチュートリアルはAltova XMLSpy とAltova Authentic Desktop 製品のドキュメント（チュートリアルのセクションを参照してください）、およびオンラインで見つけることができます。
- Authentic View メニューコマンドの詳細に関しては、次を参照してください: 製品ドキュメントのユーザーレファレンスセクション。

Altova W eb サイト: XML コンテンツの編集、XML の作成

11.2.1 GUI の概要

Authentic View は、ウィンドウの上部のメニューバーとツールバー、および、残りのインターフェイスを占める次ぎの3 つの部分により構成されています: プロジェクトウィンドウ、メインウィンドウ、と入力ヘルパーウィンドウ。これは下に示されるとおりです。

メニューバー

メニューバー内で使用することのできるメニューは、製品ドキュメントのユーザーレファレンスセクション内で詳しく説明されています。

ツールバー

ツールバー内に表示されるシンボルとアイコンは、Authentic View ツールバーアイコン[541]のセクションで説明されています。

プロジェクトウィンドウ

XML、XSL、XML スキーマ、とエンティティファイルをプロジェクト内でグループ化することができます。プロジェクトファイルのリストを作成し変更するには、「**プロジェクト**」メニュー内のコマンドを使用します（この点に関しては、製品ドキュメントのユーザーレファレンスセクションで詳しく説明されています）。プロジェクトファイルのリストはプロジェクトウィンドウ内に表示されています。プロジェクトウィンドウ内のファイルは、ダブルクリックすることによりアクセスすることができます。

情報ウィンドウ

このウィンドウは現在 Authentic View 内で選択されているノードに関する情報を提供します。

メインウィンドウ

XML ドキュメントが表示され編集されるウィンドウです。セクション Authentic View メインウィンドウ[544] 内で詳しく説明されています。

入力ヘルパー

次の8つの入力ヘルパーウィンドウがエリア内に存在します: 要素、属性、および、エンティティ。ウィンドウ（要素と属性 入力ヘルパー）内に表示される内容は状況に依存します。すなわち、ドキュメント カーソルが存在する位置により異なります。要素またはエンティティをドキュメントに入力ヘルパーをダブルクリックすることにより挿入することができます。属性入力ヘルパー内のその属性の値フィールドに属性の値が入力されます。詳細に関しては、次を参照してください: Authentic View 入力ヘルパー[546]

ステータスバー

ステータスバーは現在選択されているノードへのXPath を表示します。

コンテキストメニュー

メインウィンドウ内で右クリックすると表示されるメニューです。使用することのできるコマンドは、状況に依存した編集コマンドです。すなわち、選択されたノードに関連する構造とコンテンツを操作することができます。このような操作は、ノードの挿入、追加、または、削除、エンティティの追加、またはコンテンツの切り取りと貼り付けを含みます。

11.2.2 Authentic View ツールバーアイコン

Authentic View ツールバー内のアイコンは、コマンドのショートカットです。アイコンの一部は、Windows アプリケーション、または、Altova 製品で既に慣れ親しんだアイコンであり、その他は全く新しいアイコンです。このセクションでは、Authentic View の固有のアイコンについて説明されています。下の説明では、アイコンはグループ化されています。

XML マークアップの表示/非表示

Authentic View 内では、XML 要素、または、属性の全部（大きなマークアップ）または一部（小さなマークアップ）を名前と共に、または、名前をなしで表示することができます。4つのマークアップアイコンがツールバー内に表示され、**Authentic** メニュー内で対応するコマンドを使用できるようになります。

	マークアップを非表示にします。すべてのXML タグは既に折りたたまれたものを除いて非表示になります。(展開する通常の方法である) 「マークアップモードを非表示にする」内の折りたたまれたタグをクリックすると、ノードのコンテンツが表示されタグが非表示になります。
	小さなマークアップを表示します。XML 要素/属性 タグは名前と共に表示されます。
	大きなマークアップを表示します。XML 要素/属性 タグは名前無しで表示されます。
	複合型マークアップを表示します。StyleVision Power Stylesheet 内では、各 XML 要素、または、属性を表示のために (大きなマークアップ、または、小さなマークアップ) 指定することができ、また、すべてを非表示にすることも出来ます。要素の一部がマークアップと共に表示、または、マークアップを隠して表示するように指定されているため、複合型マークアップモードと呼ばれます。複合型マークアップモード内では、Authentic View ユーザーはカスタム化されたマークアップを確認することができます。しかしながら、このカスタム化はStyleVision Power Stylesheet をデザインした個人により作成され、Authentic View ユーザーにより定義することはできません。

動的なテーブル構造の編集

動的な SPS テーブル 内の行はデータ構造の繰り返しです。各行は単一の要素の発生を表示しています。各行は、結果として、次と同じ XML サブ構造を持ちます。

動的な テーブルの編集コマンドは、動的な SPS テーブルの行を操作します。要素の発生の数と順序を変更することができます。しかしながら、個別の要素の発生のサブ構造の変更を意味するため、動的な SPS テーブルの列が編集されます。

動的な テーブル編集コマンドのためのアイコンはツールバー内に表示され、**Authentic** メニュー内でも使用できます。

	テーブルに行を追加する
	テーブルに行を挿入する
	現在のテーブル行を複製する (すなわち、セルコンテンツが複製されます)
	現在の行を上に 1 つ移動する
	現在の行を下に 1 つ移動する
	現在の行を削除する

メモ これらのコマンドは、**動的な SPS テーブル** のみに適用されます。静的な SPS テーブル内で使用することはできません。Authentic View 内で使用されるテーブルの異なる種類は、このドキュメントの Authentic View 内のテーブルの使用 557 セクション内で説明されています。

XML テーブルの作成と編集

データをテーブルとして表示するために、自身のテーブルを挿入することができます。このようなテーブルは、XML テーブルとして挿入されます。XML テーブルと書式 テーブルの構造を変更することができます。XML テーブルの作成と編集のためのアイコンは、下に表示されるようにツールバー内で使用することができます。セクション XML テーブル編集アイコン[562] で詳細が説明されています。

アイコンに対応するコマンドは、**メニューアイテムとしては使用することはできません**。XML テーブルを使用するには、この機能は有効化され、StyleVision Power Stylesheet 内で適切に構成されている必要があります。

Authentic View 内で使用されているテーブルの型とXML テーブルの作成方法、及び、編集方法は Authentic View 内のテーブルの使用 [557] 内で説明されています。

テキストのフォーマット アイコン

Authentic View 内のテキストは、XML 要素を適用して、または、必要とされる書式設定を持つ属性を適用することにより書式設定することができます。このような書式設定が定義されると、StyleVision Power Stylesheet のデザイナーは、書式設定を適用するために Authentic View ツールバー内でアイコンを提供します。テキストのフォーマット アイコンを使用してテキストのフォーマットを適用するには、書式設定するテキストをハイライトして、適切なアイコンをクリックします。

DB 行 のナビゲーションアイコン

矢印アイコンは(左側から右側) 最初のレコードに移動する、前のレコードに移動する、レコード # ダイアログに移動する、次のレコードに移動する、最後のレコードに移動する、です。

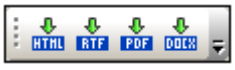
	このアイコンはクエリを入力することのできる「データベースクエリの変更」ダイアログを開きます。 Authentic View はクエリされたレコードを表示します。

XML データベースの編集

「**編集のための XML データを使用して新規の行を選択する**」コマンドにより、IBM DB2 などのXML DB 内の関連するテーブルから新規の行を選択することができます。この行は、Authentic View 内で表示され、編集することができ、DB に戻し保存することができます。

ポータブルな XML 書式 (PXF) ツールバーボタン

XMLSpy とAuthentic Desktop のAuthentic View で次のPXF ツールバーボタンを使用することができます:



個別のボタンをクリックすると、HTML、RTF、PDF、および/またはDocX 出力が生成されます。

これらのボタンはAuthentic View 内でPXF ファイルが開かれるとこれらのボタンを使用することができます。特定の出力書式のために PXF ファイルがXSLT スタイルシートを含むように構成されている場合、個別のボタンは有効化されます。例えば、PXF ファイルがXSLT スタイルシートをHTML とRTF のために含むように構成されている場合、HTML とRTF 出力のためのツールバーボタンがのみが有効化され、PDF とDocX (W ord 2007+) 出力のためのボタンは無効化されます。

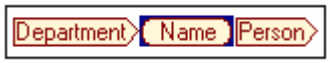
11.2.3 Authentic View メインウィンドウ

Authentic View には4つの表示モードが存在します: 大きなマークアップ、小さなマークアップ、複合型マークアップ、すべての マークアップを隠す。マークアップ 情報の多種のレベルをもつドキュメントをこれらのモードで確認することができます。モード間を切り替えるには、**Authentic** メニュー内のコマンド、または、ツールバー内のアイコンを使用します(前のセクションを参照してください: Authentic View ツールバーアイコン[541])。

大きなマークアップ

タグ内に要素/属性名を持つ要素と属性の開始と終了タグを表示します:

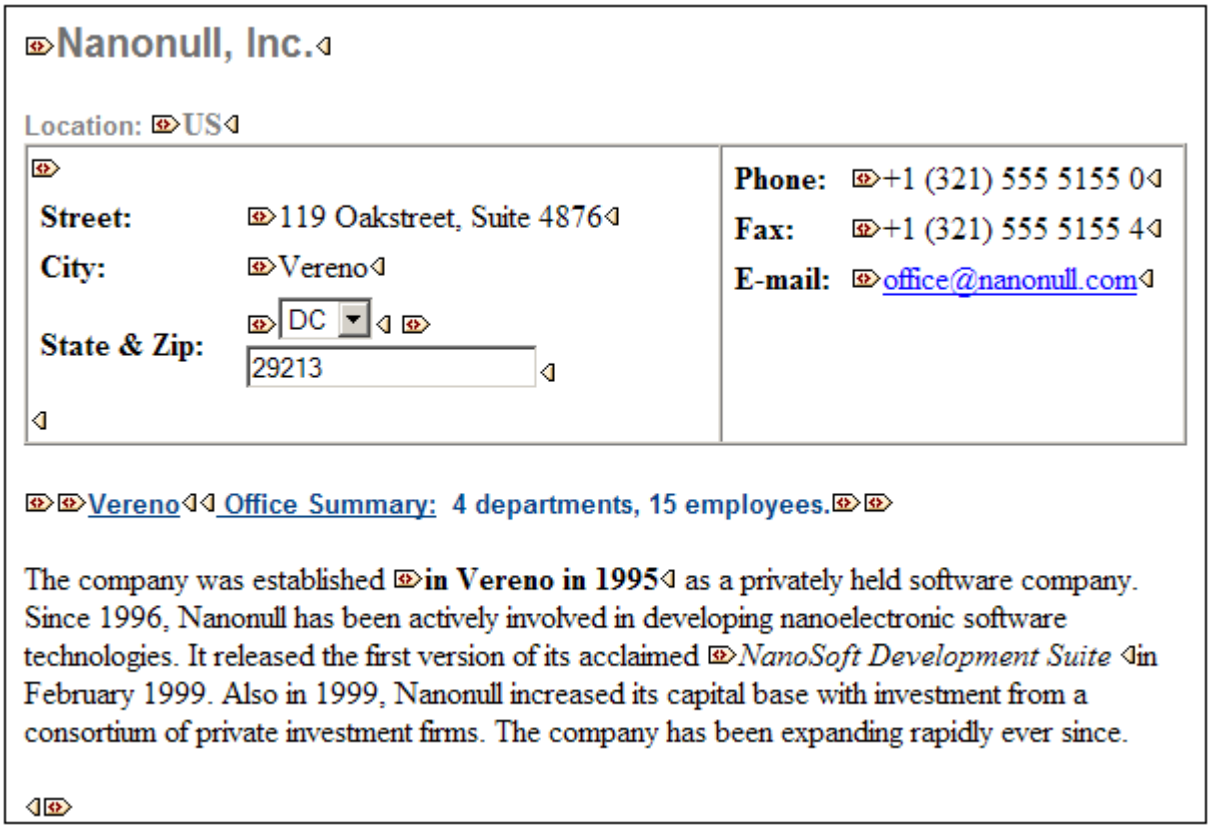
上の図の中では、要素 Name が**展開されており**、すなわち、開始と終了タグ、および要素のコンテンツが表示されています。要素/属性は、開始、または、終了タグをダブルクリックすることにより、**折りたたむことができます**。短くされた要素/属性を展開するには、短くされた タグをダブルクリックします。



大きなマークアップ内では、属性は属性の開始と終了タグ内で等号シンボルとして認識されます:

小さなマークアップ

名前の無い要素/属性の開始と終了タグを表示します :



開始タグ内にはシンボルが存在し、終了タグは空であることに注意してください。また、要素タグには角かっこが使用されており、属性タグには等号サインがシンボルとして使用されていることに注意してください（*下のスクリーンショットを参照してください*）。

要素/属性を展開、または、折りたたむには、適切なタグをダブルクリックしてください。下のサンプルは、（青色でハイライトされている）折りたたまれた要素を表示しています。折りたたまれた要素と展開された要素の開始タグの形に注意してください。

複合型マークアップ

複合型マークアップは、カスタム化されたマークアップのレベルを表示しています。StyleVision Power Stylesheet をデザインした個人はドキュメント内の個々の要素/属性のために大きなマークアップ、小さなマークアップ、または、マークアップ無しを決定することができます。Authentic View ユーザーは、内でカスタム化されたマークアップを複合型マークアップ表示モードで確認することができます。

すべての マークアップを非表示にする

すべてのXML マークアップは各されています。Authentic View 内で確認することのできる書式設定は、印刷されたドキュメントの書式であるため、この表示モードはドキュメントのW YSIW YG 表示です。

コンテンツの表示

Authentic View 内では、コンテンツは2つの方法で表示されています:

- テキスト形式。テキストを入力し、このテキストが要素のコンテンツ、または、属性の値になります。

- データ入力デバイス。表示は、入力フィールド（テキストボックス）、複数の入力フィールド、コンボボックス、チェックボックス、または、ラジオボタンのいずれかを含んでいます。入力フィールドと複数の入力フィールドの場合、フィールドに入力するテキストが要素の XML コンテンツ、または、属性の値になります。

データ入力デバイスの場合、選択は StyleVision Power Stylesheet 内で指定される対応する XML 値を生成します。

このため、コンボボックス内では、（コンボボックスのドロップダウンリスト内で使用することができる）「許可された選択」選択は「1」の XML 値、または、「許可された選択」、またはその他をマップすることができます。「許可されていない選択」は「0」、「許可されていない選択」、または、その他にマップすることができます。

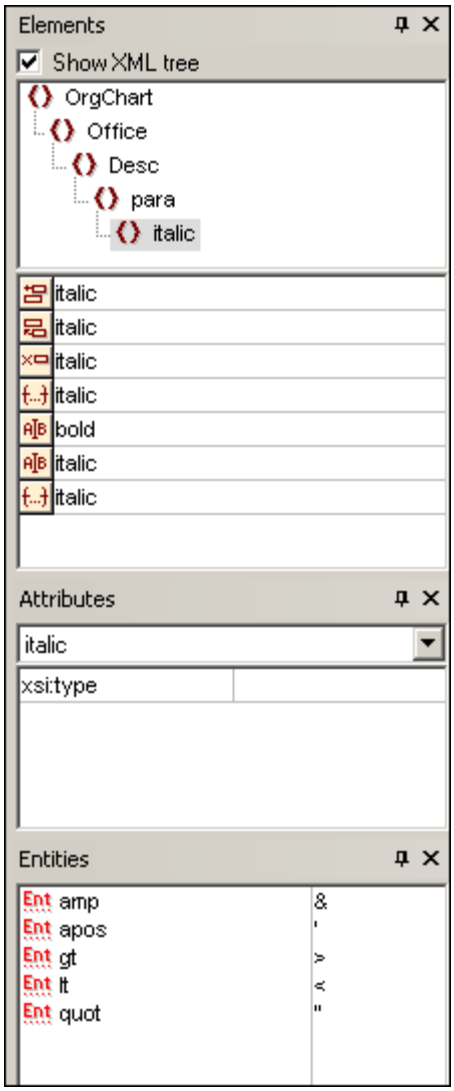
任意のノード

（参照されたスキーマに従い）要素、または、属性が **任意** の場合、型 `add [element/attribute]` のプロンプトが表示されます:

プロンプトをクリックすると、要素が追加され、カーソルがデータの入力のために表示されます。複数の任意のノードが存在する場合、プロンプト `add...` が表示されます。プロンプトをクリックすると、任意のノードのメニューが表示されます。

11.2.4 Authentic View エントリヘルパー

Authentic View 内には3つの入力ヘルパーが存在します: 要素、属性、および、エンティティ。Authentic View インターフェイスの右下のウィンドウとしてこれらは表示されています（*下のスクリーンショットを参照してください*）。



要素と属性 入力ヘルパーは状況に依存しています。すなわち、入力ヘルパー内に表示される内容は、ドキュメント内のカーソルの位置により異なります。エンティティの入力ヘルパー内に表示されているエンティティは、状況に依存しません。ドキュメントのために許可されている全てのエンティティはカーソルの場所に関わらず表示されます。

それぞれの入力ヘルパーの説明に関しては下で説明されています。

要素入力ヘルパー

要素入力ヘルパーは2つの部分から構成されています:

- 上の部分には、「XML ツリーを表示」チェックボックスを使用してオンとオフを切り替えることのできる XML ツリーが含まれています。XML ツリーは、現在の要素のドキュメントのルート要素までの祖先を表示しています。XML ツリー内の要素をクリックすると、その要素に対応する要素（このリストの次のアイテム内で説明されているとおり）は要素入力ヘルパーの下の部分に表示されます。
- 挿入することのできるノードのリストを含む下の部分は、Authentic View 内の選択された要素、または、テキストの範囲から挿入、削除またはクリアすることができます。入力ヘルパー内の要素名の左側のアイコンにより入力ヘルパー内にリストされる要素が表示されます。要素入力ヘルパー内に発生するアイコンは意味を説明する詳細と共に下にリストされます。

入力ヘルパーからノードを使用するには、アイコンをクリックします。

要素の後に挿入
入力ヘルパー内の要素は、選択された要素の後に挿入されます。正確な階層レベルに追加されている場合、. 例えば、 //sect1/para 要素内にカーソルが存在する場合、sect1 要素を追加すると、新規のsect1 要素が//sect1/para の兄弟の後ではなく、para 要素の親であるsect1 要素の兄弟の後に追加されます。

要素の前に挿入
入力ヘルパー内の要素は選択された要素の前に挿入されます。要素の後に挿入 コマンドと同様、要素は正確な階層レベルとして挿入されます。

要素の削除
要素とコンテンツを削除します。

要素の挿入
入力ヘルパーからの要素を要素内に挿入することもできます。要素内にカーソルが置かれると、その要素の許可されている子要素を挿入することができます。許可されている子要素は要素のみのコンテンツモデルの一部コンテンツモデル、および、複合型 コンテンツモデル(テキストと子要素)となることができます。

テキストの範囲が選択されていると、許可されている子要素がテキスト内で入力ポイントとしてカーソルがポイントされている場所に挿入されます。

- テキストの範囲が選択され、要素が挿入されると、テキストの範囲は挿入された要素のコンテンツになります。
- 要素が入力ポイントに挿入されると、要素はそのポイントに挿入されます。

要素の挿入後、これらのインラインの要素のための表示される(要素入力ヘルパー内の) 要素アイコンをクリックすることによりクリアすることができます。テキストの範囲、または、入力ポイントとしてテキスト内にカーソルをポイントするかにより表示されるアイコンが決定されます(下を参照してください)。

要素の適用
(大きなマークアップの表示 ビュー内の開始、および終了タグをクリックすることにより)ドキュメント内の要素を選択する場合、および、要素が他の要素により置き換えられる場合 (例えば、 para などの複合型 コンテンツ 要素内では、italic 要素 がbold 要素と置き換えられることができます)、このアイコンは入力ヘルパー内の要素が選択された(オリジナルの) 要素に適用することができることを示します。**要素の適用** コマンドを複合型 コンテンツの要素内のテキストの範囲に適用することもできます。テキストの範囲は適用された要素のコンテンツとして作成されます。

- 適用された要素が元の要素の子と**同じ名前を持つ子要素** を持つ場合、また、元の要素内でこの子要素のインスタンスが存在する場合、オリジナルの子要素が新規の要素のコンテンツ内で保持されます。
- 適用された要素が元の要素のインスタント化された子と**同じ名前を持たない子要素** を持つ場合、元の要素のインスタント化された子が新規の要素が持つ子要素、または、要素の兄弟として追加されます。
- 適用された要素が元の要素のコンテンツモデル内に**等価が存在しない子要素**を持つ場合、この子要素は直接作成されず、Authentic View が挿入するオプションを提供します。

要素ではなくテキストの範囲が選択されている場合、要素の選択への適用は、コンテンツとして選択されたテキストの範囲と共にその場所で適用された要素を作成します。カーソルが入力ポイントの場合要素の適用は許可されていません。

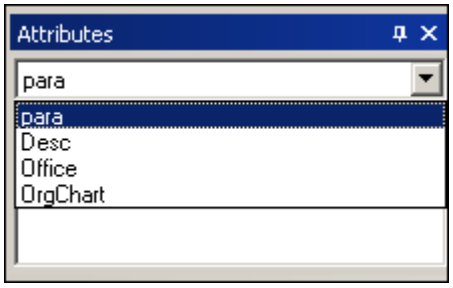
要素をクリアする
このアイコンは、複合型コンテンツの要素内のテキストが選択されると表示されます。アイコンをクリックすると選択されたテキストの範囲の周りから要素をクリアします。

要素をクリアする(挿入ポイントが選択されている場合)
複合型の子-コンテンツ 要素である要素内にカーソルが置かれるとこのアイコンが表示されます。アイコンをクリックするとインラインの要素がクリアされます。

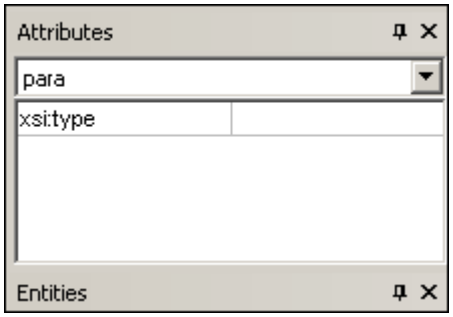
属性入力ヘルパー

属性入力ヘルパーはドロップダウンコンボボックスと属性のリストから構成されています。選択した要素がコンボボックス内に表示されます（開始、または、終了タグをクリック、または、要素コンテンツ内にカーソルをポイントして選択します）。

下に示されている属性入力ヘルパーにはコンボボックス内の `para` 要素が表示されています。コンボボックス内の矢印をクリックすると、この場合 `OrgChart` である、`para` 要素の**祖先ドキュメントのルート要素までの祖先**を表示するドロップダウンリストが表示されます。



コンボボックスの下に、要素のための有効化属性のリストが表示されます。この場合、`para` のためのリストが表示されます。要素で属性が必須の場合、太字で表示されます。（下のサンプルでは、内蔵の属性 `xsi:type` 以外の必須の属性は存在しません）。



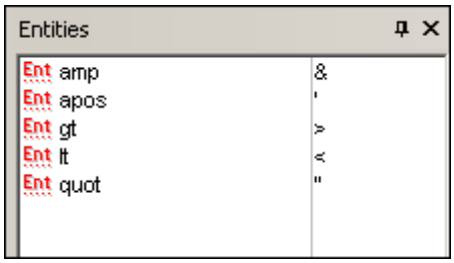
属性のために値を入力するには、属性の値フィールドをクリックして、値を入力します。これにより属性とその値がXMLドキュメント内に作成されます。

以下に注意してください：

- nillable 要素が選択された場合属性入力ヘルパー内に表示される `xsi:nil` 属性の場合、`xsi:nil` 属性の値は属性の値のためのドロップダウンリストから許可されている値（`true`、または、`false`）の1つを選択して入力することができます。
- `xsi:type` 属性を属性の値フィールド内をクリックして変更することができ、ドロップダウンリストからリストされた値を選択することができます。リストされた値は、Authentic View ドキュメントがベースとされるXMLスキーマ内で定義された使用することのできる抽象型です。

エンティティの入力ヘルパー

エンティティの入力ヘルパーによりドキュメントにエンティティを挿入することができます。エンティティを使用してドキュメントに（企業名など）頻繁に発生する特殊文字、または、テキストフラグメントを挿入することができます。エンティティを挿入するには、エンティティを挿入するテキストにカーソルを置き、エンティティの入力ヘルパー内のエンティティをダブルクリックします。



メモ 内部のエンティティはDTD 内で定義された値を持ちます。外部エンティティは他のXML ファイルなどの外部ソース内に含まれる値を持ちます。内部と外部エンティティはエンティティの入力ヘルパー内にリストされています。エンティティを挿入すると、— 値ではなく — 内部の、または、外部のエンティティがXML テキストに挿入されます。エンティティが内部のエンティティの場合、Authentic View **エンティティの値** を表示します。エンティティが外部エンティティの場合、Authentic View は値ではなくエンティティを表示します。これは、Authentic View 内で表示される外部エンティティであるXML ファイルを意味します。コンテンツは Authentic View 表示内のエンティティを置き換えません。

Authentic View 内で自身のエンティティを定義することができ、これらは入力ヘルパー内で表示されます: Authentic View セクション内の編集の[エンティティの定義](#) 573 を参照してください

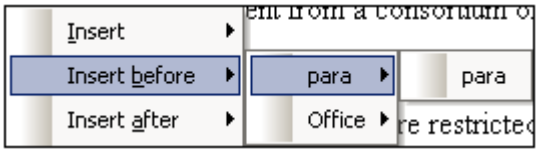
11.2.5 Authentic View コンテキストメニュー

選択されたドキュメントのコンテンツ、選択に関連したコマンドを使用して表示されるコンテキストメニュー、または、カーソルの場所を右クリックします。

要素の挿入

下の図は現在のカーソルの位置で挿入することのできる全ての要素リストである**サブメニューの挿入** を表示しています。**サブメニューの前に挿入** は現在の要素の前に挿入することのできる全ての要素をリストしています。**サブメニューの後に挿入** サブメニューは現在の要素の後に挿入することのできる全ての要素をリストしています。下の図では現在の要素は`para` 要素です。`bold` と`italic` 要素は、現在の`para` 要素内に挿入することができます。

下で確認することができるように、`para` と`Office` 要素は現在の`para` 要素の前に挿入することができます。



ノードの挿入、置換え、(**適用**)、およびマークアップの削除 (**クリア**) コマンドはコンテキストメニュー内、および[Authentic View 入力ヘルパー](#) 546 内で使用することができそれぞれのセクションで説明されています。

エンティティの挿入

「エンティティの挿入」コマンドにカーソルをポイントすると、宣言されている全てのエンティティのリスト含むサブメニューが表示されます。エンティティをクリックすると、選択された場所にエンティティが挿入されます。次を参照してください: ドキュメントのためにエンティティを定義する方法に関しては[エンティティの定義](#) 573 を参照してください。

CDATA セクションの挿入

このコマンドは、カーソルがテキスト内にポイントされると有効化されます。クリックするとカーソルの位置で CDATA セクションが挿入されます。CDATA セクションは開始と終了タグにより区切られています。大きな、小さなマークアップをオンにしてこれらのタグを確認することができます。CDATA セクション内では、XML マークアップと解析は無視されます。XML マークアップ文字(アンパサンド、アポストロフィ、より大記号(>)、より小記号(<)、および引用符) はマークアップとして扱われず、リテラルとして扱われます。ですから、CDATA セクションは、XML マークアップ文字を持つプログラムコードリスティングなどのテキストのために役にたちます。

ノードの削除

「削除」コマンドにマウスのカーソルをポイントすると、選択されたノードにより構成されたメニューリストとドキュメントを無効化しない削除可能な祖先すべてが表示されます。削除する要素をクリックします。これは要素、または、削除可能な祖先を簡単に削除する方法です。祖先要素をクリックすると、選択されて要素を含む全ての子孫が削除されます。

クリア

「クリア」コマンドは要素マークアップを選択の周りからクリアします。ノード全体が選択されると、要素 マークアップはノード全体のためにクリアされます。テキスト セグメントが選択されると、要素 マークアップがそのテキスト セグメントのみのためにクリアされます。

適用

「適用」コマンドはメインウィンドウ内の選択の選択された要素に適用されます。詳細に関しては、次を参照してください: Authentic View 入力ヘルパー 546。

コピー、切り取り、貼り付け

これらは標準ウィンドウコマンドですが「貼り付け」コマンドはコピーされたテキストを SPS 全体のために指定されたスタイルシートのデザイナーにより XML、または、テキストとして貼り付けます。「XML としてコピー」と「テキストとしてコピー」コマンドの詳細に関しては、次を参照してください: 下記の「貼り付け」コマンドの詳細。

貼り付け

「貼り付け」コマンドは Authentic View XML フラグメントにクリップボードにコピーされた内容を) XML、または、テキストとして張り付けるオプションを提供します。コピーされたフラグメントが XML として貼り付けられると、XML マークアップと共に貼り付けられます。テキストとして貼り付けられると、(XML マークアップではなく) コピーされたフラグメントのテキスト コンテンツのみが貼り付けられます。次のシチュエーションが可能です:

- *マークアップタグと共にノード全体* が Authentic View 内でハイライトされ、クリップボードにコピーされます。(i) ノードはこのノードが有効な場所で XML として解析されます。無効な場所では解析されません(ii) ノードがテキストとして解析されると、(マークアップではなく) ノードの テキスト コンテンツのみが貼り付けられます。テキストを貼り付けることのできる XML ドキュメント内にテキスト コンテンツを貼り付けることができます。
- *テキスト フラグメント* が Authentic View 内でハイライトされ、クリップボードにコピーされます。(i) フラグメントが貼り付けられる場所で XML ノードが有効な場合のみ、このフラグメントが XML として貼り付けられる場合、テキストの XML マークアップタグは、テキスト フラグメントと共に明示的にコピーされず、テキストと共に貼り付けられます。(ii) フラグメントがテキストとして貼り付けられる場合、テキストが貼り付けられる XML ドキュメント内の場所に貼り付けられます。

メモ テキストは、テキストが許可されているノードにコピーされます。コピーされたテキストがドキュメントを無効化しないように注意してください。コピーされたテキストは以下のようになります:

(i) 新規の場所で構文的に有効な場合 (例えば、数字のノード内の数字以外の文字 が無効になります)。
(ii) 以外はノードを無効化する場合 (例えば、3 つの桁のみ受け入れるノード内の 4 つの桁はノードを無効化します)。

メモ 解析されたテキストがドキュメントを無効化する場合、テキストが赤い色で表示されます。

削除

「**削除**」コマンドは選択されたノードとそのコンテンツを削除します。ノード内にカーソルをポイント、または、ノードの開始、または、終了タグをクリックすることによりノードは選択されているとして考えられます。

11.3 Authentic View 内で編集する

このセクションはAuthentic View の重要な機能について詳しく説明しています。機能は頻繁に使用されるため、または、メカニズム、または、概念について説明が必要なため、これらの機能はこのセクションに含まれています。

セクションでは以下について説明されています:

- Authentic View 内で使用されているテーブルには3つの種類が存在します。セクション Authentic View 内のテーブルを使用する[557] はテーブルの3つの種類（静的な SPS、動的な SPS、および XML）がいつどのように使用されるかについて説明しています。最初に広義および概念についての説明がされ、使用方法の詳細について説明されます。
- 日付の選択は、日付をクリックすると、正確な XML 書式で日付を入力する視覚的なカレンダーです。次を参照してください: 日付の選択[571]。
- エンティティは、特殊文字、または、テキスト文字列の短縮形です。これらの特殊文字 、または、テキスト文字列を挿入するために、対応するエンティティを挿入することにより、自身のエンティティを定義することができます。詳細に関しては、次を参照してください: エンティティの定義[573]
- Altova 製品のEnterprise とProfessional エディションでは、Authentic View ユーザーは、デジタルXML 署名[574]を使用してXMLドキュメントを署名し、署名を検証することができます。
- Authentic View で表示することのできるイメージ[575] 。

Altova W eb サイト: XML コンテンツの編集、XML の作成

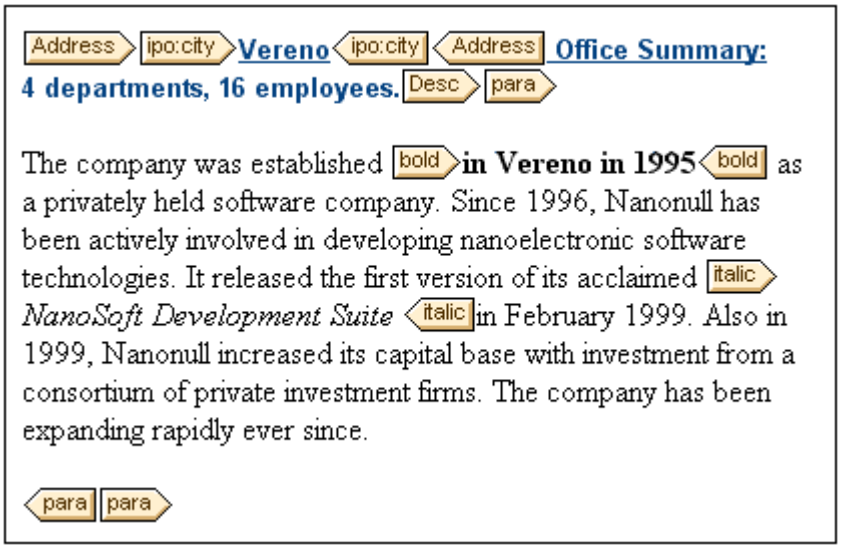
11.3.1 基本的な編集

Authentic View 内で編集を行う場合、XMLドキュメントが編集されます。Authentic View は、ドキュメントの構造的な XML マークアップを非表示にすることができ、ドキュメントのコンテンツのみを表示することができます（下最初のスクリーンショット）。 XML の技術的な側面に触れることなく、ドキュメントをノーマルなテキストドキュメントと同様に編集することができます希望する場合、編集中にマークアップをオンに切り替えることもできます（下の2番目のスクリーンショット）。

Vereno Office Summary: 4 departments, 16 employees.

The company was established **in Vereno in 1995** as a privately held software company. Since 1996, Nanonull has been actively involved in developing nanoelectronic software technologies. It released the first version of its acclaimed *NanoSoft Development Suite* in February 1999. Also in 1999, Nanonull increased its capital base with investment from a consortium of private investment firms. The company has been expanding rapidly ever since.

XML マークアップの存在しない編集することのできる Authentic View ドキュメント

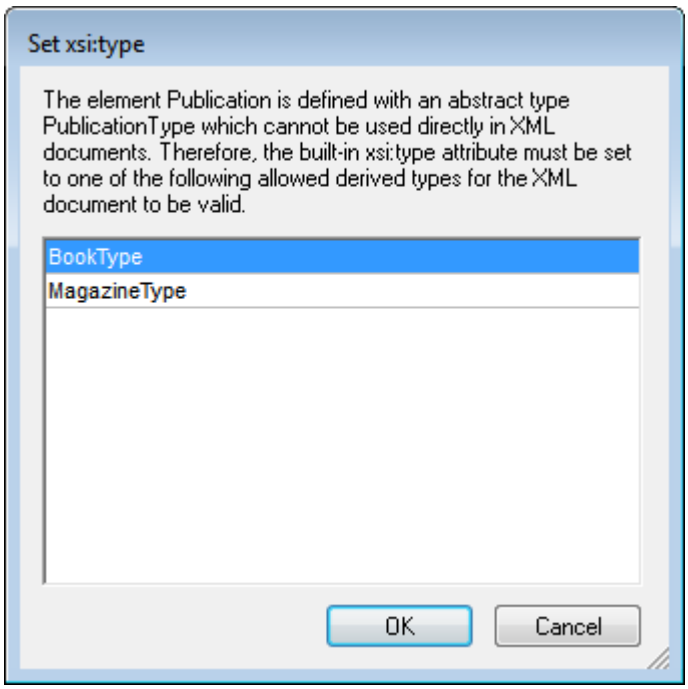


XML マークアップタグが存在する編集することのできる Authentic View ドキュメント

ノードの挿入

新規のノードを Authentic XML ドキュメントに追加する必要がある場合があります。例えば、新規の `Person` 要素はドキュメントのアドレスブックの型に追加される必要があります。このような場合、XML スキーマは新規の要素の追加を許可します。新規のノードを追加する箇所の後に Authentic View ドキュメント内でノードをクリックするだけです。表示されるコンテキストメニュー内で、必要に応じて「**前に挿入**」、または、「**後に挿入**」を選択します。ドキュメントに挿入することのできるノードはサブメニュー内にリストされます。必要とされるノードをクリックして挿入します。全ての必須の子孫ノードも挿入されます。子孫ノードが任意の場合、クリックすることのできるリンク `Add NodeName` が表示され、任意のノードを追加することができます。

追加されるノードが抽象型の要素の場合、(*下のスクリーンショットで表示されるような*) XML スキーマ内で使用することのできる生成された型のリストを含むダイアログが表示されます。



Publication 要素が追加されると、上のスクリーンショットが表示されます。Publication 要素は抽象的な複合型である型 PublicationType です。2つの複合型 BookType と MagazineType は抽象型 PublicationType から派生します。結果として、Publication 要素がXMLドキュメントに追加されると、Publication の抽象型から派生した2つの具体型の1つが指定される必要があります。新規の Publication 要素は xsi:type 属性に追加されます:

```
<Publication xsi:type="BookType"> ... </Publication>
<Publication xsi:type="MagazineType"> ... </Publication>
...
<Publication xsi:type="MagazineType"> ... </Publication>
```

使用することのできる生成された型の1つを選択し、「OK」をクリックすると以下を行うことができます:

- 選択された生成された型を要素の xsi:type 属性の値として設定します。
- 選択された生成された型のコンテンツモデル内で定義された子孫ノードと共に要素を挿入します。

属性入力ヘルパー内の要素の xsi:type 属性の値を変更することにより、選択された生成された型は後で変更することができます。要素の型がこのように変更されると、前の型のコンテンツモデルのノードは、削除され新しいコンテンツモデルのノードが挿入されます。

テキストの編集

Authentic View ドキュメントは、基本的にテキストとイメージにより構成されています。ドキュメント内のテキストを編集するには、テキストを挿入する箇所にカーソルを置き、編集します。(「削除」キーなどの)キーストロークとドラッグアンドドロップ機能を使用して、テキストのコピー、移動、削除を行うことができます。「Enter」キーは唯一の例外です。Authentic View ドキュメントは書式設定済みのため、追加のラインやスペースをアイテムの間に追加することができません。Authentic View 内の「Enter」キーは、このため、現在編集されている要素の他のインスタンスに追加するためにしようされ、この目的のみに使用することができます。

XML またはテキストとしてコピーする

XML、または、テキストとしてコピーし貼り付けることができます。

- テキストがXMLとして解析されると、XML マークアップは、ノードのテキスト コンテンツと共に解析されます。XML マークアップは、ノードのコンテンツの一部がコピーされた場合でも、貼り付けられます。マークアップを貼り付けるには、貼り付ける場所でスキーマに従い、貼り付けが許可されている必要があります。
- テキストがテキストとして解析されると、XML マークアップ破壊席されません。

XML 、または、テキストとして貼り付けるには、テキスト (**Ctrl+C**) をコピーし、テキストを貼り付ける箇所を右クリックし、コンテキストメニューコマンド「貼り付け | XML」、または、「貼り付け | テキスト」. を選択します。ショートカット「Ctrl+V」が使用される場合、テキストは、デフォルトのSPS のデフォルトの貼り付けモードで貼り付けられます。SPS のデザイナーにより貼り付けモードは指定することができます。詳細に関しては、次を参照してください: セクション コンテキストメニュー 550 .

更に、ハイライトされテキストを貼り付ける箇所にドラッグします。テキストがドロップされるとポップアップが表示され、テキストをテキストまたは、XML として貼り付けるかが問われます。希望するオプションを選択します。

テキストのフォーマット

XMLドキュメントのシステムの基本的な概念は、プレゼンテーションとコンテンツは区別されるということです。XMLドキュメントにはコンテンツが含まれますが、スタイルシートにはプレゼンテーション(書式設定)が含まれています。Authentic View 内では、XMLドキュメントはスタイルシートにより表示されています。Authentic View 内で表示されるすべての書式設定はスタイルシートにより作成されていることを意味します。太字のテキストが表示されると、スタイルシートにより太字の書式設定が提供されます。リスト書式、または、テーブル書式を表示するリスト、または、テーブルはスタイルシートにより提供されています。Authentic View 内で編集するXMLドキュメントにはコンテンツのみが含まれており、書式設定は含まれていません。スタイルシートに書式設定が含まれています。Authentic View ユーザーは編集するテキストの書式設定をする必要が無い、または、できないことを意味します。コンテンツのみが編集されます。コンテンツに自動的に適用される書式設定は、データのセマンティクスおよびまたは構造的な値にリンクされています。例えば、(セマンティックな単位として考えることのできる)電子メールアドレスは電子メールであるため、特定の方法で自動的に書式設定されます。同様に、ヘッドラインもドキュメント(構造的な及び構文的

なユニットを含む内の特定の場所で発生する必要があり、スタイルシートデザイナーがそのヘッドラインのために指定する書式に自動的に書式設定されます。電子メールアドレス、または、ヘッドラインの書式設定を変更することができます。電子メールアドレス、または、ヘッドラインのコンテンツを編集するだけです。

一部の場合、コンテンツは特別に表示される必要があります。例えば、テキスト文字列を太字で表示するなど。すべてのこのような場合、プレゼンテーションはドキュメントの構成要素に結び付けられている必要があります。例えば、表示するテキスト文字を太字で表示するなど、太字で表示することのできるスタイルシートデザイナーによるマークアップにより周りのコンテンツから構造的に区別される必要があります。Authentic View ユーザーとして、このようなテキスト文字列を使用する場合テキスト文字列を適切な要素マークアップで囲む必要があります。これを行うための情報に関しては、次を参照してください: ドキュメントの要素入力ヘルパー[547] セクション内の要素の挿入 コマンド

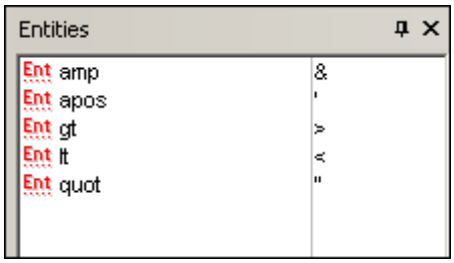
Authentic View 内で RichEdit を使用する

Authentic View 内で、RichEdit ツールバーとして作成された要素の中にカーソルが置かれている場合、ボタンとコントロールは有効化されます(*下のスクリーンショット*)。それ以外の場合、グレーアウトされます。

RichEdit ツールバーのボタンとコントロールを使用して適用するスタイルを指定し、スタイルするテキストを選択します。RichEdit により Authentic View ユーザーはフォント、フォントの大きさ、フォントのスタイル、フォントの飾り、色、テキストの背景色と配置を指定することができます。スタイルされたテキストはスタイル要素のタグに囲まれています。

エンティティの挿入

XML ドキュメント内では、文字の一部はマークアップのために保管されており、ノーマルなテキストでは使用することができません。これらの文字はアンパサンド (&)、アポストロフィ(')、より小記号 (<)、より大記号 (>)、引用符 (") 文字です。これらの文字をデータ内で使用する場合、エンティティの入力ヘルパー[549] を使用して、エンティティレファレンスとして挿入します (*下のスクリーンショット*)。



XML はカスタムのエンティティを作成する機会を与えます。

これらは以下であることができます: (i) キーボードで使用することのできない特殊文字。(ii) ドキュメントのコンテンツ内で再利用するテキスト文字列。(iii) XML データフラグメント、または、(iv) イメージなどの他のソース。Authentic View アプリケーション内でエンティティを自身で定義する[573] ことができます。定義後、これらのエンティティはエンティティの入力ヘルパー[549] 内に表示され、ドキュメントに挿入されます。

CDATA セクションの挿入

CDATA セクションは XML データとして XML パーサーが処理しない XML ドキュメント内のテキストのセクションです。エンティティレファレンスによる特殊文字の置換えを希望しない場合テキストの大きなセクションをエスケープするために使用することができます。例えば、マークアップタグを使用して再生成するプログラムコード、または、XML フラグメントです。要素のコンテンツ内で CDATA セクションは発生することができ、開始と終了で <![CDATA[and]]> を使用してそれぞれ区切ることができます。この結果、テキスト文字列]]> はセクションの終わりを完了前に示すため CDATA セクションは発生しません。この場合、より大きい文字はエンティティレファレンス(>) によりエスケープされるべきです。要素内に CDATA セクションを入力するには、希望する場所にカーソルを置き、右クリックし、コンテキストメニューから CDATA セ

(C) 2015-2021 Altova GmbH

CDATA セクションの挿入を選択します。CDATA セクションタグを Authentic View 内で確認するには、[マークアップの表示オン]に切り替えます[541]。更に、CDATA セクション内に含まれるテキストをハイライトし、**CDATA セクションの挿入**コマンドを選択します。

メモ　CDATA セクションを入力フィールド(すなわち、テキストボックスと複数のテキストボックス)に挿入することはできません。CDATA セクションは、Authentic View にテキストコンテンツコンポーネントとして表示される要素内にのみ入力することができます。

リンクの編集

ハイパーリンクは2つの部分から構成されています: リンクテキストとリンクのターゲット。リンクテキストをテキスト内をクリックして編集することにより編集することができます。リンクのターゲットを編集することはできません。((静的な ターゲット アドレス、または、ターゲット アドレスを XML ドキュメント内に含まれるデータを生成することにより)スタイルシートのデザイナーによりリンクのターゲットが設定されます。) Authentic View からは、「Ctrl」を押してをクリックしてリンクのターゲットに移動することができます。(留意点: リンクをクリックするだけで、リンクテキストを編集するためのセットアップを行うことができます)。

11.3.2　Authentic View 内のテーブル

3つのテーブルの種類は、2つのカテゴリに分けることができます: SPS テーブル(静的と動的) とCALS/HTML テーブル。

SPS テーブルには2つの種対があります: 静的と動的。XML ドキュメントがリンクされている StyleVision Power Stylesheet のデザイナーにより SPS テーブルはデザインされます。XML ドキュメント内に SPS テーブルを挿入することはできず、SPS テーブルフィールドにデータを挿入し、動的な SPS テーブルに行を追加、または、削除することができます。[SPS テーブル][557] のセクションでは、はこれらのテーブルについて説明されています。

CALS/HTML テーブルは Authentic View のユーザーにより挿入されます。目的は、ドキュメント階層構造内で希望する場所にテーブルを挿入することです。[CALS/HTML テーブル][558] と[CALS/HTML テーブル編集アイコン][562] の編集機能に関しては、下で説明されています。

11.3.2.1　SPS テーブル

Authentic View 内で2つの種類の SPS テーブルが使用されます: 静的なテーブルと動的なテーブル。

静的なテーブル

静的なテーブルは構造内およびセルのコンテンツ型内で固定されています。Authentic View のユーザーは、テーブルセルにデータを挿入することができますが、これらのテーブルの構造を変更(すなわち、行や列の追加)すること、セルのコンテンツ型の変更することはできません。テキスト内にタイプしてデータを入力、または、チェックボックス、または、ラジオボタンのフォーム内のオプションから選択して、または、コンボボックスのリストから選択してデータを入力することができます。データを入力すると編集することができます。

Nanonull, Inc.

Street:	119 Oakstreet, Suite 4876	**Phone:**	+1 (321) 555 5155
City:	Vereno	**Fax:**	+1 (321) 555 5155 - 9
State & Zip:	DC 29213	**E-mail:**	office@nanonull.com

メモ　動的なテーブルを編集するためのアイコン、または、コマンドは、静的なテーブルを使用する際に使用することはできません。

動的なテーブル

動的なテーブルには繰り返しデータ構造を表す行が存在します。すなわち、(静的なテーブルの場合は異なりますが)各行は同一のデータ構造を持ちます。結果として、行操作を行うことができます: 行の追加、行の挿入、行を上に移動する、行を下に移動する、行の削除。これらのコマンドは**Authentic** メニューとツール バー内のアイコンとして使用することができます(下に表示される).

これらのコマンドを使用するには、適切な行内にカーソルをポイントし、必要とするコマンドを選択します。

Administration

First	Last	Title	Ext	EMail	Shares	Leave		
						Total	Used	Left
Vernon	**Callaby**	Office Manager	581	v.callaby@nanonull.com	1500	25	4	21
Frank	Further	Accounts Receivable	471	f.further@nanonull.com	0	22	2	20
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com	add Shares	25	7	18
Employees: 3 (20% of Office, 9% of Company)					**Shares: 1500 (13% of Office, 6% of Company)**			
Non-Shareholders: Frank Further, Loby Matise.								

テーブル内のセルを移動するには、「上」、「下」、「右」、「左」矢印キーを使用します。前方の次のセルに移動する場合は、「**タブ**」キーを使用します。最後の行の最後のセル内の「**タブ**」キーを押すと、新規の行が作成されます。

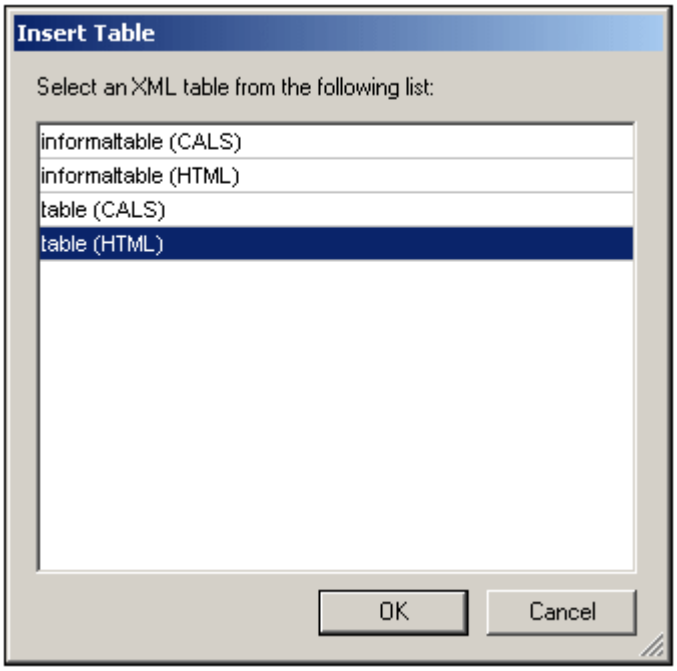
11.3.2.2 CALS/HTML テーブル

一部のXML データ構造がテーブル書式を表示するように指定されているため、Authentic View のユーザーはCALS/HTML テーブルを挿入することができます。CALS/HTML テーブルと作業するには3 つのステップが含まれています: テーブルの挿入。書式設定。データの挿入。CALS/HTML テーブルと作業するためのコマンドはツール バー内のアイコンとして使用することができます(次を参照してください: CALS/HTML テーブル編集アイコン[562])。

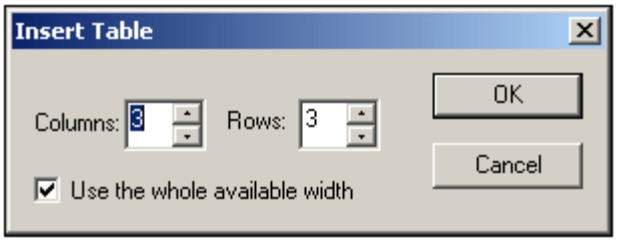
テーブルの挿入

CALS/HTML テーブルを挿入するには、以下を行います:

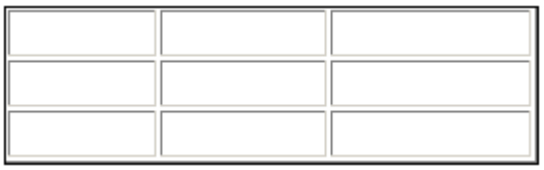
1. テーブルを挿入する箇所にカーソルをポイントし、 アイコンをクリックします。(スキーマによりテーブルを挿入する箇所は決定されることにご注意してください)。テーブルの挿入 ダイアログが表示されます(*下のスクリーンショット*)。このダイアログはテーブル構造が定義されているはすべてのXML 要素 データ構造をリストします。例えば、下のスクリーンショットではinformal テーブル要素とテーブル要素はCALS テーブル、および、HTML テーブルと定義されています。



2. 挿入する要素とテーブルモデルを含むエントリを選択し、「OK」をクリックします。
3. 次のダイアログ内では（*下のスクリーンショット*）列と行の数量を選択し、テーブルに追加されるヘッダーおよびまたはフッターを選択し、使用することのできる幅全てをテーブルが使用するかを指定します。完了すると、「OK」をクリックします。



上に示されるダイアログボックス内の仕様のために、以下のテーブルが作成されました。



テーブル メニューコマンドを使用して、列を追加、または、削除、列を作成、列をジョインし分割することができます。開始するには、広義の構造を作成する必要があります。

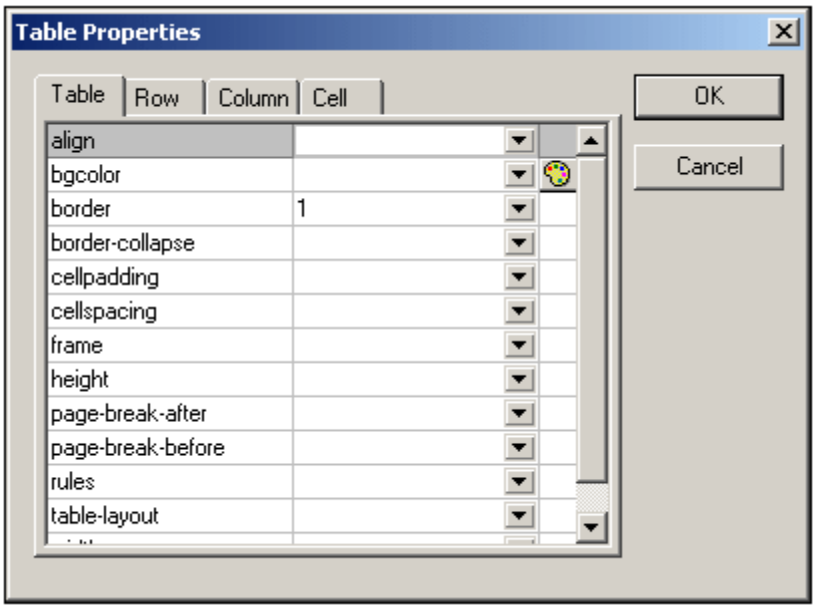
書式設定 テーブルとデータの挿入

テーブル書式設定はドキュメントデザインに既に割り当てられていますが、特定の環境ではテーブル書式設定を変更することができます。これらの環境は、以下のとおりです:

- 多種のテーブル構造要素に対応する要素は属性（基になるXML スキーマ内の）として定義されている適切な CALS、または、HTML テーブルプロパティを持つ必要があります。定義されている属性のみが書式設定のために使用できます。デザイン内で、値がこれらの属性のために設定されている場合、Authentic View 内でこれらの値をオーバーライドすることができます。
- デザイン内では、CSS スタイルを含むスタイル属性を設定してはなりません。CSS スタイルを持つスタイル属性が要素のために指定されている場合、スタイル属性その要素の他の書式設定の属性設定に対して優先順位が与えられます。 この結果、Authentic View 内で指定される書式設定は上書きされます。

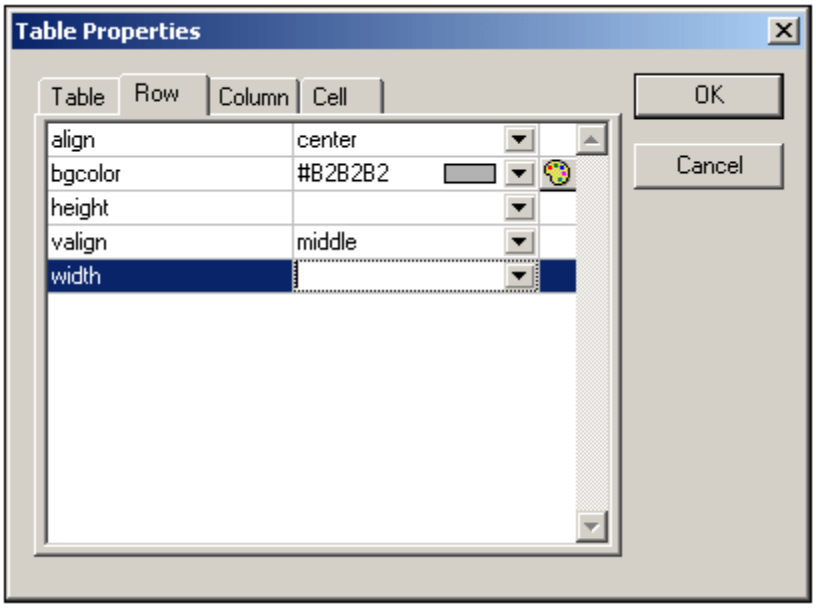
テーブル、行、列、または、セルの書式設定を行うには

1. テーブル （テーブルプロパティ）アイコンにカーソルを置きクリックします。テーブル、または、行、列、または、セルのための書式設定を行うテーブルプロパティダイアログが開かれます（*スクリーンショットを参照してください*）。



2. セルペースとセルパディングプロパティを「0」に設定します。テーブルは以下のようになります:

3. 最初の行にカーソルをポイントし、 （テーブルプロパティ）アイコンをクリックします。「行」タブをクリックします。



最初の行がヘッダー行になるため、この行を他の行と区別するために背景色を設定します。上の図で行プロパティが設定されていることにご注意してください。列ヘッダーテキストを挿入します。テーブルは以下のようになります:

Name	Telephone	Email

配置は指定されているとおり中央に揃えられています。

4. 「Telephone」列をサブ列「Office」と「Home」に分割する場合、「Telephone」列の幅を水平2つの列に分割する必要があります。最初にしかしながら、サブヘッダー行を作成するためにヘッダーセルの垂直に分割します。「Telephone」セル内にカーソルを置き、 (垂直に分割) アイコンをクリックします。テーブルは以下のようになります:

Name	Telephone	Email

5. 「Telephone」を含むセルの下のセルにカーソルを置き、 (水平に分割) アイコンをクリックします。列ヘッダー「Office」と「Home」を入力します。テーブルは以下のようになります:

Name	Telephone		Email
	Office	Home	

「Telephone」列内の各セルの幅を水平に分割します。

テーブル編集アイコンを使用して、列と行を追加、削除、垂直に整列することもできます。 CALS/HTML テーブル編集アイコンは、CALS/HTML テーブル編集アイコン[562] 内で説明されています。

テーブル内のセルを移動する

CALS/HTML テーブル内でセル間を移動するには、「上」、「下」、「右」、「左」矢印キーを使用します。

セルにデータを挿入する

セルにデータを入力するには、セル内にカーソルを置き、データを入力します。

テキストの書式設定

他のテキストXML ドキュメント内同様、CALS/HTML テーブル内のテキストは、 XML 要素、または、属性を使用して書式設定される必要があります。要素を追加するには、テキストをハイライトし、要素入力ヘルパー内の必要とされる要素をダブルクリックします。属性の値を指定するには、テキストフラグメント内にカーソルを置き、属性入力ヘルパー内に必要とされる属性の値 を挿入します。書式設定 ヘッダーテキストを太字に設定すると、テーブルは以下のようになります。

Name	Telephone		Email
	Office	Home	

上のテキストはテキストのハイライトと太字がフォントの書式として指定されているグローバルテンプレートである要素 `strong` のダブルクリックにより書式設定されています。

メモ Authentic View 内で表示されるテキストの書式に関しては、必要とされるテキストのフォーマットを持つグローバルテンプレートが要素のために StyleVision 内で作成される必要があります。

11.3.2.3 CALS/HTML テーブル編集アイコン

CALS/HTML テーブルを編集するために必要なコマンドは、ツールバー内のアイコンとして使用することができ、下にリストされています。これらのアイコンのために対応するメニューコマンドは存在しません。

CALS/HTML テーブルをいつどのように使用するかについての詳細は、CALS/HTML テーブル[558]。

テーブルの挿入

「テーブルの挿入」コマンドは現在のカーソルの場所にCALS/HTML テーブルを挿入します。

テーブルの削除

「テーブルの削除」コマンドは現在アクティブなテーブルを削除します。

行の追加

「行の追加」コマンドは現在アクティブなテーブルの最後に行を追加します。

列の追加

「列の追加」コマンドは現在アクティブなテーブルの最後に列を追加します。

行の挿入

「行の挿入」コマンドは現在アクティブなテーブル内の現在のカーソルの場所の上に行を追加します。

列の追加

「列の追加」コマンドは現在アクティブなテーブル内の現在のカーソルの場所の左に列を挿入します。

セルを左側にジョインする

「セルを左側にジョインする」コマンドは、現在のセル(現在のカーソルの場所)を左側のセルにジョインします。両方のセルのタグは新規のセル内に留まり、列ヘッダーは変更されず、結合されません。

セルを右側にジョインする

「セルを右側にジョインする」コマンドは、現在のセル(現在のカーソルの場所)を右側のセルにジョインします。両方のセルのコンテンツは新規のセルに連結されます。

下のセルにジョインする

「下のセルにジョインする」コマンドは、現在のセル(現在のカーソルの場所)を下のセルにジョインします。両方のセルのコンテンツは新規のセルに連結されます。

上のセルにジョインする

「上のセルにジョインする」コマンドは、現在のセル(現在のカーソルの場所)を上のセルにジョインします。両方のセルのコンテンツは新規のセルに連結されます。

セルを左右に分割する

「セルを左右に分割する」コマンドは、現在アクティブなセルの右に新規のセルを作成します。両方のセルのサイズは元のセルと同じサイズです。

セルを上下に分割する

「セルを上下に分割する」コマンドは、新規のセルを現在アクティブなセルの下に作成します。

上揃え

このコマンドはセルコンテンツをセルの上に揃えます。

上下中央揃え

このコマンドは、セルコンテンツを上下中央に揃えます。

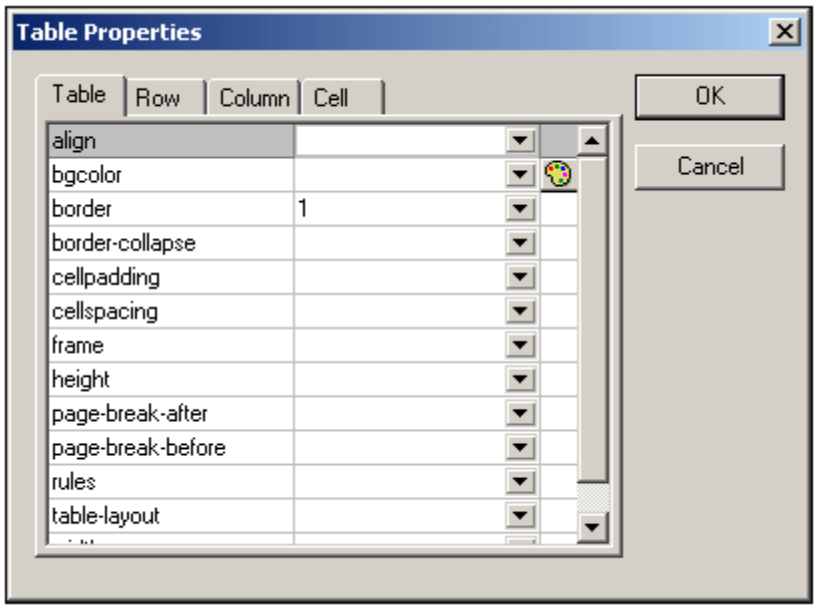
下揃え

このコマンドはセルコンテンツをセルの下に揃えます。

テーブルプロパティ

「テーブルプロパティ」コマンドは、テーブルプロパティダイアログボックスを開きます。このアイコンはHTML テーブルのためにのみ有効化することができますが、CALS テーブルのためにクリックすることはできません。

(C) 2015-2021 Altova GmbH



11.3.3 DB の編集

Authentic View 内で、データベース(DB) テーブルを編集し、データを DB に戻すことができます。このセクションには、DB テーブルを編集する際に使用することのできるインターフェイスの説明が含まれています。次の一般的な点について注意してください:

- デザインをコンパクトにまとめるために、Authentic View 内で表示される DB テーブル内のレコードの数量は StyleVision Power Stylesheet のデザイナーにより意図的に制限されている場合があります。このような場合、Authentic View にロードされるレコードの数量を制限することができます。DB テーブル行 ナビゲーションアイコンを使用して DB テーブル内に他のレコードをロードすることができます(*次を参照:* DB テーブルのナビゲート 565)。
- 特定のレコードを表示するために DB をクエリする 566 ことができます。
- DB レコードを追加、変更、削除し、DB にデータを戻すことができます。次を参照してください: DB テーブルの変更。 570

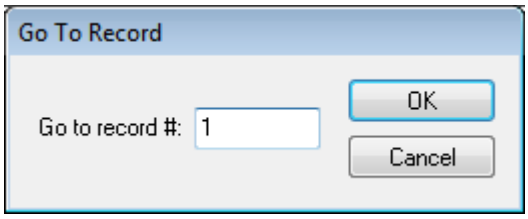
Authentic View 内で DB ベースの StyleVision Power Stylesheet を開くには、「**Authentic | データベースデータの編集**」をクリックして、必要とされる StyleVision Power Stylesheet を参照します。

メモ Authentic View 内では、SQLite データベースからのデータは編集することができません。Authentic View から SQLite データを保存しようと試みると、メッセージボックスにより既知の制限が通知されます。

11.3.3.1 DB テーブル内をナビゲートする

DB テーブル行をナビゲートするコマンドは、Authentic View ドキュメント内でボタンとして使用することができます。通常、4 – 5 つのボタンを持つナビゲーションパネルが各 DB テーブルに伴います。

矢印アイコンは、左側から右側、内の最初のレコードに移動する、前のレコードに移動する、レコードダイアログに移動する、次のレコードに移動する、最後のレコードに移動する、などがあります(*スクリーンショットを参照してください*)。



DB テーブルをナビゲートするには、必要なボタンを押してください。

XML データベース

IBM DB2 などのXML DB の場合、セル(または、行) には、単一のXMLドキュメントが含まれており、このため、単一の行が Authentic View にロードされます。他の行内のXMLドキュメントをロードする場合は、「Authentic | 編集のためにXML データを持つ新規の行を選択する」[1297] メニューコマンドを使用します。

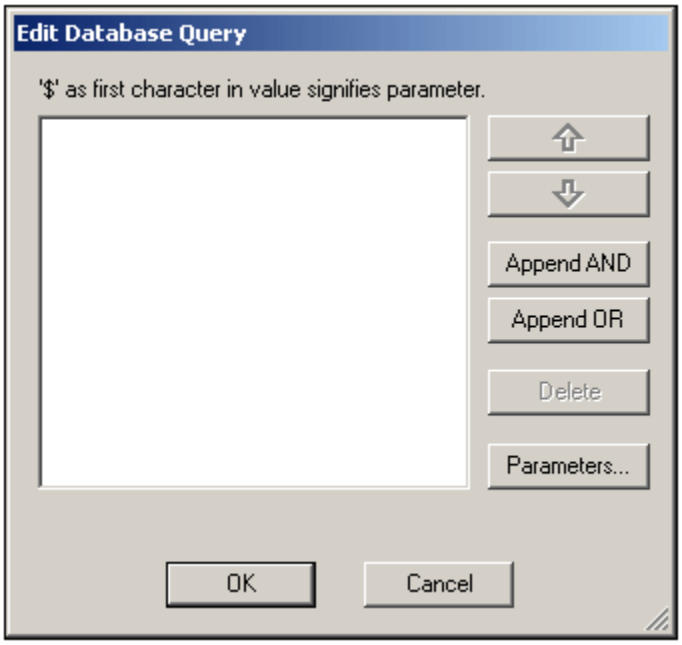
11.3.3.2 DB クエリ

DB クエリにより Authentic View 内に表示されるテーブルのレコードをクエリすることができます。クエリは個別のテーブルのために作成され、1つのテーブルに1つのクエリを作成することができます。クエリの送信時に Authentic View ドキュメント内の変更が保存されない場合、ドキュメント内の全ての変更を保存、または、すべての変更を破棄するかが問われます。他のテーブル内の変更も保存/破棄されます。クエリの送信後、テーブルはクエリの条件を使用して再ロードされます。

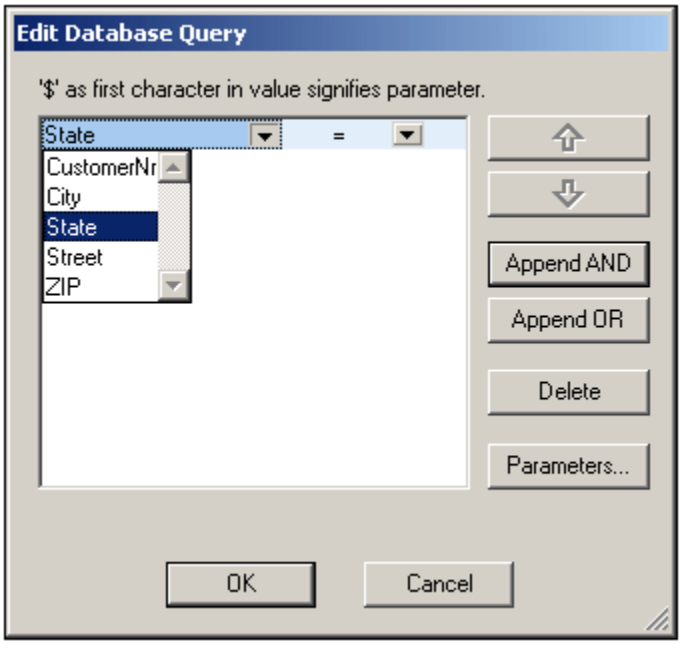
メモ 開かれているテーブルの数量が多すぎることを通知するメッセージが表示されると、開かれているテーブルの数量をクエリを使用してテーブル一部をフィルターし削減することができます。

クエリを作成して送信する方法:

1. データベースクエリの変更 ダイアログを開くために必要とされるテーブルのためのクエリボタン をクリックします(*スクリーンショットを参照してください*)。各 DB テーブルの上、または、下にこのボタンが表示されます。クエリボタンがテーブルに存在しない場合、StyleVision Power Stylesheet のデザイナーはそのテーブルに対して DB クエリ機能を有効化していません。



2. 「AND を追加」、または、「OR を追加」ボタンをクリックします。これにより（下に表示されるように）クエリのための空の条件が追加されます。



3. 条件のために式を入力します。式は以下により構成されています: (i) (関連付けられたコンボボックスで使用することのできる) フィールド名、(ii) (関連付けられたコンボボックスで使用することができます) 演算子、(iii) (直接入力することのできる) 値。式の構築方法に関しては、必要条件内の式 567 セクションを参照してください。
4. 他の必要条件を追加する場合、2つの条件をジョインするために使用するロジカルな演算子 (AND、または、OR) に従い、「AND を追加」、または、「OR を追加」ボタンをクリックします。新規の必要条件を追加します。ロジカルな演算子の詳細に関しては、次を参照してください: セクション DB クエリ内の必要条件を再度並べ替える 568 。

条件内の式

DB クエリの必要条件内の式はフィールド名、演算子、および値により構成されています。**使用することのできるフィールド名** は、選択されたトップレベルのデータテーブルの子要素です。これらのフィールドの名前は、コンボボックス内にリストされます (*上のスクリーンショットを参照してください*)。「**演算子**」は下にリストされています:

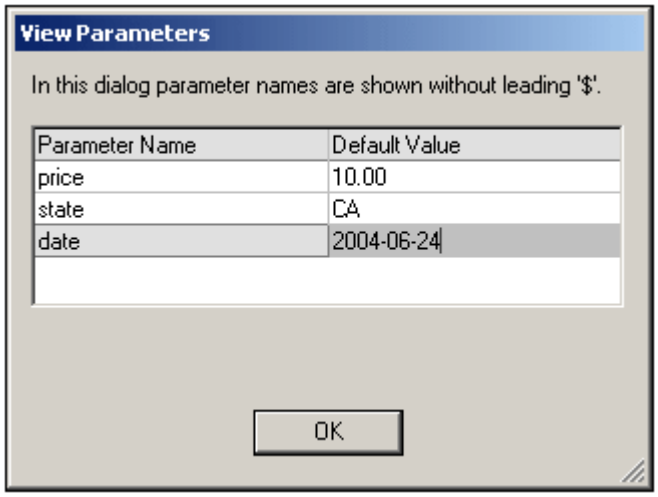
=	等しい記号
<>	等しくない記号
<	より小記号
<=	より小記号、または、等しい記号
>	より大記号
>=	より大記号、または、等しい記号
LIKE	発音が似ている
NOT LIKE	発音が似ていない
IS NULL	空である
NOT NULL	空ではない

IS NULL、または、NOT NULL が選択される場合、値フィールドは無効化されます。**値** は引用符 (または、他の区切り文字) 無しに入力される必要があります。値は対応するDB フィールドと同じ書式設定を持つ必要があります。それ以外の場合、式は `FALSE` を評価します。例えば、MS Access DB 内の `date` データ型のフィールドのための必要条件が式 `StartDate=25/05/2004` を持つ場

合、式は FALSE を結果として評価します。これは、MS Access DB 内の date データ型は書式 YYYY-MM-DD の書式を持つからです。

DB クエリを持つパラメーターの使用

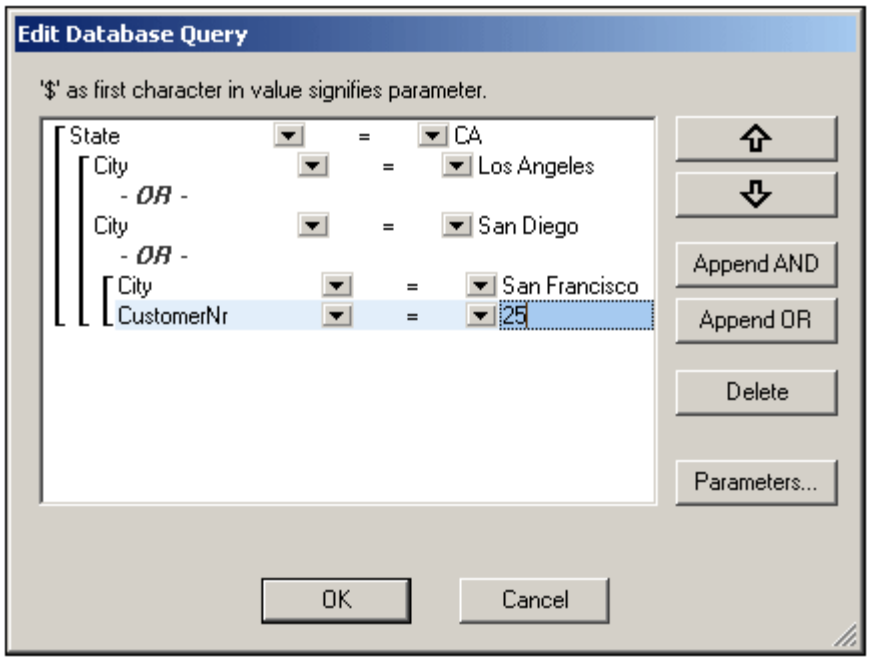
クエリを作成する再、式の値としてパラメーターの名前を入力することができます。パラメーターはクエリ内のリテラルの値の代わりに使用することのできる変数です。式内に入力すると、式内で値が使用されます。使用することのできるパラメーターはSPS 内のSPS デザイナーにより定義されており、パラメーターのビューダイアログ内で確認することができます（*下のスクリーンショットを参照してください*）。（出力ドキュメントがコマンドラインでコンパイルされる場合）コマンドラインからパラメーターに値をパスしオーバーライドすることのできるSPS 内のデフォルトの値にパラメーターが割り当てられます。SPS のために定義されたパラメーターをビューするには、データベースクエリの変更 ダイアログ内のパラメーターボタンをクリックします。これによりパラメーターのビュー ダイアログが開かれます（*スクリーンショットを参照してください*）。



SPS とパラメーター内でスタイルシートのために定義されているすべてのパラメーターを含むパラメーターのビューダイアログはスタイルシートのデザイン内で編集される必要があります。

DB クエリ内の必要条件を再度並べ替える

DB クエリのロジカルな構造と2つの必要条件、または、必要条件のセット間のルーションシップが視覚的に示されます。ロジカルな構造の各レベルは角かっこで示されています。2つの隣接する必要条件、または、必要条件のセットはAND 演算子を示し、2つの必要条件が OR で区切られている場合、演算子が示されます。必要条件はDB クエリのロジカルな構造のクリアな概要を与えることを目的としています。



上のスクリーンショットで表示されているDB クエリはテキストでは以下の用に表されています:

```
State=CA AND (City=Los Angeles OR City=San Diego OR (City=San Francisco AND
CustomerNr=25))
```

必要条件を移動して、または、必要条件のセットをDB クエリ内の他の条件に相対して上下に移動して、DB クエリの順番を変更することができます。必要条件、または、必要条件のセットを移動するには、以下を行います:

1. クリックして必要条件を選択し、そのレベルを示すかっこをクリックして、レベル全体を選択します。
2. ダイアログ内の「上へ」、または、「下へ」矢印ボタンをクリックします。

次の点について注意してください:

- 進行方向内の隣接する必要条件 が同じレベルの場合、2つの条件は場所を交換することができます。
- 必要条件のセット（すなわち、かっこ内の必要条件）は同じレベル内で位置を変更することができますが、レベルは変更しません。
- 同じレベル内で個別の条件は位置を変更します。隣接する必要条件が更に外向き/内向き（すなわち、同じレベルではない）の場合、選択された必要条件は外向き/内向きに1度に1つのレベル移動します。

DB クエリ内の条件を削除するには、条件を選択して「削除」をクリックします。

DB クエリの変更の方法

DB クエリの変更方法:

1. クエリボタン をクリックします。データベースクエリの変更 ダイアログボックスが開かれます。式の条件内の式を編集し、新規の条件を追加し、条件を並べ替え、DB クエリ内の条件を削除することができます。
2. 「OK」をクリックします。DB からのデータは、DB クエリへの変更が反映されるとAuthentic View に自動的に再ロードされます。

11.3.3.3 DB テーブルの変更

レコードを追加する方法

DB テーブルにレコードを追加する方法:

1. (行を追加するために)DB テーブル行 アイコン内にカーソルを置き、または、(行を入力するために) アイコンをクリックします。これにより一時的な XML ファイル内に新規のレコードが作成されます。
2. 「ファイル | 保存する」コマンドをクリックして、DB に新規のレコードを追加します。Authentic View 内では、新規のレコード内の行が DB テーブルに追加されます。このレコードのために AltovaRowStatus が A (追加済み)に設定されます。

新規のレコードのためのデータを入力すると、太字と下線と共に入力されます。これにより、既存のレコードとレコードを区別することができます。既存のレコードがテキストのフォーマット プロパティを使用して書式設定されていない場合、赤字で表示されているデータ型エラーがフラグされます。

「ファイル | 保存する」をクリックして新規のレコードは DB に追加されます。新規のレコードが DB 保存されると、(一により示された) `AltovaRowStatus` フィールドが初期化され、 Authentic View 内で通常のレコードとしてレコードが表示されます。

レコードを変更する方法

レコードを変更するには、必要に応じて DB テーブル内の必要とされる箇所にカーソルを置き、レコードを編集します。表示されるレコードの数量は制限され、必要とされるレコードをナビゲートする必要がある可能性があります(次を参照してください: DB テーブルのナビゲート 565)。

レコードを変更すると、レコードの全てのフィールド内のエントリに下線が引かれ、このレコードのすべての プライマリインスタンスの `AltovaRowStatus` は U (更新済み)に設定されます。`AltovaRowStatus` を持つこのレコードのすべての セカンダリインスタンスは u (小文字) に設定されます。レコードのプライマリとセカンダリインスタンスは DB の構造と生成された XML スキーマに従い定義されます。例えば、Address テーブルが Customer テーブルに含まれる場合、Address テーブルはインスタンス化の 2 つの型内のドキュメントのデザイン内で発生することができます: これは Address テーブル自身が Customer テーブルのインスタンス化内に存在するためです。これらの 2 つの型の片方が変更されると最初に変更される型になります。他の種類としては、1 つ以上の他の型とセカンダリ型が存在する場合があります。データ型エラーは赤い色で表示されフラグされます。

「ファイル | 保存する」をクリックして変更は DB に保存されます。変更されたレコードが DB に保存されると、(一により示された) `AltovaRowStatus` フィールドが初期化され、Authentic View 内で通常のレコードとしてレコードが表示されます。

以下の点に注意してください:

- Authentic View 内のレコードの単一のフィールドが変更され、データが DB に保存されると、レコード全体が更新されます。
- 日付 値 0001-01-01 が一部の DB ために NULL 値として定義されます。エラーメッセージが発生する場合があります。

レコードを削除する方法

レコードの削除の方法:

1. 削除するレコード内にカーソルを置いて、 アイコンをクリックします。削除されるレコードは取り消し線と共にマークされます。AltovaRowStatus は以下のとおりです: レコードのプライマリインスタンスは D に設定され、セカンダリインスタンスは d に設定され、間接的に削除されるレコードは X に設定されます。間接的に削除されるレコードは個別のテーブル内に保管された削除されたレコード内のフィールドです。例えば、Address テーブルが Customer テーブルに含まれる場合があります。Customer レコードが削除される場合、対応する Address レコードが直接削除されます。Customer テーブル内の Address レコードが削除された場合、Customer テーブル名の Address レコードが主に削除されますが、インスタンス化されている場合、独立した Address テーブル内の同じレコードが 2 番目に削除されます。
2. 「ファイル | 保存する」をクリックして変更を DB に保存します。

メモ DB にデータを保存することは、元に戻すコマンドをリセットします。このため、保存前に行われたアクションを元に戻すことはできません。

11.3.4 日付と作業する

Authentic View 内で日付を編集するには2つの方法があります:

- 日付の選択を使用して日付が入力、または、変更されます。
- 値を入力して572 日付が入力、または、変更されます。

Authentic View ユーザーが使用するメソッドはSPS 内で定義されています。両方のメソッドはこのセクションの2つのサブセクションで説明されています。

日付の書式に関するメモ

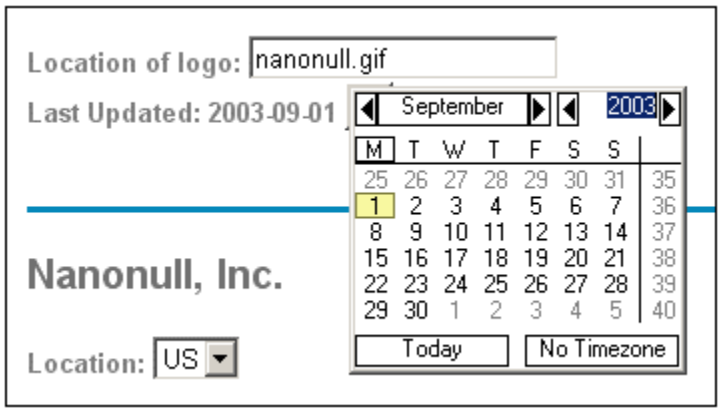
XML ドキュメント内では、日付は複数の日付データ型の1つに保管することができます。これらのデータ型のそれぞれは、XML ドキュメントが有効であるために日付は特定の文法の書式内で保管される必要があります。例えば、`xs:date` データ型は`YYYY-MM-DD` の書式を必要とします。`xs:date` オード内の日付がこの書式以外の書式で入力されると、XML ドキュメントは無効になります。

正確な書式で日付が入力されるように、SPS デザイナーは視覚的な日付の選択をデザイン内に含むことができます。これにより、日付の選択で選択された日付が、正確な文法の書式であることを確認できます。日付の選択が存在しない場合、Authentic View は正確な文法の書式で日付の挿入を行います。XML ドキュメントの検証により必要とされる文法の書式に関するヒントを得ることができます。

11.3.4.1 日付の選択

XML ドキュメントに標準の書式で日付を挿入するために視覚的なカレンダーである日付の選択が使用されます。ドキュメント内のデータの処理のために標準の書式は重要です。日付の選択 アイコンが変更された日付フィールドの近くに表示されます(*スクリーンショットを参照してください*)。

日付の選択を表示するには、日付の選択アイコンをクリックしてください(*スクリーンショットを参照してください*)。



日付を選択するには、希望する日付、付き、または、年度を選択します。XML ドキュメントに日付が挿入され、表示される日付は必要に応じて変更されます。必要な場合は、タイムゾーンも入力してください。

11.3.4.2 テキストのエントリ

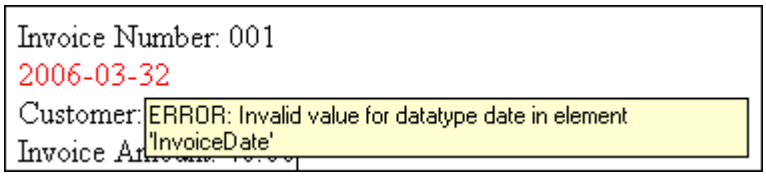
日付の選択が存在しない日付フィールドの場合、新規の値を直接入力することにより日付を編集することができます（*スクリーンショットを参照してください*）。

Invoice Number: 001
2006-03-10
Customer: The ABC Company
Invoice Amount: 40.00

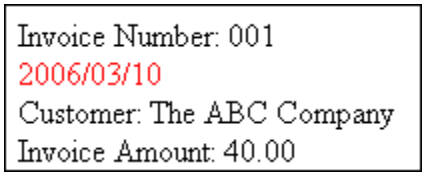
エラー

以下の種類のエラーがフラグされます:

- 日付を編集し、日付が有効な日付の範囲外になった場合、日付は赤い色で表示されエラーを通知します。マウスのカーソルを無効な日付にポイントすると、エラーメッセージが表示されます（*スクリーンショットを参照してください*）。



- 日付の書式を変更しようとすると、日付は赤い色で表示されエラーを通知します（*スクリーンショットを参照してください。ハイフンの代わりにスラッシュが使用されています*）。



11.3.5 エンティティを定義する

エンティティに関して

ドキュメントがDTD、または、XML スキーマをベースにしているかに関わらず、Authentic View 内で使用するために*エンティティ*を定義することができます。定義後、これらのエンティティはエンティティの入力ヘルパー内に表示され、コンテキストメニューの「**エンティティの挿入**」サブメニュー内に表示されます。エンティティの入力ヘルパー内のエンティティをダブルクリックすると、カーソル挿入ポイントにエンティティが挿入されます。

テキスト文字列、XML フラグメント、または、他の外部リソースをドキュメント内の複数の場所で使用する場合、エンティティは役にたちます。必要とされるデータのための短い名前であるエンティティを定義するには、エンティティの定義 ダイアログを使用します。エンティティの定義後、ドキュメント内の複数の場所で使用することができます。これにより時間を節約し、効率よくメンテナンスすることができます。

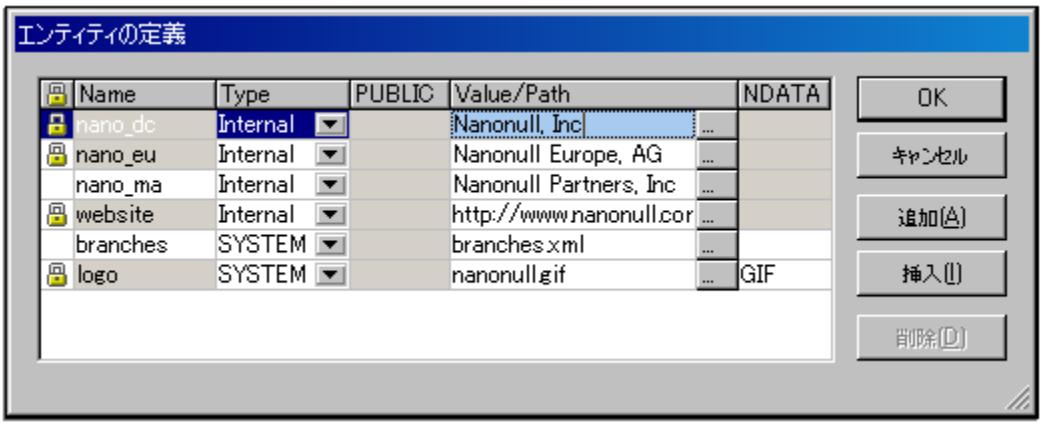
エンティティの型

ドキュメント内で使用することのできるエンティティには2つの広い型が存在します: XML データ（テキスト文字列、または、XML ドキュメントのフラグメント）である「**解析済みのエンティティ**」、または、非-XML データ（グラフィック、サウンド、または、マルチメディア オブジェクトなどの）バイナリファイルである「**解析されていないエンティティ**」。各 エンティティには名前と値が存在します。解析されたエンティティの場合、エンティティはXML データのためのプレースホルダです。エンティティの値 はXML データ 自身、または、データを含む `.xml` ファイルをポイントする URI です。解析されていないエンティティの場合、エンティティの値は非-XML データ ファイルをポイントするURI です。

エンティティの定義

エンティティを定義する方法:

1. 「Authentic | XML **エンティティの定義**....」をクリックすると、エンティティの定義 ダイアログが開かれます（*下のスクリーンショット*）.



2. 名前フィールドにエンティティ名を入力します。これは、エンティティの入力ヘルパーに表示される名前です。
3. エンティティの型を型フィールドのドロップダウンリストから入力します。次の型を使用することができます: テキストが使用する **Internal** エンティティがXMLドキュメント自身内に保管されています。**PUBLIC** または**SYSTEM** の選択はリソースがXMLファイルの外部にあることを指定し、公開識別子またはシステム識別子を使用してロケートされます。システム識別子はリソースの場所を与えるURI です。公開識別子はプロセッサーがリソースを識別するためのロケーションに影響されない識別子です。公開とシステム識別子の両方を指定すると、公開識別子はシステム識別子、システム識別子を使用して解決します。
4. PUBLIC がType として選択されている場合、PUBLIC フィールド内にリソースの公開識別子を入力します。**Internal** 、または、型としてシステムを選択すると、PUBLIC フィールドは無効化されます。
5. Value/Path フィールド内に、次を入力することができます:

- エンティティの型がInternal の場合、エンティティの値として使用するテキスト文字列を入力します。エントリに引用符を入力しないでください。入力する引用符はテキスト文字列の一部として扱われます。
- SYSTEM がエンティティの型である場合、参照ボタンを使用してリソースのURI 、または、ローカルネットワーク上のリソースを入力します。解析されたデータがリソースに含まれている場合、XML ファイルである必要があります（すなわち、ファイルは.xml 拡張子を持つ必要があります）。更に、リソースはGIF ファイルなどのバイナリファイルであることができます。
- エンティティの型がPUBLIC の場合、このフィールドにシステム識別子を追加で入力する必要があります。

6. NDATA エントリはプロセッサーにこのエンティティを解析せず適切なプロセッサーに送信されるように命令します。NDATA フィールドはエンティティが解析されていないエンティティを示す値を含む必要があります。

ダイアログの機能

エンティティの定義 ダイアログ内で以下の機能を使用することができます:

- エンティティの追加
- エンティティの挿入
- エンティティの削除
- 列ヘッダーをクリックしてエンティティをアルファベット順の値に並べ替えます。1度クリックすると昇順に並べ替え、2度クリックすると降順に並べ替えられます。
- ダイアログボックスと列の幅のサイズを調整します。
- ロック。XML ドキュメント内でエンティティが使用されると、このエンティティはロックされ、エンティティの定義 ダイアログ内で編集することはできません。ロックされたエンティティは、最初の列内でロックシンボルにより表示されます。エンティティをロックすることにより、エンティティに対してXML ドキュメントが有効であることを保証することができます（エンティティが参照されており、定義されていない場合は、ドキュメントは無効になります）。
- エンティティの複製はフラッグされます。

エンティティの制限

- 他のエンティティ内に含まれるエンティティはダイアログ Authentic View、またはXSLT 出力内で解決されません。このようなエンティティのアンパサンド文字は & などのエスケープ文字と共に表示されます。
- イメージファイルではない外部の解析されていないエンティティはAuthentic View 内で解決されません。デザイン内のイメージは外部の解析されていないエンティティを読み取るように定義され、URI がエンティティ名（例: `'logo'`）になるように設定されています。エンティティの定義 ダイアログ内でエンティティ名を（上のスクリーンショット内のロゴエンティティのために設定されているように）イメージファイルのURI に対して解決する値を持つ外部の解析されていないエンティティとして定義することができます（*上のスクリーンショットを参照してください*）。

11.3.6 XML 署名

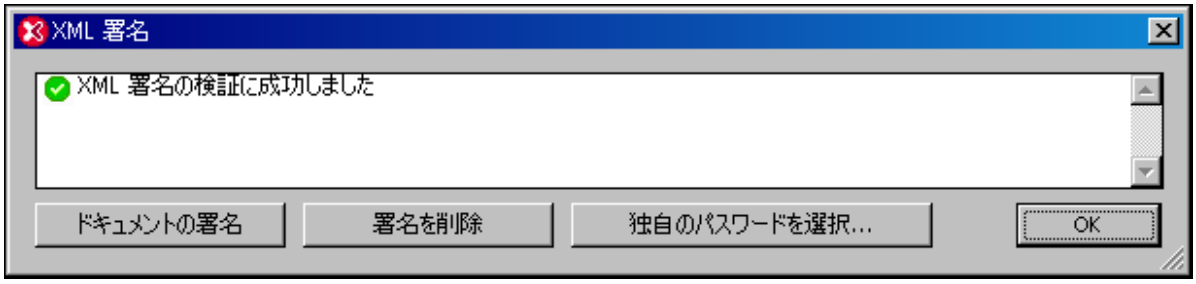
Authentic View のために構成されたXML 署名を使用してSPS をデザインすることができます。XML 署名がSPS 内で有効化された場合、Authentic View ユーザーは有効化された署名を使用してAuthentic XML ファイルを電子的に署名することができます。ドキュメントの署名後、変更を加えると署名の検証は失敗します。署名されたAuthentic XML ドキュメントがAltova 製品内のAuthentic View で開かれると、ドキュメントの検証プロセスが実行され、検証の結果はウィンドウ内に表示されます。

メモ 次のAltova 製品のEnterprise とProfessional のAuthentic View 内でXML 署名を使用し、検証することができます: Authentic Desktop、Authentic Browser、XMLSpy、とStyleVision。

XML 署名アクション

次の署名のためのAuthentic View ユーザーアクションを使用することができます:

- *証明書/パスワードの選択:* 署名は、証明書、または、パスワードで認証されます。署名が作成され、検証される際に認証オブジェクト（証明書、または、パスワード）が必要とされます。Authentic XML ドキュメントが署名が有効化されている SPS を割り当てられているかにより、デフォルトの証明書、または、署名のパスワードを指定する場合があります。デフォルトの証明書、または、パスワードが指定されているかにより、Authentic View ユーザーが自身の証明書とパスワードを選択できるかを決定する署名を構成することができます。XML 署名ダイアログ内でこの操作を行うことができます（*下のスクリーンショット*）。自身の証明書及びパスワードの選択はデフォルトの証明書とパスワードをオーバーライドします。自身の証明書とパスワードはメモリ内に保管され、現在のセッションで使用することができます。自身の証明書とパスワードの選択の後、ファイルまたはアプリケーションを閉じます。SPS は証明書とパスワードのためのデフォルトの設定に戻ります。



- *ドキュメントの署名:* Authentic XML ドキュメントを自動的に、または、手動で署名することができます。自動署名は、SPS デザイナーにより署名の構成内で指定することができ、保存されると Authentic XML ドキュメントが自動的に署名されます。自動署名が有効化されていない場合、ドキュメントを手動で署名することができます。XML 署名 *ツールバーアイコン*）すべての 、または、「**Authentic | XML 署名** コマンドをクリックすると、XML 署名 ダイアログがポップアップされるので、（*上のスクリーンショット*）**ドキュメントの署名** ボタンをクリックします。埋め込まれた署名を使用したドキュメントの署名はスキーマにルート（ドキュメント）要素の最後の子要素として署名要素を許可するように要求します。ドキュメントを署名する場合、認証オブジェクトと署名の配置は署名の構成に従い決定されます。認証情報にアクセスできることを確認してください。詳細に関しては、SPS デザイナーに確認してください。

- *Authentic XML ドキュメントの検証:* SPS のために XML 署名が有効化されていると、検証プロセスは Authentic View XML ドキュメントがロードされる都度、各署名に対して行われます。パスワード、または、証明書 キー情報が SPS と署名と保存されていない場合、Authentic View ユーザーはパスワードを入力、および、検証のための証明書を選択するようにプロンプトされます。XML ファイルと共に保存しない場合、埋め込まれた署名が生成されると、XML ファイルが保存される際に XML ファイルと共に保存されます。（XML 署名ダイアログの「**署名を削除**」ボタンを使用して）生成された署名は明示的に削除される必要があります 同様に、デタッチされた署名が生成されると、必要では ない場合、明示的に削除されます。

11.3.7 Authentic View 内のイメージ

Authentic View を使用すると、最終出力ドキュメント（HTML、RTF、PDF および Word 2007）内で使用することのできるイメージを指定することができます。イメージの書式の一部は一部の書式、または、アプリケーションでサポートされない場合があることに注意してください。例えば、SVG 書式 は PDF 内でサポートされていますが、RTF ではサポートされておらず、HTML 内で表示するためにはブラウザーアドオンが必要になります。イメージ 書式を選択する場合、ドキュメントの出力書式でサポートされている書式を選択してください。
イメージの書式の多数は、多くの出力書式でサポートされています（*下のリストを参照してください*）。

Authentic View は Internet Explorer をベースにしており、Internet Explorer が表示することのできるイメージの書式の大部分を表示することができます。次の一般的なイメージの書式がサポートされています:

- GIF
- JPG
- PNG
- BMP

- WMF (Microsoft ウィンドウメタファイル)
- EMF (拡張されたメタファイル)
- SVG (PDF 出力のためのみ)

相対的なパス

相対的なパスはSPS ファイルに相対して解決されます。

11.3.8 Authentic View 内のキーストロール

Enter キー

Authentic View 内では、「Enter」キーは、特定のカーソルの場所で追加の要素を追加する際に使用されます。例えば、(スキーマに従い)本のチャプターには複数の段落が含まれており、段落のテキスト内で「Enter」を押すと、新規の段落が現在の段落の後に追加されます。チャプターがタイトルと複数の段落を含むことができる場合(タイトル要素を含む)、段落要素外のチャプター内で「Enter」を押すと、(スキーマにより複数のチャプターが許可されていると想定し)現在のチャプターの後に新規のチャプターが追加されます。

メモ 「Enter」キーは新規のラインを入力しません。これは、段落などのテキストノード内にカーソルがある場合でも同じです。

キーボードの使用

入力とナビゲートのためにキーボードを通常の方法で使用することができます:

- 「タブ」キーはカーソルを前に進め、ノードの前後で停止し、ノードコンテンツをハイライトし、静的なコンテンツをステップオーバーし、静的なコンテンツをステップオーバーします。
- `add...` と`add Node` ハイパーリンクは、ノードコンテンツとして考えられ、タブ付けされるとハイライトされます。スペースキーまたは、「Enter」キーを押すと有効化することができます。

11.4 Authentic スクリプト

Authentic スクリプト機能により、SPS デザインに対して更なる柔軟性と双方向性を追加することができます。これらデザインは StyleVision Enterprise ならびに Professional Edition で作成または編集することができ、Altova 製品の Enterprise ならびに Professional Edition に搭載された Authentic ビューにて閲覧することができます。

以下のテーブルでは Altova 製品ごとにサポートされる機能のリストを示します。Authentic Browser Plugin Trusted バージョンはセキュリティー上の観点から内部スクリプトが無効になっていることにご注意してください。

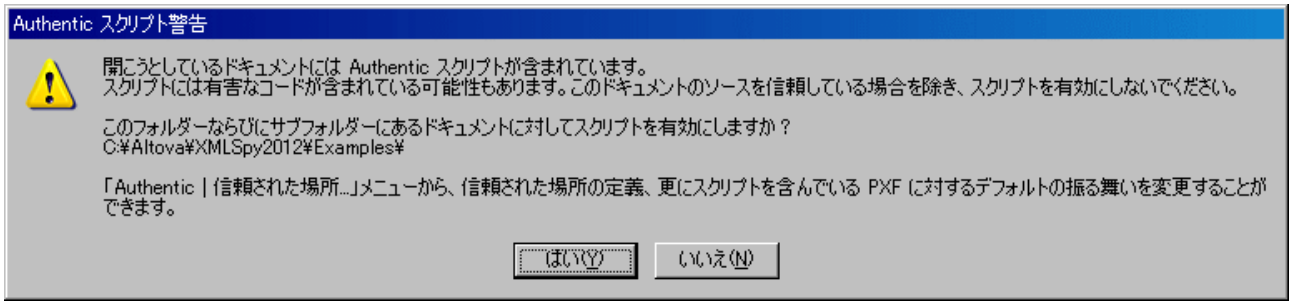
Altova 製品	Authentic スクリプトの作成	Authentic スクリプトの有効化
StyleVision Enterprise	はい	はい
StyleVision Professional	はい	はい
StyleVision Standard *	いいえ	いいえ
XMLSpy Enterprise	いいえ	はい
XMLSpy Professional	いいえ	はい
XMLSpy Standard	いいえ	いいえ
AuthenticDesktop Enterprise	いいえ	はい
AuthenticDesktop Community	いいえ	いいえ
Authentic Browser Plug-in Community **	いいえ	いいえ
Authentic Browser Plug-in Enterprise Trusted ***	いいえ	はい
Authentic Browser Plug-in Enterprise Untrusted	いいえ	はい

* *Authentic ビュー無し*
** *Trusted ならびに Untrusted バージョン*
*** *スクリプトありのデザインは表示可能。内部マクロの実行やイベントハンドリングは無し。外部イベントの実行。*

Authentic スクリプトの動作は全 Altova 製品において同一のものとなり、ある製品に特化したコードや設定は必要ありません。

Authentic スクリプト警告ダイアログ

PXF ファイルまたは SPS へリンクされた XML ファイルにスクリプトが含まれており、そのファイルが開かれた状態で Authentic ビューへの切り替えが行われると、警告ダイアログが表示されます（以下のスクリーンショットを参照）。



以下のオプションから選択を行うことができます:

- **はい**をクリックすると、ファイルを含んでいるフォルダーがAuthentic スクリプトの信頼された場所リストへ追加されます。今後、信頼されたフォルダー内にふくまれるファイルをAuthentic ビューで開いても、警告ダイアログは表示されなくなります。信頼された場所のリストへは「Authentic | 信頼された場所[1303]」メニューコマンドからアクセスならびに編集することができます。
- **いいえ**をクリックすることで、ファイルが含まれているフォルダーを信頼された場所のリストへ追加することなく処理が継続されます。ファイルはスクリプトが無効になった状態で開かれます。ファイルをAuthentic ビューで開くたびにAuthentic スクリプト警告ウィンドウが表示されます。ファイルが含まれているフォルダーを信頼された場所のリストへ追加するには、「Authentic | 信頼された場所[1303]」メニューコマンドから信頼された場所のダイアログを開き、フォルダーを追加ならびに必要に応じて変更することができます。

信頼された場所ダイアログに関する詳細については、ユーザーリファレンスの「Authentic | 信頼された場所[1303]」メニューコマンドのセクションを参照ください。

メモ: COM インターフェースを介したXMLSpy へのアクセスを行う場合、**セキュリティーのチェックは行われず、「Authentic スクリプト警告」ダイアログは表示されません**(COM インターフェースに関する情報はプログラマーのレファレンス[1503] を参照ください)。

Authentic スクリプトの使用方法

SPS デザインを作成したデザイナーは以下にある2つの方法により、Authentic ドキュメントをインタラクティブなものにすることができます:

- スクリプトをユーザー定義のアクション(マクロ)に割り当てることで、要素、ツールバーボタン、そしてコンテキストメニューのアイテムをデザイン
- Authentic ビューイベントに反応するデザインイベントハンドラーを追加

Authentic ドキュメントをインタラクティブなものにするために必要な全てのスクリプトは、StyleVision GUI (Enterprise ならびにProfessional Edition)内で作成することができます。フォーム、マクロ、イベントハンドラーはStyleVision に搭載されたスクリプトエディター内で作成され、これらスクリプトはSPS 内に保存されます。その後、StyleVision のデザインビューにて、保存されたスクリプトは要素、ツールバーボタン、コンテキストメニューに割り当てられることになります。SPS をベースにしたXML ドキュメントがAuthentic スクリプトをサポートするAltova 製品(上のテーブルを参照)により開かれると、ドキュメントには新たな柔軟性や双方向性が追加されます。

Authentic スクリプトのドキュメンテーション

Authentic スクリプトに関する情報はStyleVision のドキュメンテーションでご利用になれます。Altova ウェブサイトの製品ドキュメンテーションページから参照することができます。

12 HTML と CSS

XMLSpy では、HTML[580]、とCSS[582] ドキュメントに対するインテリジェントな編集機能が提供されます。全てのドキュメントはテキストビュー[130]にて編集することができ、アクティブな HTML はブラウザービューにてプレビューを行うことができます。

各ドキュメントの種類に関するインテリジェント編集機能についてはこのセクション以下のサブセクションに記されます：HTML[580]、とCSS[582]。

12.1 HTML

テキストビューではHTMLドキュメントを編集することができ、編集された内容を、すぐにブラウザービューで確認することができます。テキストビューでは以下にあるような数々のHTML 編集機能が提供されています。詳細はテキストビュー[130]のセクションを参照ください。

サポートレベル

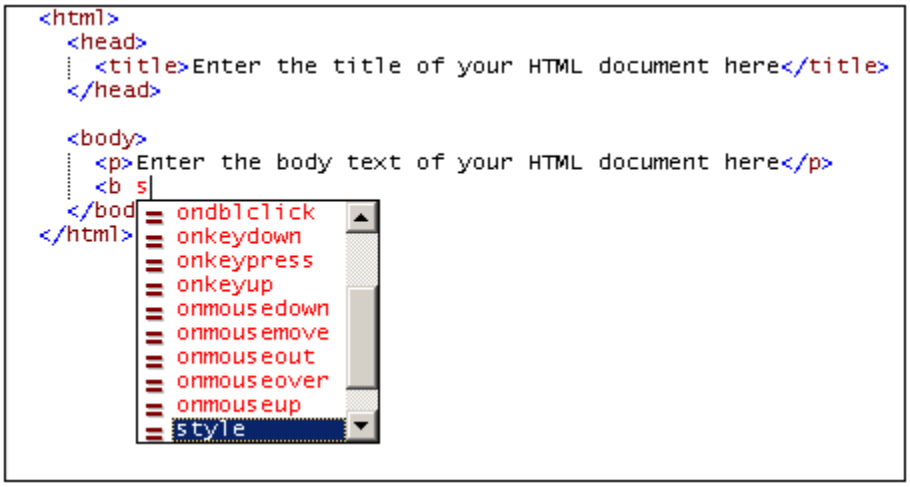
XMLSpy ではHTML 4.0 ならびにHTML 5.0 がサポートされます。入力ヘルパーならびにインテリジェントな編集機能が対応したHTML のバージョンに対して利用可能です。これらの機能を以下に記します。

入力ヘルパー

HTMLドキュメントがアクティブになっていれば、要素、属性、エンティティ入力ヘルペを利用することができます。入力ヘルパーの内容は文脈により変化し、現在カーソルのある位置に入力することができるアイテムが入力ヘルパーには表示されます。HTML 入力ヘルパーはテキストビュー[146]にあるその他の入力ヘルパーと同じように使用することができます。

自動補完

HTMLドキュメントにマークアップを入力する際、XMLSpy は自動補完ヘルプを表示します。ポップアップにはカーソルのある位置に挿入することができるノード全てが表示され、入力がなされるごとに、リスト内にある最も近いマッチの先頭に選択が移動します(スクリーンショットを参照)。選択されたアイテムをクリックすることで、そのアイテムがカーソル位置に挿入されます。



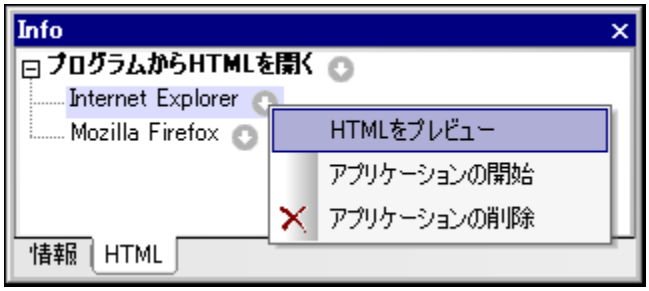
要素に対する自動補完は、ノード開始タグが入力されたときに表示されます。ノードの開始タグがドキュメントに入力されると、その終了タグも自動的に挿入されます。この機能により整形式が保証されます。

ノードの開始タグにて要素名のあとにスペースが入力されると、属性に対する自動補完が表示されます。自動補完ポップアップにある属性名をクリックすると、属性名と引用句が入力され、カーソルが引用句の中間に移動します。

エンティティ入力ヘルパーにはHTML 4.0 、とHTML 5.0エンティティセット Latin-1、特殊な文字、そしてシンボルが含まれています。

HTML 情報ウィンドウ

(以下のスクリーンショットにある)HTML 情報ウィンドウを使えば、アクティブな HTML ファイルに素早くアクセスすることができます。例えば、HTML ファイルがXMLSpy でアクティブであれば、HTML 情報ウィンドウ内にある Mozilla Firefox アイテムをダブルクリックすることで、Mozilla Firefox が起動され、そのHTMLドキュメントがロードされます。



以下の点に注意してください:

- *プログラムから HTML を開く* アイテムの右にあるアイコンを使うことで、*プログラムから HTML を開く* リストにアプリケーションを追加することができます。システムにインストールされているブラウザーまたはその他の(テキストエディターのような)アプリケーションを追加することができます。通常追加されるのはブラウザーまたはエディターアプリケーションです。
- アプリケーションが*プログラムから HTML を開く* リストに追加されたら、(「**インストールされたブラウザーを追加**」コマンドを使用した場合を除いて)アイテムを選択して、「**F2**」を押すことで、リスト内に表示されるアプリケーションの名前を変更することができます。
- リストの各アイテムの右側にあるアイコンを使えば、(i) アプリケーションを開始し、(ii) アプリケーションを開始した後でリンクされた HTML ファイルを開くか、(iii) リストからアプリケーションを削除することができます。アプリケーション名をダブルクリックすることで、リンクされた そのアプリケーションで HTML ファイルが開かれます。
- プログラムから HTML を開く リストに表示されるアプリケーションは、CSS 情報ウィンドウのアプリケーションアイテムと連動しています。

DTD の割り当て

「**DTD/スキーマ**」メニューを使えば、XHTML ドキュメントに対して DTD または XML スキーマを参照、割り当てることができます。XHTML ドキュメントの編集は[XML ドキュメントと同じように](#)[266]行うことができます。

ブラウザービューコマンド

ブラウザービューコマンドは「**ブラウザー**」メニューから使用することができます。

12.2 CSS

テキストビューのインテリジェント編集機能を使用して CSS ドキュメントの編集を行うことができます。CSS の編集に利用することができる編集機能は以下のとおりです。

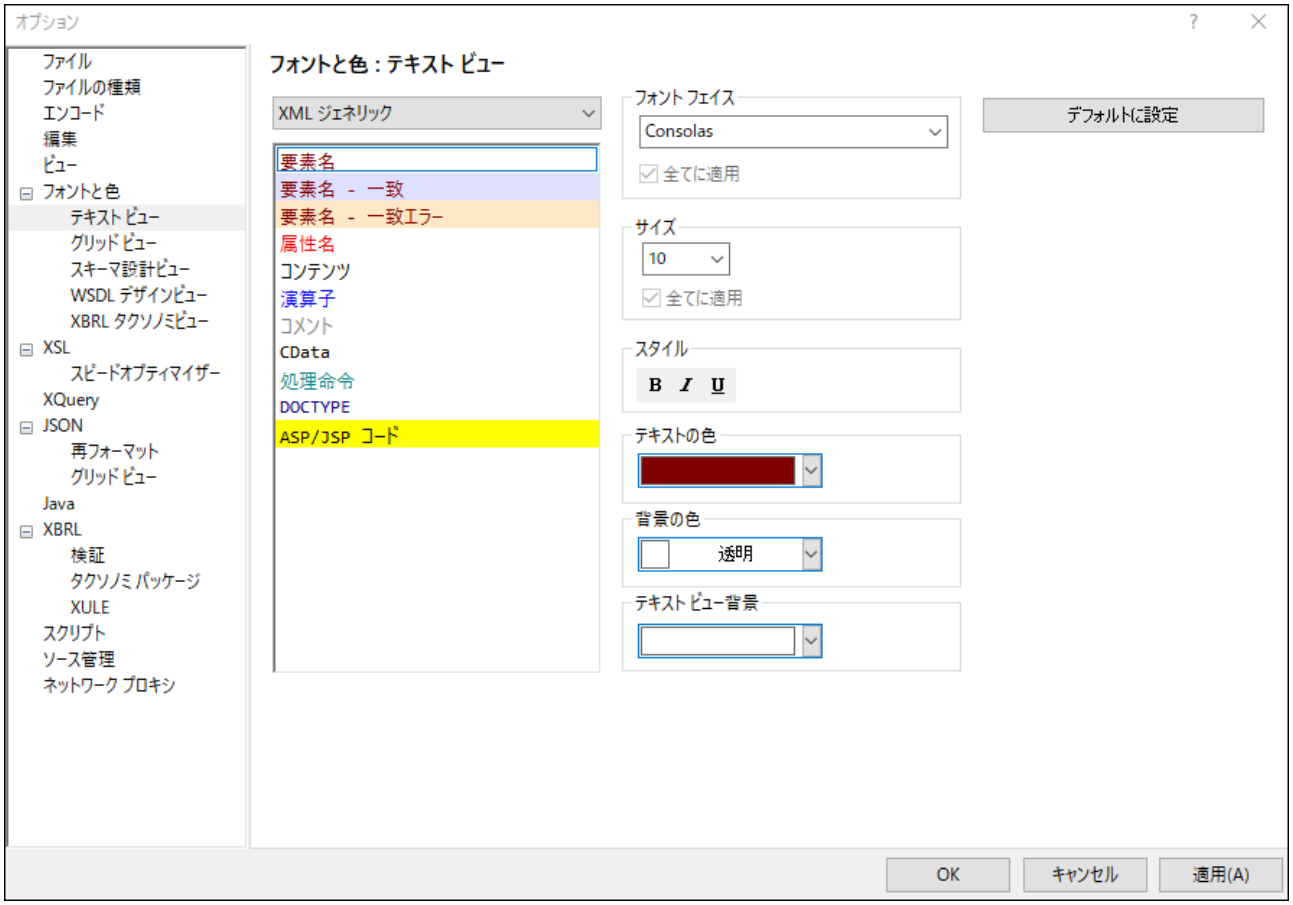
構文色分け

CSS ルールはセレクター、(1つ以上の) プロパティ、そしてそれらプロパティの値により構成されます。これら3つのコンポーネントは、細かな指定を行うため、更に細かなカテゴリに分割されることもあります。例えば、セレクターにより指定されるものにクラス、擬似クラス、ID、要素、または属性があります。更に、CSS ドキュメントは(例えばコメントのような) ルール以外のアイテムも含むことができます。テキストビュー内ではこのようなアイテムのカテゴリはオプションダイアログ内で設定された設定に従い異なる色で表示されます (*下のスクリーンショット*) (*下のスクリーンショット参照*)。

```
.header
{
    font-family: "Arial", sans-serif;
    font-weight: bold;
    color: red;
}
```

オプションダイアログのフォントと色のセクションで多種の CSS コンポーネントの色を設定することができます (*下のスクリーンショット*). (以下のスクリーンショットにあるように) オプションダイアログのテキストフォントタブから、様々な CSS コンポーネントの色を指定することができます。

(C) 2015-2021 Altova GmbH



ソースの折りたたみ

*ソースの折りたたみ*は、*ソースの折りたたみ*余白内で +/- サインにより表示される、各 CSS ルールを展開または折りたたむ機能を指します。余白は[テキストビュー設定ダイアログ](#)1363により切り替えのオンとオフを行うことができます。ルールが折りたたまれると、視覚的に省略記号により表示されます。マウスのカーソルが省略記号にポイントされると、折りたたまれたルールの内容がポップアップ内に表示されます。内容がポップアップに大きすぎる場合、これは、ポップアップの下に省略記号で表示されます。

テキストツールバー内の **全ての折りたたみの切り替え** アイコン はすべての ルールを切り替え、トップレベルのドキュメント要素に対するルールを展開または*折りたたみ*ます。

メモ 中かっこのペアは、ルールを区切り、(*上のスクリーンショット*) 中かっこの前または後ろにカーソルがポイントされると太字で表示されます。これにより特定のルールがどの位置で開始し終了するかの定義が表示されます。

CSS アウトライン

(*以下のスクリーンショット*) CSS アウトライン入力ヘルパーでは、ドキュメント内にあるセレクターが表示されます。CSS アウトラインにあるセレクターをクリックすると、ドキュメント内の対応する箇所がハイライトされます。左下にあるスクリーンショットでは、セレクターの並べ替えは行われず、ドキュメント内で表示される順番通りに表示されています。右下のスクリーンショットでは、(ツールバーアイコンの) アルファベット順の並び替え機能が有効になっており、セレクターがアルファベット順に並べ替えられています。

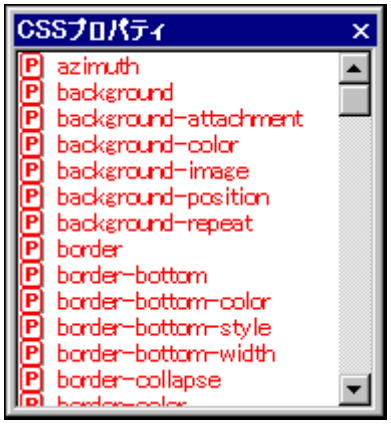
以下の点に注意してください:(i) セレクターの並べ替えを行う際には、(ピリオド、ハッシュ、コロンといった) セレクター全ての箇所が並べ替え評価の対象になり、(ii) 複数のセレクター(例:`h4, h5, h6 {...}`)が1つのルールとしてグループ化されている場合でも、CSS アウトラインではグループ内の各セレクターは別々に表示されます。

CSS アウトライン入力ヘルパーにあるツールバーアイコンを左から順に説明します:

	ドキュメントとの自動同期の有効/無効を切り替えます。自動同期が有効になっている場合、ドキュメント内で入力されたセレクターが(入力中でも)入力ヘルパーに表示されるようになります。
	入力ヘルパーの内容を現在のドキュメントの状態から更新します
	アルファベット順による並び替えの有効/無効を切り替えます。オフになっている場合、セレクターはドキュメント内に記述されている順序で表示されます。アルファベット順で並び替えられた場合、(例: `#intro` などの)ハッシュ記号により記述されるID が最初のセレクターとして表示されます。

プロパティ入力ヘルパー

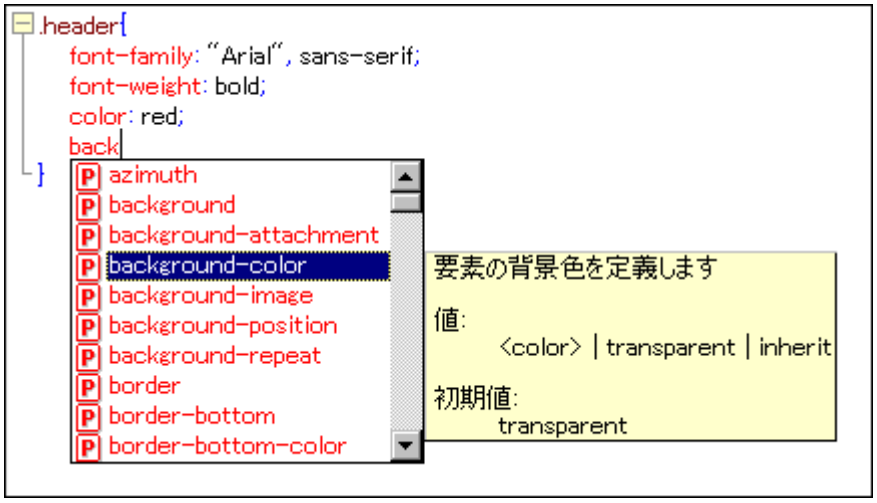
(*以下のスクリーンショットにある*)プロパティ入力ヘルパーには、全てのCSS プロパティがアルファベット順に表示されます。プロパティのアイテムをダブルクリックすることで、カーソルのある位置にそのプロパティが挿入されます。



プロパティの自動補完とプロパティのヒント

プロパティの名前を入力しようとすると、XMLSpy により、入力された文字から始まるプロパティのリストが表示されます(*下のスクリーンショット*)。その他にも、カーソルをプロパティ名の中まで移動させ、**Ctrl+Space** を押下することでCSS プロパティのリストが表示されます。

(C) 2015-2021 Altova GmbH

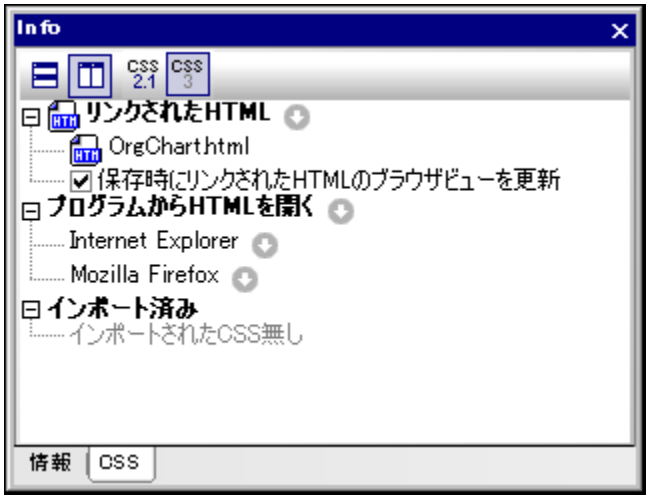


リストをスクロール、または矢印キーによりリストのナビゲーションを行うことで、プロパティの定義と使用可能な値が記されたツールチップが表示されます。表示されるツールチップはハイライトされたプロパティのものです。選択されたプロパティを挿入するには、「Enter」を押下するか、プロパティをクリックしてください。

CSS 情報ウィンドウ

CSS ファイルがアクティブになっていれば、CSS 情報ウィンドウが有効になります(*下のスクリーンショット*)CSS 情報ウィンドウでは以下の機能が利用可能です:

- CSS 2.1 とCSS 3.0 との切り替えを行うことができます。GUI における入力ヘルパーとインテリジェントな編集機能が、情報ウィンドウで選択されているCSS のバージョンにより切り替わります。
- CSS ファイルをHTML ファイルにリンクすることができます。この機能によりCSS ドキュメントの修正により反映される結果をただちに確認することができます。他にも、リンクされたHTML を複数のブラウザーで表示することができ、CSS ドキュメントの修正を異なるブラウザーにて確認することができます。
- CSS 情報ウィンドウにはインポートされたCSS スタイルシートが表示され、アクティブなCSS ドキュメントにインポートされたファイル構造の概要を知ることができます。



以下の点に注意してください:

- 情報ウィンドウのツールバーにはCSS 2.1 とCSS 3.0 のアイコンが含まれます。目的のバージョンを選択することで、選択したバージョンのCSS に対応した入力ヘルパーならびに編集機能を使用することができます。
- アクティブな CSS ドキュメントからリンクすることのできるHTML ファイルは1つだけです。*リンクされた HTML* の左にあるアイコンをクリック、「HTML リンクを設定」コマンドを選択し、目的のHTML ファイルを選択します。選択されたファイルが*リンクされた HTML* アイテムの下に表示されます。このリンクを作成してもCSS やHTML ドキュメントへの修正が施されるわけではありません。このリンク機能により、アクティブな CSS ファイルを適用して、ファイルを試験することができます。
- リンクされたHTML ファイルをダブルクリックすることで、そのHTML ファイルがXMLSpy により開かれます。
- ツールバーアイコンにより、CSS ファイルとHTML ファイルを水平、または垂直に配置することができます。
- 変更されたCSS ドキュメントを保存する際、XMLSpy で開かれているHTML ファイルを自動的に更新することができます。この機能を有効にするには、*保存時にリンクされた HTML のブラウザービューを更新* ボックスにチェックを入れてください。この機能による自動更新は、編集されたCSS ドキュメントへの参照を行っているHTML ファイルに対してしか行われないことに注意してください。
- リンクされたHTML ファイルを変更するには、「HTML リンクを設定」コマンドから、別のHTML ファイルを選択してください。
- HTML ファイルへのリンクを削除するには、*リンクされた HTML ファイル*の右にあるアイコンをクリックし、「*リンク*を削除」コマンドを選択してください。
- プログラムからHTML を開く の右にあるアイコンを使うことで、リストへアプリケーションを追加することができます。コンピューターにインストールされたブラウザーまたは(テキストエディターの様な)任意のアプリケーションをメニューコマンドに追加することができます。割り当てを行うアプリケーションは、通常ブラウザーかエディターです。
- アプリケーションがプログラムが *HTML を開く* リストに追加された後は、(「インストールされたブラウザーを追加」コマンドにより追加されたものを除いて) *HTML を開く* リスト内の名前は選択範囲を選択し、「F2」キーを押下することにより名前の変更を行うことができます。
- *HTML を開く* に表示されているアプリケーションの右にあるアイコンからは、(i) アプリケーションを開く、(ii) アプリケーションを開き、リンクされているHTML ファイルを開く、(iii) リストからアプリケーションを削除するコマンドを選択することができます。アプリケーション名をダブルクリックすることで、リンクされているHTML ファイルをそのアプリケーションで開くことができます。
- *HTML を開く* リストに(から)追加/削除されたアプリケーションは、HTML 情報ウィンドウのアプリケーションとも連動しています。
- *インポート済み* アイテムには、アクティブな CSS ファイルにインポートされたCSS ファイルが表示されます。

13 JSON と JSON スキーマ

JSON（ジェイソン、JavaScript Object Notation）は、軽量のデータストレージとJavaScript 構文を使用する交換形式です。また、XML のように人間が判読できる、テキストのみの書式です。JSON テキストは、すべてプログラミング言語で読み込まれ、使用されることができるので、広範囲におよび、特にウェブ上で、データ交換書式として使用されるようになりました。

IDE 機能の一部として、XMLSpy は、JSON データ（インスタンスドキュメント）[589] の編集および検証をサポートし、構文的にも、文法的にも正確な JSON スキーマ[592] ドキュメントの作成をサポートします。

XMLSpy は Avro と Avro スキーマのサポート[675] を提供します。

JSON5

ECMAScript 5 拡張を追加する JSON の拡張が JSON5 です（詳細に関しては json5.org を参照してください）。JSON5 は、JavaScript の厳密なサブセットあり、新規の JSON データ型を追加せず、全ての既存の JSON コンテンツと作業することができます。

JSON インスタンスドキュメントのために使用することのできる、すべての XMLSpy 機能は、JSON5 インスタンスドキュメントに対して使用することができます。しかしながら、JSON5 と JSON の間には以下の主要な違い、および、これら2種類のフォーマットを XMLSpy が扱う方法の違いが存在します:

- JSON5 は JSON の公式な継承ではありません。このため、自身のファイル拡張子を使用しています: `json5`。
- デフォルトでは、XMLSpy は、`.json` ファイル拡張子を JSON インスタンスドキュメントとして認識します。`.json5` ファイル拡張子を持つインスタンスドキュメントを JSON5 インスタンスドキュメントとして認識します。
- JSON5 インスタンスドキュメントは JSON スキーマに対して検証することができます。Avro インスタンスを表す、JSON インスタンスドキュメントは、JSON スキーマと Avro スキーマに対して検証することができます。詳細に関しては、*JSON ドキュメントの検証*[662] を参照してください。

このドキュメント内では、*JSON インスタンス*という用語は、指定されない限り、JSON と JSON5 インスタンスドキュメントを指します。次を参照してください: JSON5 と JSON 間の違い[590]。

XMLSpy 内の JSON、JSON スキーマ

JSON インスタンスと JSON スキーマ の両方のドキュメントの型は、JSON フォーマットで書かれており、有効性のために整形式の JSON ルールに準拠しなければなりません。（インスタンスとスキーマ）の両方のドキュメントは、通常、`.json` ファイル拡張子を持っています。JSON インスタンスは、テキストビュー[595] とグリッドビュー[600] で編集することができ、JSON スキーマドキュメントをこれら2つのビュー、および、視覚的なスキーマビューである JSON スキーマビュー[630] で、編集することができます。

XMLSpy は、JSON インスタンスと JSON スキーマドキュメントとの作業のために以下をサポートしています:

- テキストビュー[595] には、構文の色分け、構文のチェック、JSON スキーマの関連性がある場合のインスタンスドキュメント内の自動補完、余白の折り返し、構造マーキングなどの機能を使用して、有効な JSON インスタンスと JSON スキーマドキュメントの編集を素早く行うことができます。テキストビュー[595] には、インスタンスおよびスキーマドキュメントの検証が搭載されています。
- グリッドビュー[600] には、ドキュメント構造を更にビジュアル化するテーブルグリッド構造が搭載されています。グリッドビュー[600] で直接編集することができます。切り替え可能な テキストビュー[595] および グリッドビュー[600] は、編集に関するニーズに応えます。グリッドビュー[600] には、インスタンス および スキーマドキュメントの検証が搭載されています。
- テキストビュー[595] および グリッドビュー[600] 内の JSON インスタンスドキュメント検証。情報ウィンドウ[662] 内で割り当てられた JSON スキーマに対して検証は実行されます。
- JSON スキーマビュー[630] は、JSON スキーマをグラフィカルなレイアウトで表示します。これにより、JSON スキーマを素早く作成するためのドラッグアンドドロップ 機能を（テキスト入力に加え）使用することができます。ビュー内の入力ヘルパーは、編集入力を与えます。更に、スキーマの有効性が継続的にチェックされ、エラーが発生するとフラグされます。

JSON インスタンス: 既存のインスタンスドキュメントを開き、新しいインスタンスドキュメントを作成する

- 「オプション | ファイルの型」[1458] タブ内で、JSON/JSON5 インスタンスドキュメントを開くためのデフォルトのビュー(テキストビュー[595] またはグリッドビュー)[600] を設定することができます。既存の JSON/JSON5 ドキュメントは、選択されたデフォルトのビューで開かれます。テキストビュー[595] とグリッドビュー[600] をいつでも切り替えることができます。
- 新しい JSON または JSON5 インスタンスドキュメントを作成するためには、「ファイル | 新規作成」をクリックして、`json: JavaScript Object Notation` または `json5: JSON with ECMAScript 5 extensions.` をそれぞれ選択します。任意で、新しいインスタンスファイルのための JSON 、または、(JSON5 ではなく JSON のための) Avro[675] スキーマファイルを選択するように問われます。スキーマを割り当てるには、割り当ては情報ウィンドウ[662] 内で入力されます。「オプション | ファイルの型」[1458] タブ内の設定に従い、新規 インスタンスドキュメントは、テキストビュー[595] またはグリッドビュー[600] で開かれます。

JSON スキーマ: 既存のスキーマを開く、または、新しいスキーマを作成する

- 既存の JSON スキーマドキュメントを JSON スキーマビュー[630] で開きます。テキストビュー[595] またはグリッドビュー[600] にいつでも切り替えることができます。
- 新規 JSON スキーマドキュメントを作成するには、「ファイル | 新規作成」をクリックして、`json: JSON スキーマ` を選択します。ドキュメントの最初に **`$schema`** キーワードを持つ、新規 JSON スキーマドキュメントが JSON スキーマビュー[630] で開かれます。テキストビュー[595] またはグリッドビュー[600] にいつでも切り替えることができます。

これら全てのビュー(テキストビュー[595] グリッドビュー[600] および JSON スキーマ[630]) に関しては、このセクションのサブセクションで説明されています。

このセクション

このセクションは以下のトピックに整理されています:

- JSON データについて[589] は、JSON ドキュメントの基本的情報について説明しています。
- JSON スキーマについて[592] は、JSON スキーマについてまたその作動について説明しています。
- JSON ラインと JSON コメント[594] ではXMLSpy によりサポートされる2つの JSON 追加仕様についての情報が提供されています。
- テキストビュー内の JSON ドキュメント[595] は、テキストビューの JSON に関連した機能と作業する方法について説明しています。
- グリッドビュー内の JSON ドキュメント[600] は、グリッドビュー内の JSON ドキュメントの編集方法について説明しています。
- JSON スキーマビュー[630] は、ビューの JSON スキーマ編集機能および JSON プロジェクトを作成する際での使用方法について説明しています。
- JSON データドキュメントの検証[662] は、JSON ドキュメント への JSON スキーマの割り当てと JSON ドキュメントの検証について説明しています。
- JSON フラグメントの挿入[664] は JSON テキストフラグメントを JSON ドキュメントに外部ソースから素早く挿入する方法について説明しています。
- XSLT/XQuery を使用した JSON 変換[666] は XPath/XQuery 3.1 を使用してどのように JSON ドキュメントをクエリするかについて説明しています。
- JSON のための XQuery 式[668] は JSON ドキュメントを持つ XQuery の使用方法について大まかに説明しています。
- JSON インスタンスからの JSON スキーマの生成[670] は、インスタンスからのスキーマの生成機能について説明しています。
- JSON スキーマからの JSON インスタンスの生成[673] はスキーマからインスタンスの説明方法について説明しています。
- JSON と XML 間の変換[674] は XMLSpy 内の JSON と XML 間の変換方法について説明しています。

(C) 2015-2021 Altova GmbH

13.1 JSON データ

このセクションは、どのようにJSON データが構成されているかを簡単に説明します。JSON データは通常ドキュメントに保管されていますが、JSON (*インスタンス*) ドキュメントまたは他の型内で*データフラグメント*として保管されることもできます。JSON データフラグメントまたはドキュメントは、下で定義されているようにJSON データ構造です。

XMLSpy は、ECMScript 5 拡張を追加するJ SON の拡張である**JSON5** もサポートします。詳細に関しては、json5.org を参照してください。

JSON オブジェクトと配列

(通常ファイル拡張子 `.json` と共に保存される) JSON ドキュメント は、次のコアデータ構造により構築されます:

オブジェクト

オブジェクト は、中かっこで区切られており、順序のないゼロまたは`key:value` ペアのコレクションです。これらの`key:value` ペアは、**オブジェクトのプロパティ**です。キーは常に文字列である必要があり、引用符で囲まれている必要があります。(プロパティ名と呼ばれる) キーは、コロンにより値から区切られています。プロパティ値は、JSON データ型 (*下のリスト参照* 590) のすべての値を使用することができます。プロパティはカンマで区切られています。下のリストは、(アトミック型の値を持つ) 3 つのプロパティを持つオブジェクトの例です:

```
{
  "emailtype": "home",
  "emailaddress": "contact01.home@altova.com",
  "citycode": 22
}
```

配列

配列 は、角かっこで区切られており、カンマで区切られたゼロまたは複数の**アイテム**のリストです。これらのアイテムは、JSON データ型のすべてを使用することができます (*下のリスト参照* 590)。

- *2 つのオブジェクトを含む配列*

  したの配列は、(それぞれ中かっこにより囲まれた) 2 つのオブジェクトから構成されています。配列自身は、角かっこにより表示されています。

```
[
      {
      "emailtype": "home",
      "emailaddress": "contact01.office@altova.com",
      "citycode": 22
      },
      {
      "emailtype": "office",
      "emailaddress": "contact01.office@altova.com",
      "citycode": 22
      }
  ]
```

- *オブジェクトのプロパティの値である配列*

  したのリストアは、3 つのキーを持つオブジェクトです。各値は、タプル (シーケンス) を含む配列です。(タプルは、ひとつの次元からなる配列と考えられることができます)。各タプル内の 3 つのアイテムは単純型です。

```
{
      "x": [ 1, 2, "abc" ],
      "y": [ 3, 4, "def" ],
      "z": [ 5, 6, "ghi" ]
}
```

JSON データ型

オブジェクトプロパティ値と配列アイテムは次の型であることができます:

- `string` (引用符で囲まれている必要があります)。文字列は `date-time` または `email` フォーマットなどの追加的にフォーマットを持つように指定されることができます。
- `number`: 小数点の部分を持つ数値。整数を含みます。
- `integer`: 小数点の部分を持たない数値。`number` 型のサブセット
- `boolean` (引用符で囲まれていない `true/false`)
- `object`: 他のオブジェクト内で使用されると、データをネストすることができます。
- `array`: オブジェクトにより許可されているより複雑な構造を作成する機能を与えます。
- `null` (引用符で囲まれていない `null`)

JSON データのサンプル

下は、JSON データフラグメントのサンプルです。ドキュメントがどのようにオブジェクトと配列に構造化されているかに注意してください。また、キーの値のデータ型、引用内の文字列の値、他の型は緑に色付けされている点などに注意してください。

```
{
  "first": "Jason",
  "last": "Jones"
  "isManager": true,
  "age": 35,
  "address": {
    "street": "Jason Avenue",
    "city": "Jasonville",
    "state": "JS",
    "postcode": "JS12 ON34"
  },
  "phone": [
    {
      "type": "home",
      "number": "12 3456-7890"
    },
    {
      "type": "office",
      "number": "789 012-34567"
    }
  ],
  "children": [],
  "partner": null
}
```

JSON5 とJSON 間の違いについて

JSON5 は、JavaScript の厳密なサブセットあり、新規のJSON データ型を追加せず、全ての既存のJSON コンテンツと作業することができます。主要な違いは以下にリストされるとおりです:

- JSON5 はコメントをサポートします。コメントは以下のように区切ることができます: `// comment //` または `/* comment */`.
- JSON5 内では、`key:value` ペアのキーは、引用符で囲まれる必要はありません。
- JSON5 内では、文字列を複数のラインにわたり作成することができます。
- JSON5 ドキュメントを Avro に対してではなく JSON スキーマに対して検証することができます。

13.2 JSON スキーマ

XML スキーマがXML ドキュメントの構造とコンテンツを指定するようにJSON スキーマはJSON ドキュメント内のJSON データが整理されるかを指定します。どのデータフィールドが期待されるか、また、値がどのように表示されるかを指定します。JSON スキーマ仕様とJSON スキーマについての情報はここで確認することができます。

JSON スキーマ自身は、JSON オブジェクトです。構文的には、スキーマ全体がJSON オブジェクトの区切りである中かっこで囲まれています(*下のリスト参照*)。スキーマはJSON 構文で書かれており、通常 `.json` の拡張子を持ちます。トップレベルオブジェクトの最初のキーワードである **`$schema`** キーワードを使用して、JSON スキーマとして表示されています。また以下の値を持つべきです: **`"http://json-schema.org/draft-N/schema#"`**。**`N`** がバージョンの番号です。`$schema` キーワードの使用方法のサンプルは以下のとおりです。

```
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   ...
}
```

メモ `$schema` キーワードは、スキーマの最新のバージョンを指定する以下の値を持つことができますが **`"http://json-schema.org/schema#"`**、特定のバージョンを指定するを使用することが奨励されます。. 詳細に関しては、JSON スキーマバージョン[631] を確認してください。

XMLSpy のJSON スキーマビュー内で、グラフィック性に富んだJSON スキーマを作成することができます。作成方法は、JSON スキーマビュー[630] に説明されています。スキーマ編集機能に加え、JSON スキーマビューで以下のスキーマに関連した機能を使用することができます:

- XMLSpy のJSON バリデータを使用して検証: JSON インスタントドキュメントにJSON スキーマを割り当て、XMLSpy 内から、インスタントドキュメントを検証します。 詳細に関しては、JSON ドキュメントの検証[662] を参照してください。
- JSON インスタンスからのJSON スキーマの生成[670]: JSON インスタントドキュメントが既存の場合、JSON スキーマを生成することができます。その後、必要に応じて、スキーマを編集することができます。
- JSON とXML の間の変換[674]: 2つのフォーマットのドキュメント間での変換が行えます。

用語

GUI とこのドキュメント一般的に使用されるJSON スキーマ用語が以下にリストされています。

用語	定義
スキーマ	JSON スキーマドキュメント内のトップレベルのスキーマオブジェクトであるスキーマファイル
オブジェクト	ゼロまたは複数のプロパティを含むJSON 型
プロパティ	オブジェクトの`key:value`[589] ペア。値は任意のJSON データ型であることができます。
キーワード	オブジェクトの`key:value`[589] ペアの`key` 部分。常に文字列です。
サブスキーマ	演算子または依存性の子であるオブジェクト
定義	任意のJSON 型の完全な詳細。定義は、グローバルまたはローカル[637] であることができます。
配列	任意のJSON データ型のゼロまたは複数のアイテムのコンマにより区切られた順序リスト。
アトミック型	`string`, `number`, `integer`, `boolean`, および`null` JSON データ型
型セレクター	すべておよび複数型の選択する[652] `any` および`multiple` 型

演算子	子として定義に追加することのできる発生セレクター。*詳細に関しては、演算子[655] を参照してください。*

JSON データ型

オブジェクトプロパティ値と配列アイテムは次の型であることができます:

- `string`（引用符で囲まれている必要があります）。文字列は`date-time` または`email` フォーマットなどの追加的にフォーマットを持つように指定されることができます。
- `number`: 小数点の部分を持つ数値。整数を含みます。
- `integer`: 小数点の部分を持たない数値。`number` 型のサブセット
- `boolean`（引用符で囲まれていない`true/false`）
- `object`: 他のオブジェクト内で使用されると、データをネストすることができます。
- `array`: オブジェクトにより許可されているよりも複雑な構造を作成する機能を与えます。
- `null`（引用符で囲まれていない`null`）

13.3 JSON ラインと JSON コメント

XMLSpy は JSON ライン（JSONL） とコメント付きの JSON（JSONC）ドキュメントをサポートします。これは JSON ドキュメントの他の型のために これらのドキュメントの検証とインテリジェントな編集機能を使用することができることを意味します。このセクションでは JSON ドキュメントのこれらの型の主要な機能について説明されています。

JSON ライン

JSON ライン（JSONL）は各レコードが次のレコードから新しいラインにより区分されている構造データを保管するためフォーマットです。この結果、レコードは一度に1つ処理され、ログファイルなどのデータを処理する際にこのようなフォーマットがとても役に立ちます。

Example JSON ライン document

```
["Team", "Played", "Won", "Drew", "Lost", "Points"]
["USA", 2, 1, 1, 0, 4]
["France", 3, 1, 1, 1, 4]
["Germany", 1, 0, 1, 0, 1]
["USA", 1, 0, 0, 1, 0]
```

サンプルに関しては、http://jsonlines.org/examples/ を参照してください。

JSON ラインファイルはファイルが `.jsonl` 拡張子を持つ場合、XMLSpy 内などで識別されます。.

コメント付きの JSON

other than JSON5 ドキュメント以外の JSON ドキュメントはコメントを許可していません。コメント付きの JSON フォーマット（JSONC）は JSON ドキュメント内でコメントを許可するように紹介されています。次のコマンドは JSONC 内で使用されており、XMLSpy 内でサポートされています:

- 単一ラインコメント: プレフィックス `//`. 例: `// My comment`
- 複数ラインコメント: 区切り文字 `/*` と `*/`. 例: `/* My comment */`

ファイルが `.jsonc` 拡張子を持つ場合、JSONC はそのままで XMLSpy 内で識別されます。i

メモ JSON5 ドキュメント内でコメントは許可されています。（JSONC と JSON5 以外の）f JSON ファイルの他の型内でのコメントは処理中にエラーを引き起こす可能性があります。

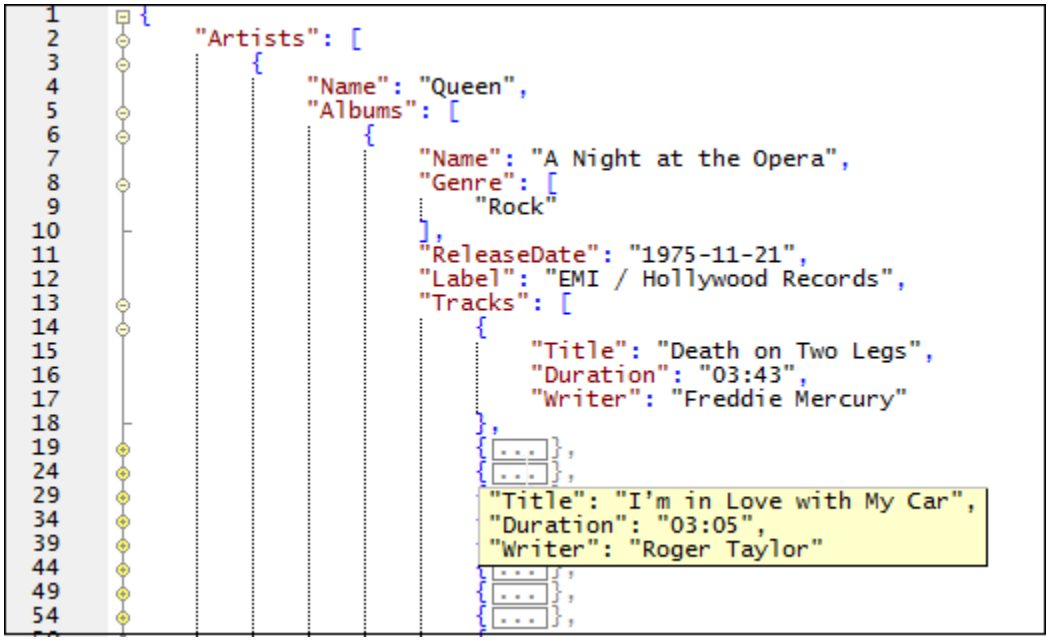
13.4 JSON テキストビュー

Altova ウェブサイト: JSON エディター

JSON スキーマ、Avro スキーマ[678] 、および JSON/JSON5 インスタンスドキュメント (JSON フォーマットの Avro データインスタンスを含む) 場合、テキストビューの機能は以下を含みます: 折り返し余白[595] 、構造マーキング[596]、構文色分け[596]、構文チェック[597] および自動補完[598]。XMLSpy はまた両方の方向の JSON/JSON5 と XML 間の変換[674] を行うことができ、JSON/JSON5 インスタンスからの JSON スキーマの生成[670]を可能にします。

折りたたみ余白

ソースの折りたたみは、JSON キーワードと定義に対して有効化され、ノードの展開および折りたたみを行う機能を意味します。このようなノードは +/- サインによりソースの折りたたみ余白内で表示されます (*下のスクリーンショット参照*)。余白は、テキストビュー設定ダイアログ[1363]内で切り替えオンとオフすることができます。ノードが折りたたまれると、視覚的に省略記号で表示されます (*下のスクリーンショット参照*)。マウスのカーソルが省略記号にポイントされると、折りたたまれたノードの内容がポップアップで表示されます (*スクリーンショット参照*)。内容がポップアップに大きすぎる場合、これは、ポップアップの下に省略記号で表示されます。

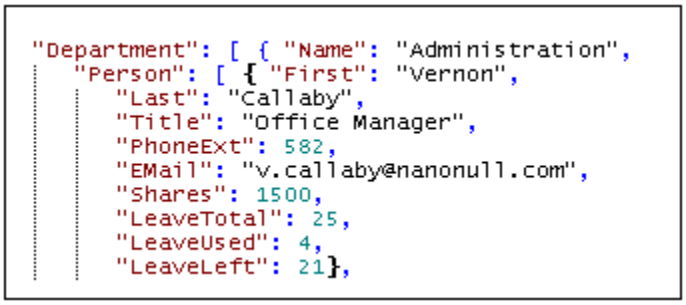


テキストツールバー内の「全ての折りたたみを切り替える」アイコン は、全てのノードを展開された書式でまたはトップレベルのドキュメント要素まで折りたたみ表示します。
ノードの +/- アイコンをクリックすると以下のオプションを使用することができます:

クリック [-]	ノードを折りたたみます。
クリック [+]	子孫ノードが展開されて表示されるか、ノードが折りたたまれた前の状態に従い折りたたまれ、ノードが展開されます。
Shift+ [-]クリック	すべての子孫ノードを折りたたみ、展開フォーム内でクリックされたノードを残します。
Ctrl+ [+]クリック	クリックされたノードとそのすべての子孫ノードを展開します。

構造マーキング

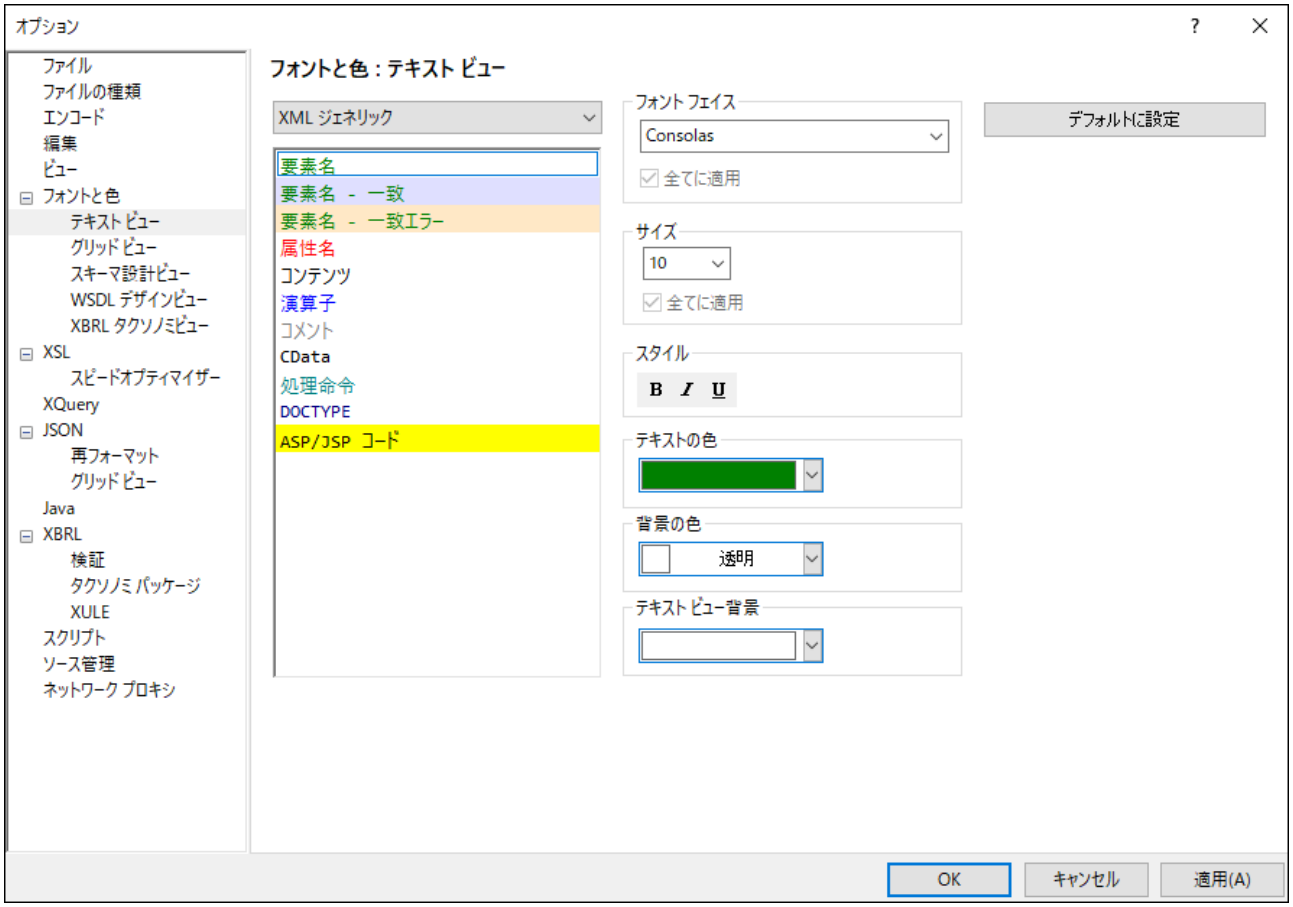
JSON オブジェクトまたは配列の区切りを行う波かっこや角かっこ上にカーソルが位置している場合、対応するペアと共にかっこが太字で表示されます（*下のスクリーンショット*）。この機能により、特定の要素の開始点と終了点を簡単に確認することができます。



構文による色分け

JSON ドキュメント（JSON または Avro 675 *インスタンス/スキーマ*）、および JSON5 ドキュメントはオブジェクト、文字列、値文字列、演算子、数値、そしてキーワードから構成されます。テキストビューでは、（*下のスクリーンショット*にあるように）オプションダイアログ 1454 の設定に従う形で（*上のスクリーンショット*で示されるように）各カテゴリーが異なる色で表示されます。オプションダイアログのテキストフォントタブでは、異なる JSON コンポーネントに対する色を指定することができます。左上のコンボボックスから JSON を選択し、各 JSON アイテムに対する目的の色を（*スタイルペイン*）から選択してください。

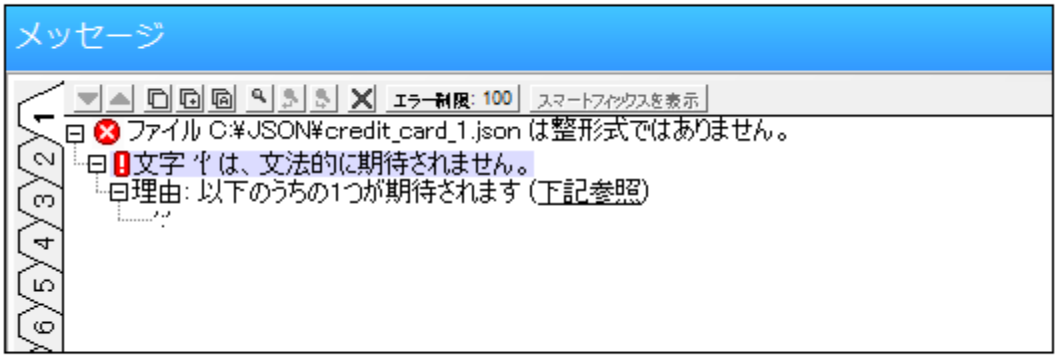
(C) 2015-2021 Altova GmbH



メモ JSON ではなく、JSON5 構文はコメントの使用を許可しています。JSON5 内のコメントは次のように区切ることができます: `// comment //` or `/* comment */`。

構文のチェック

[XML | 整形式のチェック(F7)] コマンドを選択することで JSON ドキュメント (JSON または Avro 675 インスタンス/スキーマ) の構文チェックを行うことができます。整形式チェックの結果は(以下のスクリーンショットにあるように) メッセージウィンドウに表示されます。



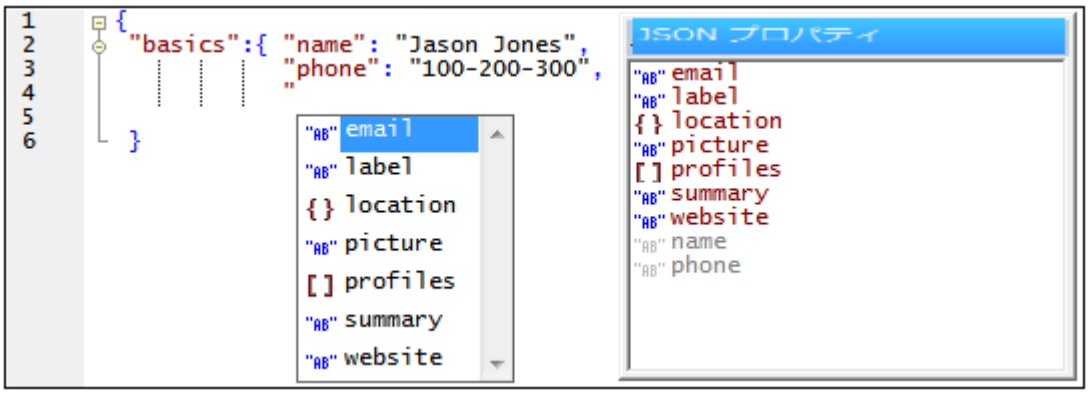
上のスクリーンショットで示されるエラーメッセージでは以下のことが指摘されています: コロンが必要な箇所で中かっこが存在します。

自動補完

自動補完は、編集される JSON ドキュメント (JSON または Avro インスタンス/スキーマ) がスキーマに関連する場合に有効化されます。

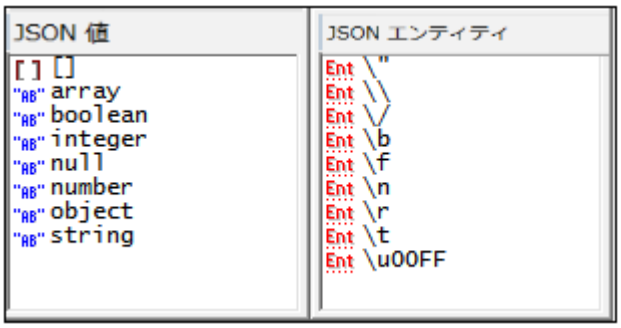
- ドキュメントが JSON スキーマの場合、自動補完は `$schema` [592] キーワードにより指定されているスキーマバージョンをベースにしています。詳細に関しては、JSON スキーマバージョン [631] を確認してください。
- ドキュメントが JSON/JSON5 インスタンスの場合、自動補完が有効化されるためには、JSON スキーマがインスタンスに割り当てられていなければなりません。 [662]
- ドキュメントが JSON フォーマットの Avro データドキュメントの場合、自動補完が作動するためには、 Avro スキーマがインスタンスに割り当てられていなければなりません。 [662]
- ドキュメントが Avro スキーマ [678] の場合、自動的に Avro スキーマのためのスキーマに関連付けられ、自動補完はこのスキーマをベースにします。

自動補完は、カーソルのロケーションで使用可能な入力オプションを提供します。これは以下により行うことができます (i) メインウィンドウ内のポップアップ、(ii) 入力ヘルパーの使用 (*下のスクリーンショット参照*)。ポップアップと入力ヘルパーはそれぞれ、カーソルのロケーションで有効な入力のリストを表示します。ポップアップリストのエントリを閲覧するには、矢印キーを使用して移動します。スキーマがエントリの詳細を含む場合は、詳細はスキーマ内のエントリの 説明 キーワード) ハイライトされたポップアップのエントリの横に表示されます。ポップアップウィンドウからエントリを選択または入力ヘルパー内のエントリをダブルクリックして挿入します。



上のスクリーンショットに表示されるインスタンスドキュメント内で、ポップアップとJSON プロパティ入力ヘルパーが、プロパティ名を示す引用符の後にカーソルがポイントされると表示されます。入力ヘルパーはこの時点で許可されているすべてのプロパティを表示し、既に入力されているプロパティは灰色で表示され無効化されています。ポップアップはその時点で使用することのできるプロパティを表示します。

他にも2つの入力ヘルパーがあります: JSON 値とJSON エンティティ(下のスクリーンショット)。これらはそれぞれ、文字列内のエスケープ文字のための許可されている `key:value` ペアとエンティティを表示します。JSON 値入力ヘルパーは、JSON スキーマを編集中に許可されている型キーワードを表示しています。JSON エンティティ入力ヘルパー `\u00FF` 内の最後の入力は、ユニコード文字をあらわすプレースホルダです。青でハイライトされた部分を使用するユニコード文字で置き換えることができます。



他のコンテキストを考慮した自動補完のエントリまたはヒントは、スキーマ内で指定されている場合、以下を含みます: 列挙、詳細、必要とされる発生とデフォルトの値。

13.5 JSON グリッドビュー

グリッドビューにより、JSON ドキュメント（JSON インスタンス/スキーマ または Avro スキーマ 678）の構造をグリッドで確認することができ、構成されたテキストのブロックを再構成することができます。これにより、特に長く複雑に構成されたドキュメントの場合、テキストビューで必ず使用できるとも限らない、概要と編集機能が与えられます。JSON ドキュメントの場合、そのような複雑性は、他の配列とオブジェクトで複数のレベルにおよびネストされている配列とオブジェクトのフォームで出現する可能性があります。例えば、（テキストビューに表示されると同様）下にリストされる JSON テキストおよびグリッドビューでの表示を比較する場合（更に下のスクリーンショットで表示されています）。

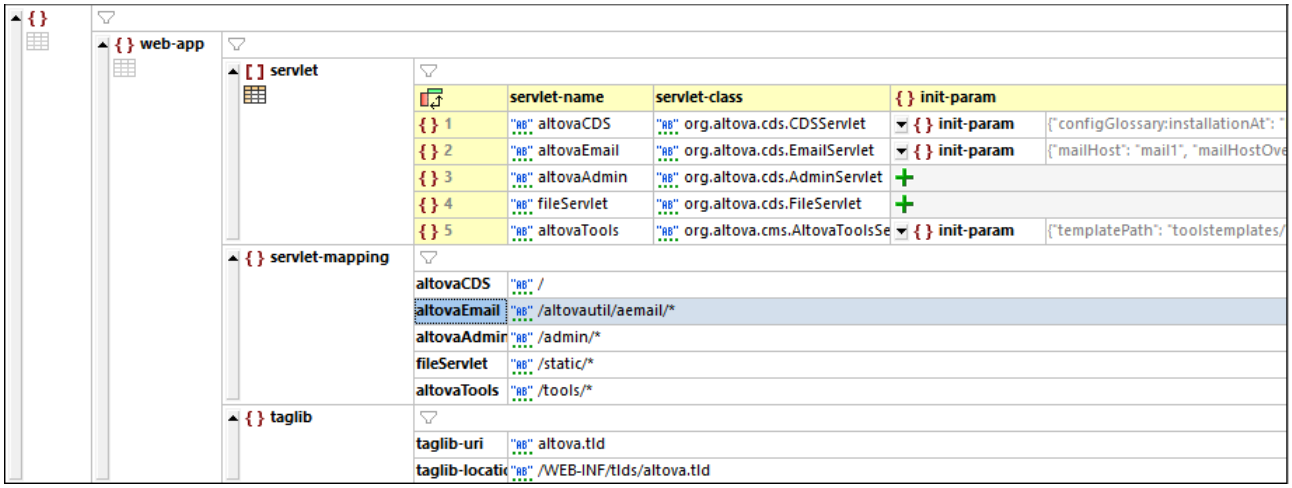
メモ **Avro** は、Enterprise Edition のみでサポートされています。

テキストビュー内にリストされる JSON コード

```
{
  "web-app": {
  "servlet": [
    {
        "servlet-name": "altovaCDS",
        "servlet-class": "org.altova.cds.CDSServlet",
        "init-param": {
        "configGlossary:installationAt": "Philadelphia, PA",
        "configGlossary:adminEmail": "ksm@pobox.com",
        "configGlossary:poweredBy": "Altova",
        "configGlossary:poweredByIcon": "/images/altova.gif",
        "configGlossary:staticPath": "/content/static",
        "templateProcessorClass": "org.altova.WysiwygTemplate",
        "templateLoaderClass": "org.altova.FilesTemplateLoader",
        "templatePath": "templates",
        "templateOverridePath": "",
        "defaultListTemplate": "listTemplate.htm",
        "defaultFileTemplate": "articleTemplate.htm",
        "useJSP": false,
        "jspListTemplate": "listTemplate.jsp",
        "jspFileTemplate": "articleTemplate.jsp",
        "cachePackageTagsTrack": 200,
        "cachePackageTagsStore": 200,
        "cachePackageTagsRefresh": 60,
        "cacheTemplatesTrack": 100,
        "cacheTemplatesStore": 50,
        "cacheTemplatesRefresh": 15,
        "cachePagesTrack": 200,
        "cachePagesStore": 100,
        "cachePagesRefresh": 10,
        "cachePagesDirtyRead": 10,
        "searchEngineListTemplate": "forSearchEnginesList.htm",
        "searchEngineFileTemplate": "forSearchEngines.htm",
        "searchEngineRobotsDb": "WEB-INF/robots.db",
        "useDataStore": true,
        "dataStoreClass": "org.altova.SqlDataStore",
        "redirectionClass": "org.altova.SqlRedirection",
        "dataStoreName": "altova",
        "dataStoreDriver": "com.microsoft.jdbc.sqlserver.SQLServerDriver",
        "dataStoreUrl":
"jdbc:microsoft:sqlserver://LOCALHOST:1433;DatabaseName=goon",
        "dataStoreUser": "sa",
        "dataStorePassword": "dataStoreTestQuery",
        "dataStoreTestQuery": "SET NOCOUNT ON;select test='test';",
        "dataStoreLogFile": "/usr/local/tomcat/logs/datastore.log",
```

```
          "dataStoreInitConns": 10,
          "dataStoreMaxConns": 100,
          "dataStoreConnUsageLimit": 100,
          "dataStoreLogLevel": "debug",
          "maxUrlLength": 500
        }
        }, {
          "servlet-name": "altovaEmail",
          "servlet-class": "org.altova.cds.EmailServlet",
          "init-param": {
            "mailHost": "mail1",
            "mailHostOverride": "mail2"
          }
        }, {
          "servlet-name": "altovaAdmin",
          "servlet-class": "org.altova.cds.AdminServlet"
        }, {
          "servlet-name": "fileServlet",
          "servlet-class": "org.altova.cds.FileServlet"
        }, {
          "servlet-name": "altovaTools",
          "servlet-class": "org.altova.cms.AltovaToolsServlet",
          "init-param": {
            "templatePath": "toolstemplates/",
            "log": 1,
            "logLocation": "/usr/local/tomcat/logs/AltovaTools.log",
            "logMaxSize": "",
            "dataLog": 1,
            "dataLogLocation": "/usr/local/tomcat/logs/dataLog.log",
            "dataLogMaxSize": "",
            "removePageCache": "/content/admin/remove?cache=pages&id=",
            "removeTemplateCache": "/content/admin/remove?cache=templates&id=",
            "fileTransferFolder":
    "/usr/local/tomcat/webapps/content/fileTransferFolder",
            "lookInContext": 1,
            "adminGroupID": 4,
            "betaServer": true
          }
        }
      ],
      "servlet-mapping": {
        "altovaCDS": "/",
        "altovaEmail": "/altovautil/aemail/*",
        "altovaAdmin": "/admin/*",
        "fileServlet": "/static/*",
        "altovaTools": "/tools/*"
      },
      "taglib": {
        "taglib-uri": "altova.tld",
        "taglib-location": "/WEB-INF/tlds/altova.tld"
      }
    }
  }
```

(*上にリストされる*) テキストビュー内のドキュメント構造を完成することは、時間をかけ慎重に確認する必要があります。(*下のスクリーンショットの*) グリッドビューでの構造は一目で確認することができます。



更に、構造は、簡単にオブジェクトを追加、削除、またはグリッド内でオブジェクトを移動させることで簡単に変更することができます。テキストの全体のブロックを(例えば、並べ替えまたは移動して)再編成することができます。コンテンツも同様にグリッドビューで編集することができます。これは個別のセル内でインセルコマンド[607]が使用できるようになるためです。

グリッドビューツールバー

アプリケーションウィンドウの上のツールバーエリアにあるグリッドビューツールバーを使用してグリッドビュー内の編集に関連するコマンドにアクセスすることができます(*下のスクリーンショット*)。名前とショートカットを確認するにはツールバーアイコンにマウスをポイントしてください。

メモ これらのコマンドはJSON メニュー[1239]内で使用可能です。

このセクション

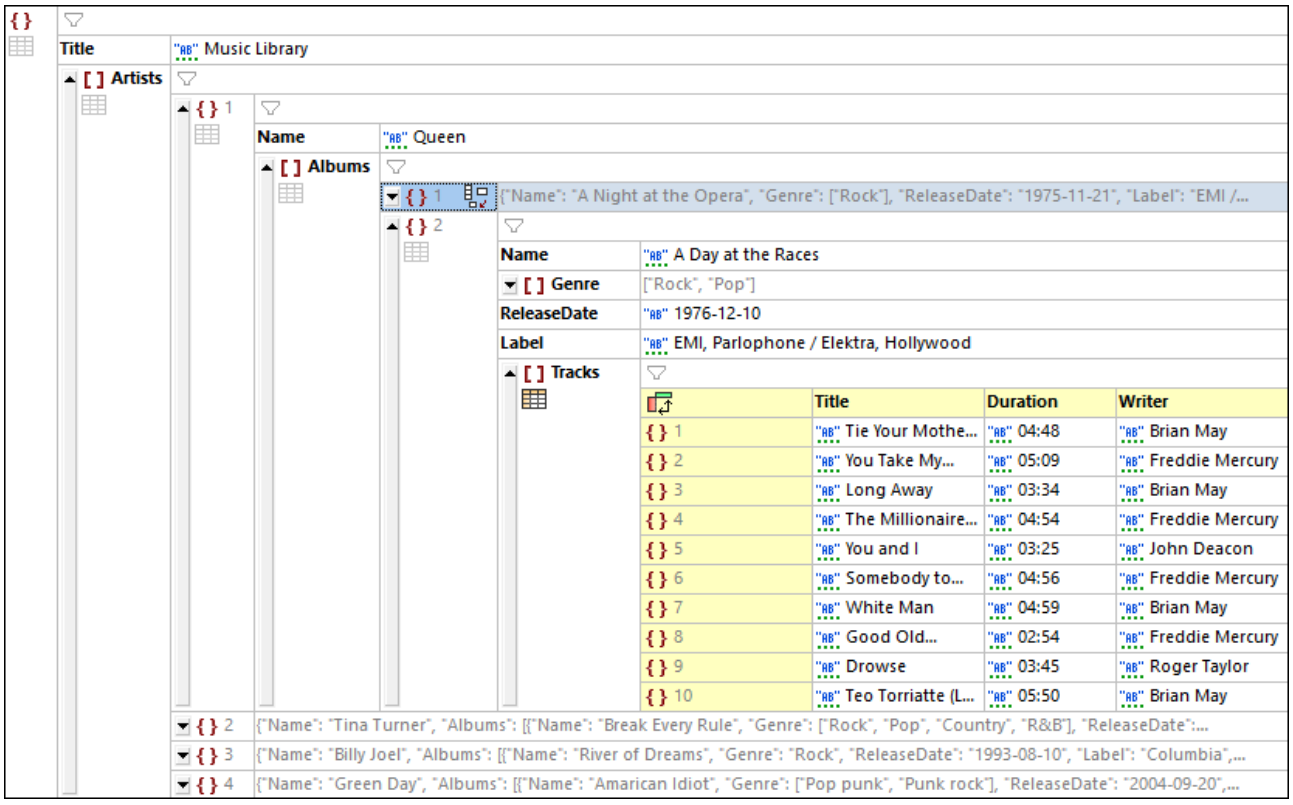
このセクションは以下のリストに整理されています: JSON グリッドビュー固有の機能について説明されています。

- JSON ドキュメントがどのように JSON グリッドビューで表示されるか[603]
- ドキュメント構造の編集[606]
- ドキュメントコンテンツの編集[607]
- グリッドビュー内でのドラッグとドロップ[610]
- テーブルの表示の使用[613]
- XQuery フィルターの使用[617]
- XQuery フォーミュラの使用[618]
- グリッドビュー内のイメージの表示[622]
- グリッドビュー内のチャート[623]
- グリッドビュー内のコンテキストメニュー[625]
- グリッドビュー設定[627]

(C) 2015-2021 Altova GmbH

13.5.1 グリッドビュー内でのドキュメントの表示

JSON グリッドビューはグリッド内にJSON ドキュメントを階層的に表示します（*下のスクリーンショットを参照*）。



各グリッドラインは次のデータ構造を表示します。それぞれが下のテーブルでシンボルと共に表示されています。配列とオブジェクトの両方がオブジェクト、配列、または動的な値であることのできる子コンポーネントを含むことができます。

{ }	オブジェクト（*定義を参照してください*[589]）。オブジェクトには `key:value` ペアが含まれています。
[ ]	配列（*定義を参照してください*[589]）。配列には通常オブジェクトまたは値であるアイテムが含まれています。
	`key:value` ペア（*定義を参照してください*[589]）。

上のスクリーンショットでは、2つの `key:value` ペアを含むルートオブジェクトを確認することができます: (i) **`"Title":"Music Library"`**, (ii) **`"Artists":[Array]`**.**`Artists`** にキーされた配列にはオブジェクトである4つのアイテムが含まれています。

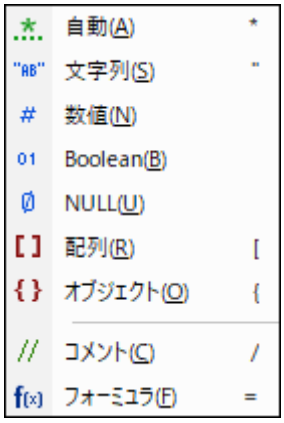
以下の点に注意してください：

- Ctrl + **マウスのホイール** または Ctrl+[プラス] と Ctrl+[マイナス] を使用してグリッドをズームすることができます。
- 同じレベルのオブジェクト、配列、配列アイテムは 1 から開始する番号が与えられています。配番されています。上のスクリーンショットでは、例えば、`Artists` 配列内ではオブジェクトは `1` から `4` に番号が付けられています。`Albums` と `Tracks` 配列内でも同様です。この番号づけは実際のドキュメント内には含まれておらず、ドキュメントの構造を確認するためのJSON グリッドビュー機能です。

- オブジェクトと配列は構造のシンボルの左側の矢印アイコンを押して展開または退縮することができます *(上のスクリーンショットを参照)*。データ構造が退縮されると、そのコンテンツは単一のラインで表示されます。上のスクリーンショットの最後の3列を確認してください。
- 複数のコンポーネントを同じレベルで選択すると、これらを **Shift** を押し、選択されたコンポーネントの矢印をクリックして展開/折りたたむことができます。
- 配列とオブジェクトが展開されると以下に注意してください：(i) 配列とオブジェクトのコンテンツはシンボルの次のラインから開始します。(iii) 配列 (キー) の名前は配列シンボルと同じラインに存在しますが、配列のアイテムは次のラインから開始します。配列またはオブジェクトが退縮されていると、コンテンツは同じラインに表示されます。
- テーブルディスプレイ[613]の一部である以外 `key:value` ペアは1つのラインに表示されることに注意してください (この場合キーは列ヘッダーになります)。テーブルディスプレイ[613] の例としては、上のスクリーンショット内の `Tracks` 配列のコンテンツを参照してください。

データ型 (または型) のシンボル

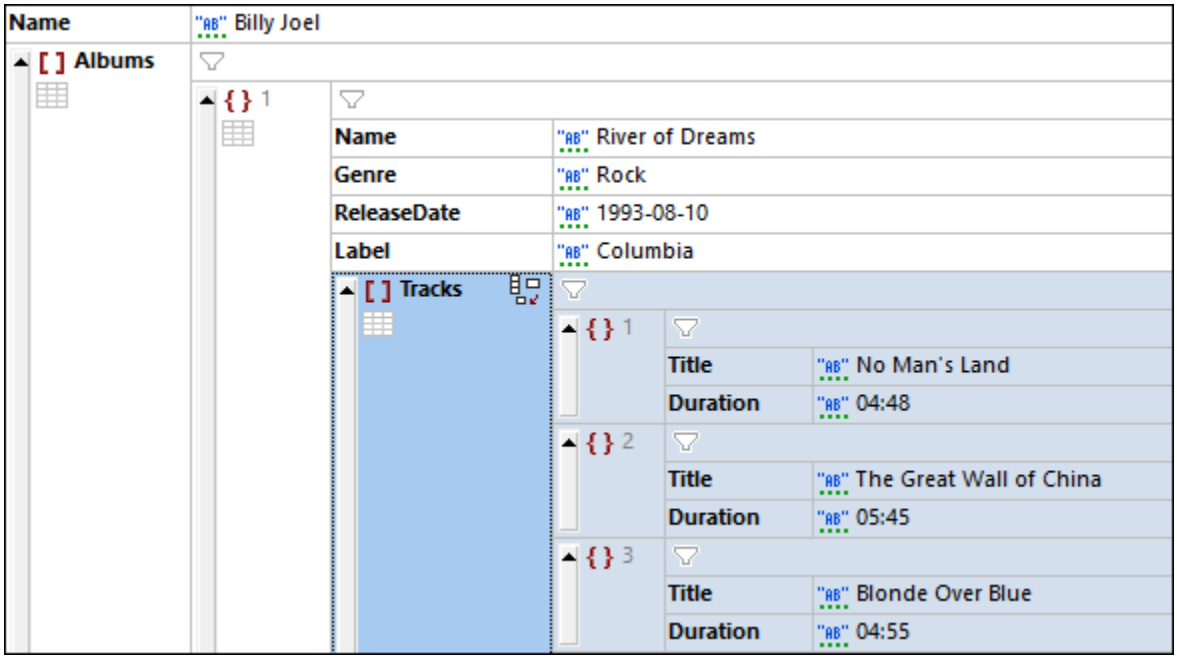
オブジェクトプロパティ値と配列アイテムは異なる型[590]であることができます。シンボルにより示されるそれぞれの型は下のスクリーンショットで示されています。(上のスクリーンショットでは、全ての `key:value` ペアの値の型は `string` 型であることに注意してください)。

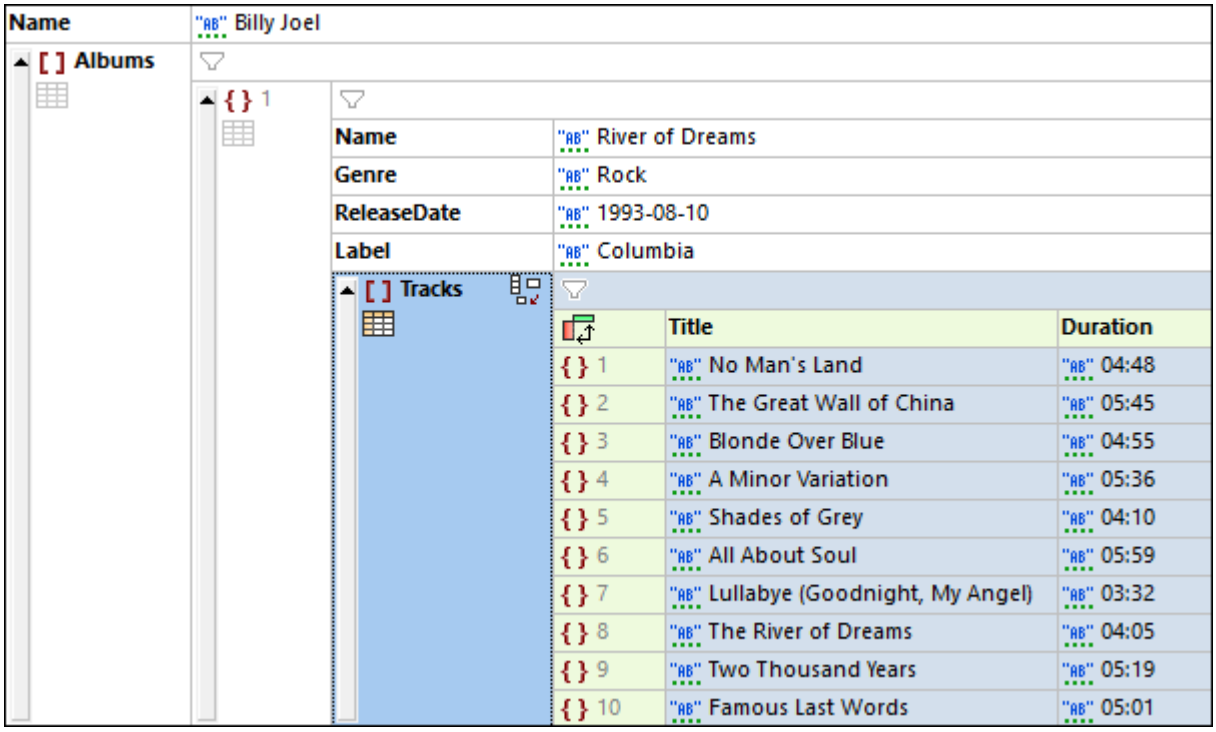


メモ SONC とJSON5 内でコメントはサポートされています。フォーミュラ[617] はXMLSpy に固有です。

テーブルディスプレイ

少なくとも1つのオブジェクトまたは配列を含むオブジェクトと配列は*(左側のスクリーンショット内でハイライトされている)* リストまたは *(右側のスクリーンショット内でハイライトされている)* テーブルでビューすることができます。表示は個別のオブジェクトと配列のためにリストディスプレイとテーブルディスプレイの間を切り替えることができます。



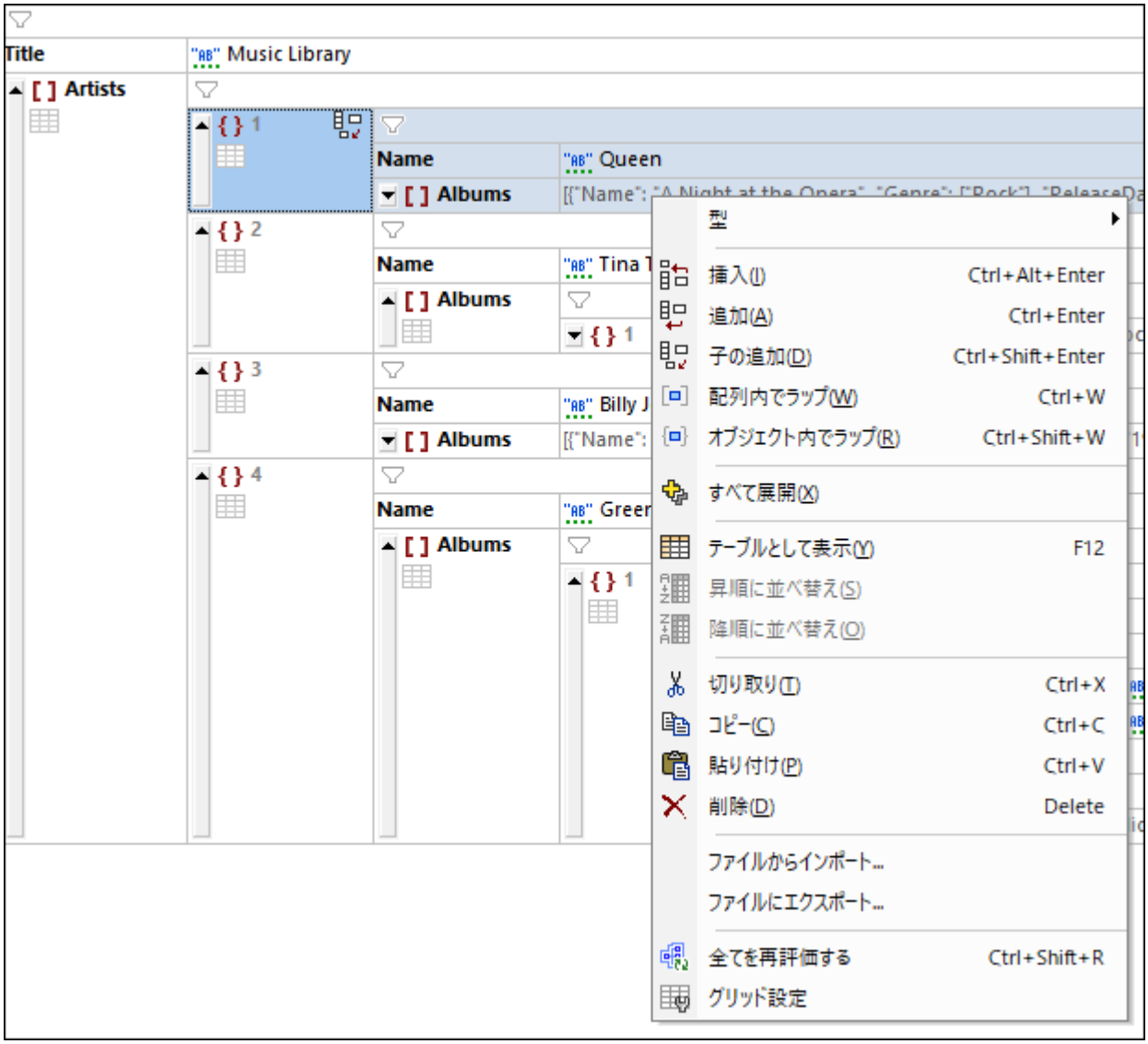


JSON グリッドビューのテーブルディスプレイの詳細に関してはセクション テーブルディスプレイ[613]を参照してください。

13.5.2 ドキュメント構成の編集

JSON ドキュメント構造を新規コンポーネントを追加、選択されたグリッドブロックを削除または移動してして編集することができます。ドキュメント構造を以下の方法で編集することができます:

- コンテキストメニュー内でコマンドを使用するためにコンポーネントを選択して右クリックします (*下のスクリーンショットを参照*)。セクション コンテキストメニュー[625] でコマンドについて説明されています。
- このセル内で使用可能な編集ボタンをします。これらのボタンについては下のセクション セル内のコマンドボタン[607] で説明されています。
- コンポーネントを選択してドラッグアンドドロップを使用して新規のロケーションにドロップし構造を変更します。



構造を作成するためのコマンドのショートカットは下のテーブルにリストされています。

コマンド	ショートカット

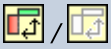
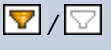
(C) 2015-2021 Altova GmbH

後に(兄弟) を挿入	Ctr+Enter
前に(兄弟) を挿入	Ctr+Shift+Enter
付加(兄弟)	Ctr+E
子の追加	Ctr+Alt+Enter
配列内でラップ	Ctr+W
オブジェクト内でラップ	Ctr+Shift+W

メモ プロパティとしてコンポーネントが追加(挿入、付加、子として追加) され、型をオブジェクトまたは配列に変更する場合、これをコンポーネントの型シンボルをクリックして型を選択します[607]。

セル内のコマンドボタン

JSON グリッドビュー内では、グリッドのセル内のボタンは対応するセルのコンテキスト内で使用可能な機能を提供します。これらのボタンコマンドの使用は一般的に使用される操作を更に素早く簡単に実行する方法です。使用可能なセル内のボタンコマンドは下のテーブルにリストされています。表示に関連したコマンドは青い背景と共に表示されています。ドキュメント (構造とコンテンツ) の編集のためのコマンドはベージュの背景で表示されています。特定のセルのコンテキスト内の追加コマンドのためにセルのコンテキストメニュー[625]を使用してください。メモ ボタンをポイントするとセルボタンのコマンドの名前はツールのヒントとして表示されます。

	セルコンテンツに従いグリッド列の幅を最適化
	色で表示されている場合 テーブルの表示[613] がオンに切り替えられています。それ以外の場合はオフです。表示するためにクリックします。
	テーブルの左側のセル内のコマンドをトグルします。行を列に、または列を行に切り替えます。
	色で表示されている場合、テーブル、オブジェクト、または配列のためのフィルター[617] が有効化されています。それ以外の場合フィルターは無効化されています。無効化/有効化するためにクリックします。式を編集するには、ダブルクリックします。
	セルをワードラップするためのトグルコマンド。選択されると、ワードラップがオンになります。デフォルトでは、アイテムのコンテンツのみがラップされます。アイテムの名前を追加でラップする場合、このオプションをアイコンのドロップダウンリストから選択します。
	配列内: リストまたはテーブルに配列アイテムをふかします。 オブジェクト内: `key:value` ペア(をリスト内で、または(テーブル表示で) 新規のテーブル列のセルとして) 付加します。
	`key:value` ペアを追加します。デフォルトでは値の型は `string` です。
	プロパティ値のデータ型を選択、またはコメントまたはフォーミュラを入力します。

メモ 他のツールバーコマンドの詳細に関しては、セクション グリッドビュー内のコンテキストメニュー[625]を参照してください。

13.5.3 ドキュメントコンテンツの編集

JSON グリッドビュー内のコンテンツの編集はとても理解しやすいです: 編集するセルの内部をダブルクリックして必要に応じてコンテンツを編集します。更にJSON グリッドビューは以下の編集機能を提供します:

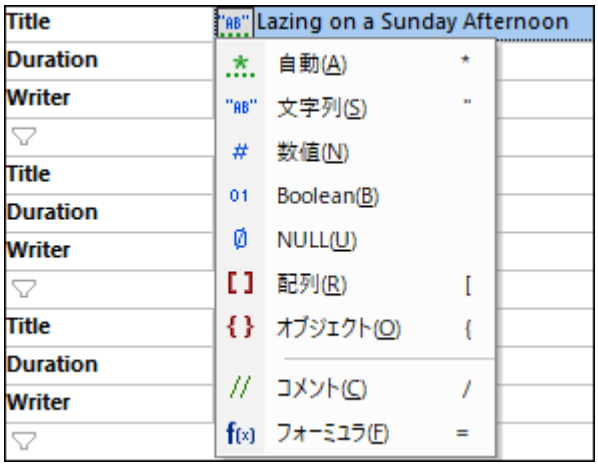
- 型選択機能および自動型検知と値の変換機能
- JSON スキーマをベースにした JSON ドキュメント内の自動補完

型の自動検知の型選択

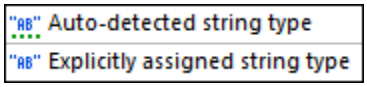
JSON グリッドビューは次の型のカテゴリを区別します:

- *単純型:* String、Number、Boolean、Null
- *特別な単純型:* (値から検知される) Auto、Comment (JSONC、JSON5)
- *コンテナー型:* Object、Array
- *特別のコンテナー型:* フォーミュラ

値の前のシンボルにより示される値の型 *(下のスクリーンショットを参照)*。型を変更するには、シンボルをクリックして、表示されるメニューから型を選択します。型のシンボルとショートカットは下のスクリーンショットに表示されています:



自動検知される型は緑の点線が下に表示されています *(下のスクリーンショットを参照)*。明示的な型は割り当てられる型です。



型に関連したアクションは2つのシチュエーションで発生します:

- JSON ドキュメントがロードされる場合: 全ての単純型は値から自動的に検知された `Auto` に変換されます。例えば、`"MyString"` は自動的に文字列型、`123` が数値型として、`true` がブール型として、および `null` が Null 型として検知されます。曖昧な文字列の場合、型を明示的に選択してください。
- データ構造、または値が入力された場合: 単純型は自動的に検知され型は自動的に割り当てられます。型を後に変更することができます。

メモ JSON グリッドビュー設定 627 により以下を指定することができます (i) 複数のセルが選択されている場合型の変更がどのように処理されるかを指定。(ii) 型が配列またはオブジェクトに変更される場合、動的な型の値がどのように扱われるかを指定。

自動補完

編集中の自動補完は JSON ドキュメント (JSON インスタンス/スキーマ または Avro スキーマ) がスキーマに関連付けられている場合有効化されます。

- ドキュメントがJSON スキーマの場合、自動補完は$schema[592] キーワードにより示されるスキーマバージョンをベースにします。詳細に関しては、JSON スキーマバージョン[631]を参照してください。
- ドキュメントがJSON/JSON5 インスタンスの場合、自動補完を有効化するには、JSON スキーマがインスタンスに割り当てられている必要があります[662]。
- ドキュメントがJSON 書式のAvro データドキュメントの場合、自動補完が作動するにはAvro スキーマがインスタンスに割り当てられている必要があります[662]。
- ドキュメントがAvro スキーマ[678]の場合、Avro スキーマのためのスキーマに自動的に関連付けられ、自動補完はこのスキーマをベースにします。

自動補完はカーソルの位置で使用することのできる入力オプションを提供します。これは以下を使用して行うことができます(i) メインウィンドウ内のポップアップを使用して(ii) 入力ヘルパーを使用して(*下のスクリーンショットを参照*)。ポップアップと入力ヘルパーはそれぞれカーソルのロケーションで有効なエントリのリストを表示します。ポップアップリスト内のエントリを移動するには矢印キーを使用してください。エントリをポップアップウィンドウから選択、または入力ヘルパー内のエントリをダブルクリックして挿入します。

セル内のコマンドボタン

JSON グリッドビュー内では、グリッドのセル内のボタンは対応するセルのコンテキスト内で使用可能な機能を提供します。これらのボタンコマンドの使用は一般的に使用される操作を更に素早く簡単に実行する方法です。使用可能なセル内のボタンコマンドは下のテーブルにリストされています。表示に関連したコマンドは青い背景と共に表示されています。ドキュメント(構造とコンテンツ)の編集のためのコマンドはベージュの背景で表示されています。特定のセルのコンテキスト内の追加コマンドのためにセルのコンテキストメニュー[625]を使用してください。メモ ボタンをポイントするとセルボタンのコマンドの名前はツールのヒントとして表示されます。

	セルコンテンツに従いグリッド列の幅を最適化
/	色で表示されている場合テーブルの表示[613]がオンに切り替えられています。それ以外の場合はオフです。表示するためにクリックします。
/	テーブルの左側のセル内のコマンドをトグルします。行を列に、または列を行に切り替えます。
/	色で表示されている場合、テーブル、オブジェクト、または配列のためのフィルター[617]が有効化されています。それ以外の場合フィルターは無効化されています。無効化/有効化するためにクリックします。式を編集するには、ダブルクリックします。
	セルをワードラップするためのトグルコマンド。選択されると、ワードラップがオンになります。デフォルトでは、アイテムのコンテンツのみがラップされます。アイテムの名前を追加でラップする場合、このオプションをアイコンのドロップダウンリストから選択します。
	配列内: リストまたはテーブルに配列アイテムをふかします。 オブジェクト内: `key:value` ペア(をリスト内で、または(テーブル表示で) 新規のテーブル列のセルとして)付加します。

	key:value ペアを追加します。デフォルトでは値の型は string です。
	プロパティ値のデータ型を選択、またはコメントまたはフォーミュラを入力します。

メモ 他のツールバーコマンドの詳細に関しては、セクション[グリッドビュー内のコンテキストメニュー](625)を参照してください。

変更中に検証

編集中に検証 モードはデフォルトでオンに切り替えられています。オンに切り替えられていると、JSON グリッドビュー内でのドキュメントの変更中に整形式チェックと検証チェックが実行されます。(整形式のチェックに加え) JSON ドキュメントの検証を実行するには、JSON スキーマが JSON ドキュメントに割り当てられている必要があります。エラーはエラーの存在するテキストを赤色で表示し、エラーの箇所を赤い感嘆符で表示することで表示されています。詳細に関しては[JSON ドキュメントの検証](662) を参照してください。

編集中に検証 モードは (i) [「XML | 編集中に検証」](1233) メニューコマンド (ii) **編集中に検証** ツールバーボタン または (iii) [オプションダイアログの検証設定](1455) の *編集時* オプションによりオンとオフに切り替えることができます。

検索と置換

JSON グリッドビュー内でテキストを検索し置き換えるために「**検索**」(Ctrl+F) と「**置換**」(Ctrl+H) コマンドを編集メニューで使用することができます。結果は含まれるコンポーネントもオレンジ色でハイライトされた状態でオレンジ色でハイライトされています。長いテキスト文字列を含むセルの場合、セルのみがハイライトされています。

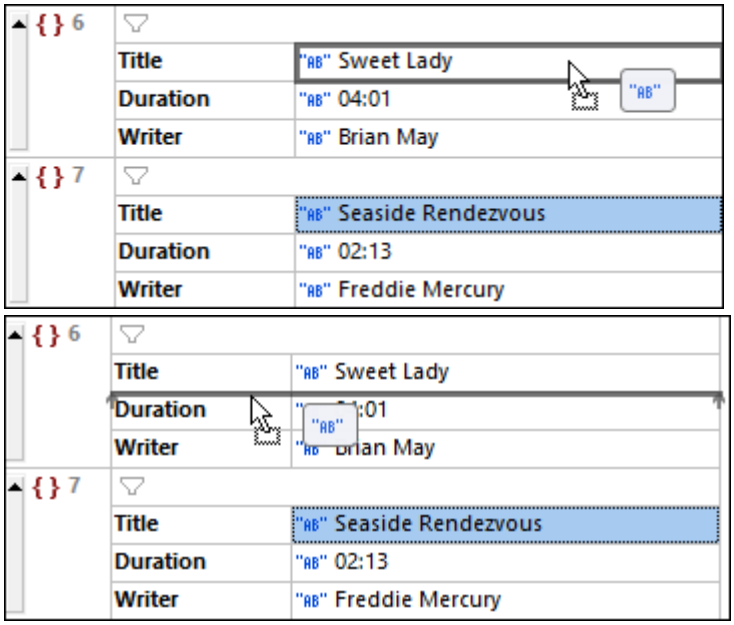
13.5.4 グリッドビュー内でのドラッグとドロップ

グリッドビューは JSON ドキュメントフラグメントをグリッドビューに XMLSpy 内、外部アプリケーション、または Web サイトで開かれているドキュメントからドラッグすることのできる役に立つドラッグオーバーレイを提供します。

ターゲットノードにカーソルをポイントすると、ドラッグされているアイテムだけでなくドロップされた際にどのようにアイテムが作成されるかについての情報を提供するドラッグオーバーレイが表示されます。例えば、下のスクリーンショットでは `Seaside Rendezvous` がドラッグ (移動) されています。アイテムが文字列であることが AB ポップアップにより示されています。

- (左側のスクリーンショットで示されている通り) 値フィールドにポイントすると、フィールド全体がマークされ、マークされたフィールドの値がドラッグされたフィールドの値と置き換えられます。
- (右側のスクリーンショット内の `Duration` などの) ノード名にポイントすると、その箇所にドロップされる値文字列が値キーペアの値ノードを作成することを示す挿入ラインが表示されます。この場合のキーは空の文字列になります。

メモ **Ctrl+Z** または メニューコマンド **Edit | Undo** を使用して移動を元に戻すことができます。

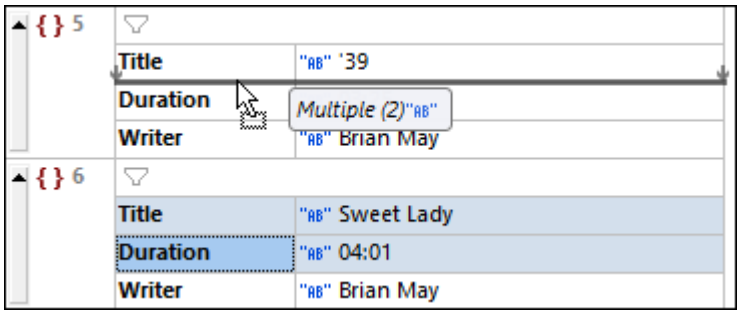


ドラッグオーバーレイ内に含まれている情報

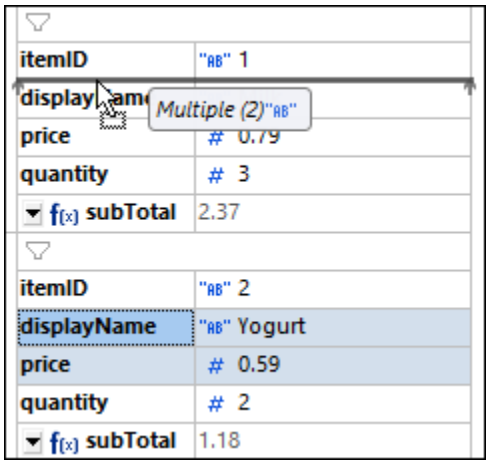
以下の種類のドラッグオーバーレイ情報が表示されます。

通常のグリッドビュー

- 値フィールドにドロップされた値はターゲットフィールド内の値を上書きするためにソース値を移動します（*左上のスクリーンショット参照*）。
- ツリーにノードとしてドロップされる値はキー値ペアの値ノードを作成します（*右上のスクリーンショット参照*）。キーは空の文字列になります。
- 単純型の複数のノードのオーバーレイはドロップされたノード型とこれらのノードの数値を表示します。（値ではなく）ノードを選択するために、ノードの名前をクリックする必要があります。下のスクリーンショットでは、ドラッグオーバーレイ内の情報は2個の型文字列のキー値ペアがドロップされることを示しています。

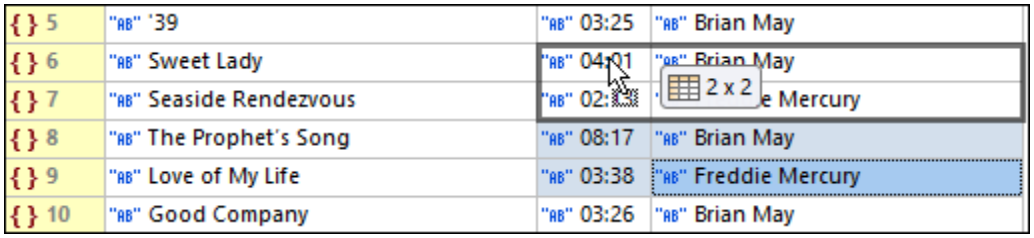


- 異なる型の複数のノードのためのオーバーレイは新規のノードが文字列のデフォルトの型と共に作成されることを表示しています（*下のスクリーンショット参照*）。対応する *型の変更* アイコンをクリックすることでドロップされたノードの型を変更することができます。（値ではなく）ノードを選択するために、ノードの名前をクリックする必要があります。



テーブルの表示

- 値がドラッグされると、選択されたセルの数値は `列 x 行` により示されています(*下のスクリーンショット参照*)。ドラッグされたセルの間トリックは対応するマトリックスを置き換えます。ターゲットセル(マトリックスをドロップするセル)はマトリックスの左のセルの値を受け取ります。他のドラッグされたセルがターゲットセルから右方向および下方向にフィルします。例えば、下のスクリーンショットでは青色のセルがドラッグされ行 6 の `Duration` セルにドラッグされています。これによりマークされたセルが置き換えられます。ドラッグされたセルのマトリックスがテーブルの枠を超えると、列および/または行の適切な数量がすべてのドラッグされたセルに対応するように追加されます。この場合、拡張される枠は点線で表示されます。



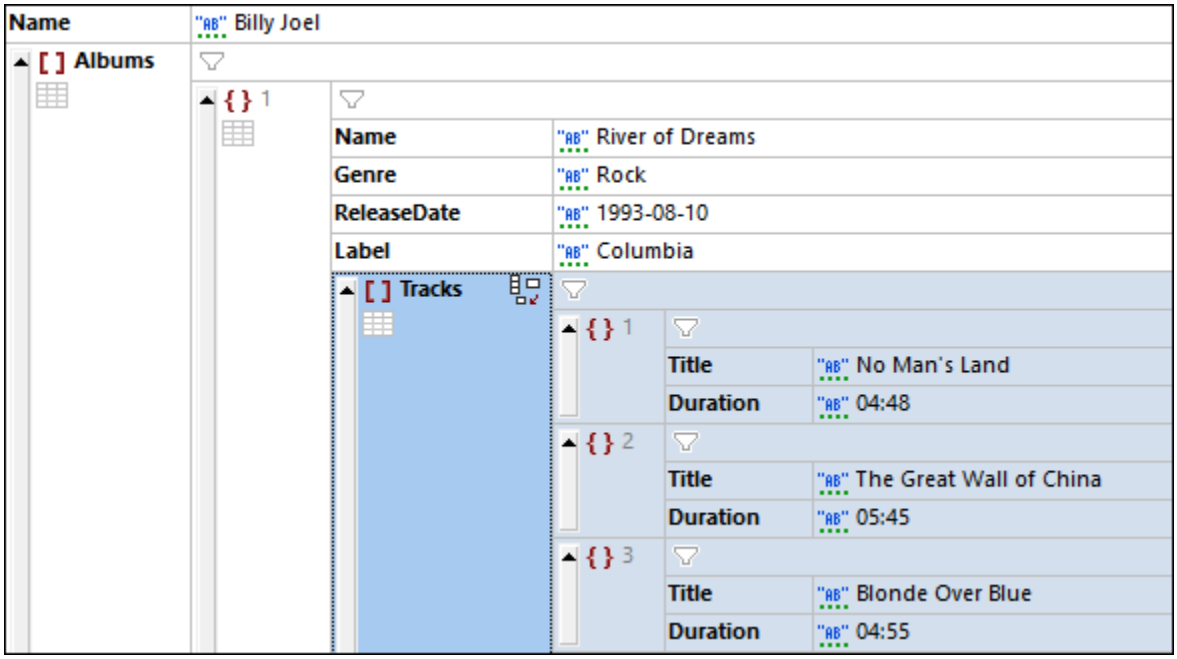
- ターゲットがセルではなくノードツリーの場合、追加されるノードは(i)(角かっこに含まれる)追加されるノード、と(ii)新規のアイテムのインスタンスの数量と共に表示されます。例えば、下のスクリーンショットでは、2つのオブジェクト `Title`、`Duration` と `Writer` キー値ペアを含む配列が追加されます。

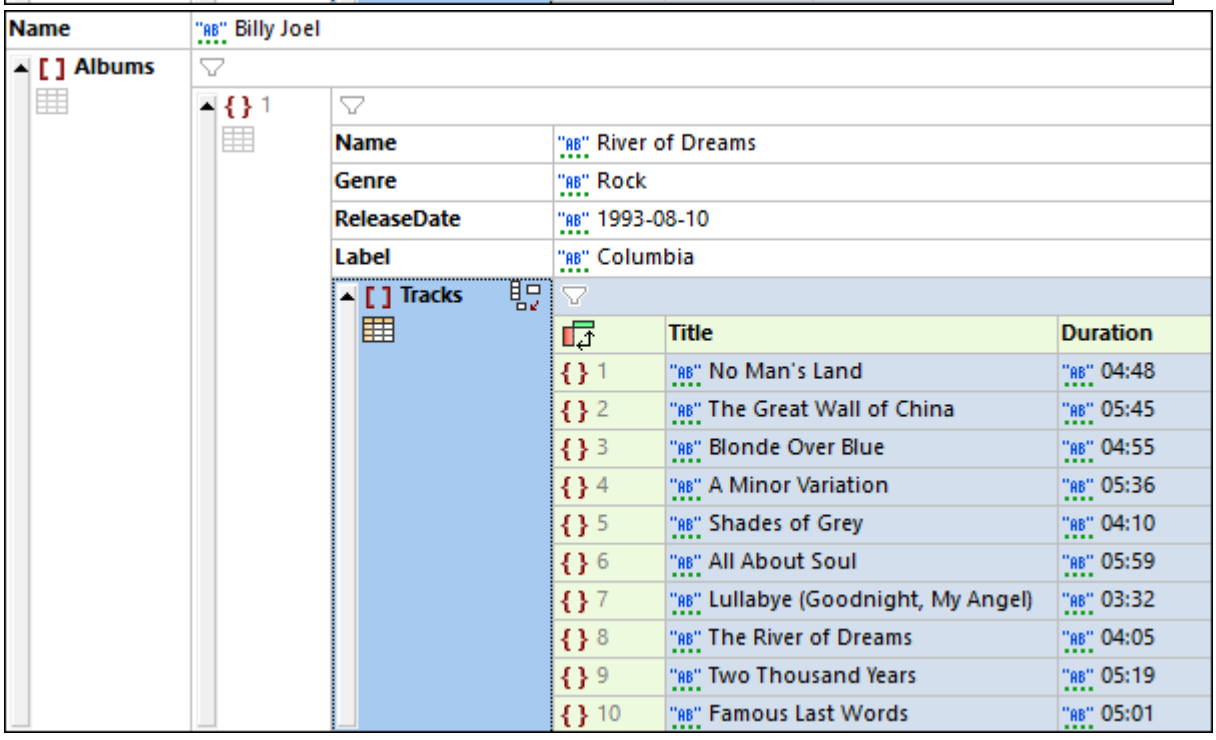
- (ヘッダーをクリックして)選択し、隣接する列ヘッダーにドロップして列を移動することができます。挿入ラインは移動された列が配置されるターゲット列のサイドを示しています。

13.5.5 テーブルの表示

テーブルの表示ビュー

少なくとも1つのオブジェクトまたは配列を含むオブジェクトと配列は(*左側のスクリーンショット内でハイライトされている*)リストまたは(*右側のスクリーンショット内でハイライトされている*)テーブルでビューすることができます。これら2つのビュー間を切り替えるには、オブジェクトまたは配列アイコンの下にあるテーブルアイコンをクリックします(*下のスクリーンショット参照*)。配列テーブル内で、配列アイテムは行として表示されます。例えば、右下のスクリーンショットではトラック配列がテーブルとして表示されています。この配列は行として表示される子オブジェクトにより構成されています。各オブジェクトのプロパティ(下のスクリーンショット内の`Title`と`Duration`)は列のヘッダーとして表示されている共通のプロパティの名前と共に列として表示されています。行と列を切り替えるには、テーブルの左上のセル内のアイコンをクリックします。





テーブル表示内のビューと編集のためのアイコン

下に表示されるアイコンはテーブル表示内で使用可能なアイコンで、ビューと編集機能を提供します。

	セルコンテンツに従いグリッド列の幅を最適化

(C) 2015-2021 Altova GmbH

/	色で表示されている場合 テーブルの表示 613 がオンに切り替えられています。それ以外の場合はオフです。表示するためにクリックします。
/	テーブルの左側のセル内のコマンドをトグルします。行を列に、または列を行に切り替えます。
/	色で表示されている場合、テーブル、オブジェクト、または配列のためのフィルター 617 が有効化されています。それ以外の場合フィルターは無効化されています。無効化/有効化するためにクリックします。式を編集するには、ダブルクリックします。
	セルをワードラップするためのトグルコマンド。選択されると、ワードラップがオンになります。デフォルトでは、アイテムのコンテンツのみがラップされます。アイテムの名前を追加でラップする場合、このオプションをアイコンのドロップダウンリストから選択します。
	配列内: リストまたはテーブルに配列アイテムをふかします。 オブジェクト内: `key:value` ペア(をリスト内で、または(テーブル表示で) 新規のテーブル列のセルとして)付加します。
	`key:value` ペアを追加します。デフォルトでは値の型は `string` です。
	プロパティ値のデータ型を選択、またはコメントまたはフォーミュラを入力します。

テーブル表示内の編集

テーブルの表示では、以下にリストされる編集アクションを実行することができます。

テーブルに行を追加する
以下の方法で行を追加することができます:

- *テーブルに子行を追加します:* テーブルを選択してコンテキストメニュー内でコマンド **子の追加** (Ctrl+Shift+Enter) を選択します。または、テーブルの **子の追加** アイコンをクリックします(*上のアイコンリストを参照*)。新しい行は既存のテーブルの下に付加されます。
- *行が選択された際に兄弟行を追加します:* 行を選択してコンテキストメニュー内でコマンド **追加** (Ctrl+Enter) を選択します。新しい行は既存のテーブルの下に付加されます。
- *選択された行の上に兄弟行を追加します:* 行を選択してコンテキストメニュー内でコマンド **挿入** (Ctrl+Alt+Enter) を選択します。

プロパティの値の入力と編集
プロパティ値が存在するテーブルセルを選択し値を入力します。値が文字列、数値、ブール値または null の場合、データベースは自動的に検知されます。曖昧な場合、型は `文字列` に設定されます。自動選択をデータ型選択アイコンを使用してオーバーライドすることができます(*上のアイコンリストを参照してください*)。または、適切なショートカットを使用することができます(*下のショートカットテーブルを参照してください*)。

*	自動(文字列、番号、ブール値、null を検知し適宜設定します)
"	文字列
[	配列
{	オブジェクト
/	コメント(ドキュメントは JSONC または JSON5 である必要があります)
=	フォーミュラ(ドキュメントは JSONC または JSON5 である必要があります)

新規の列に内に空の行を追加します。
行に新規のセルを追加することができます。新規のセルは新規に作成された列の一部になります。新規のセルを追加するには行を選択して、コンテキストメニュー内で コマンド**子の追加** (Ctrl+Shift+Enter)を選択します。または、行の **行セルの追加** アイコン をクリックします(*上のアイコンリストを参照*)。新規の列内に新規の行セルが作成されます。セル内の値の型はデフォルトで 文字列 です。プロパティ値をセル内に、プロパティの名前を列ヘッダーとして入力します。 新規に作成された列内の他のセルは空です。

選択された列の値のテーブル行を並べ替える
自身の子ノード(列)の一つの相対的な値の上のテーブルの行を並べ替えることができます。例えば、アルファベット順に並べ替えられたテーブルの繰り返し要素を与えるために *LastName* 列をベースに行を並べ替えることができます。列を並べ替えるには、列ヘッダーを選択し、グリッドビューツールバー[602] コマンド **昇順で並べ替え** または **降順で並べ替え** をクリックします。これらのコマンドはJSON メニュー[1242]内でも使用可能です。

テーブル表示と外部アプリケーション

テーブル表示と(MS Excel などの)スプレッドシートアプリケーション間でデータを転送するためにテーブル構造を活用することができます。データの表示からデータを移動するには、以下を行います:

1. テーブルの表示内で、コピーするノードを選択します。セル自身、列ヘッダー、行ヘッダーまたはテーブル全体をクリックしてこれを行います。テーブル全体または列ヘッダーをクリックすると、列ヘッダーもコピーされます。それ以外の場合はコピーされません。(下のスクリーンショットでは行ヘッダーと共に行 1 から 8 が選択されています。)

	Title	Duration	Writer
1	Death on Two Legs	03:43	Freddie Mercury
2	Lazing on a Sunday Afternoon	01:08	Freddie Mercury
3	I'm in Love with My Car	03:05	Roger Taylor
4	You're My Best Friend	02:50	John Deacon
5	'39	03:25	Brian May
6	Sweet Lady	04:01	Brian May
7	Seaside Rendezvous	02:13	Freddie Mercury
8	The Prophet's Song	08:17	Brian May

2. コンテキスト コピー | タブで区切られたテキストとしてコピーする[1150]を選択します。
3. データを直接スプレッドシートプログラムに張り付けます。

	A	B	C
1	Title	Duration	Writer
2	Death on Two Legs	3:43	Freddie Mercury
3	Lazing on a Sunday Afternoon	1:08	Freddie Mercury
4	I'm in Love with My Car	3:05	Roger Taylor
5	You're My Best Friend	2:50	John Deacon
6	'39	3:25	Brian May
7	Sweet Lady	4:01	Brian May
8	Seaside Rendezvous	2:13	Freddie Mercury
9	The Prophet's Song	8:17	Brian May

データの転送は両方向に行うことができます。スプレッドシートからテーブルの表示にデータをコピーすることができます。以下の手順を行ってください:

1. 外部アプリケーション内で範囲を選択して（**Ctrl+C** を使用して）クリップボードにコピーします
2. テーブルの表示内で単一のセルを選択します。
3. コピーされたデータを **Ctrl+V** を使用して貼り付けます。

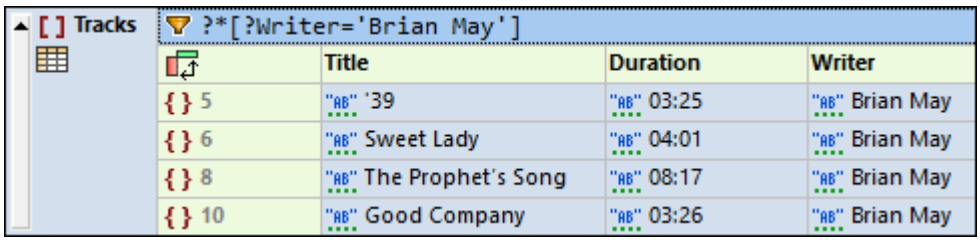
データは元の構造に対応し、テーブルビュー内で選択されたセルから開始する構造を持つテーブルに張り付けられます。以下の点に注意してください。

- 既存のデータを無効化することなく新規の行および/または列の作成を許可するセルにデータがコピーされていると、新規の行および/または列が作成されます。
- 列はテーブル（行）のオブジェクトのプロパティとして作成されますが、行は新規のオブジェクトとして作成されます。
- テーブル構造が有効に変更できない場合、新規のデータはテキストとして選択されたセルの元のデータを上書きします。

更に複雑なデータ転送タスクのために、XMLSpy のユニークな インポート/エクスポート 1331 関数を使用してください。

13.5.6　フィルター

フィルターはオブジェクトと配列をフィルターし表示することを可能にします。例えば、下のスクリーンショットでは（XQuery 3.1 式である）フィルターが配列に適用され、Brian May により作成されたトラックのみが表示されます。JSON では、文字列値 `Brian May` を含む `Writer` プロパティを持つ `Tracks` のオブジェクト子のみが表示されます。）フィルターの XQuery 式は `Tracks` のすべての子オブジェクトをルックアップし、文字列 `'Brian May'` に一致する `Writer` プロパティをルックアップするものを選択します。



各フィルターは個別に実行されドキュメント内の他のフィルターまたはフィルターにより影響されません。

JSON のための XQuery 式 ドキュメントの作成に関する詳細は、JSON のための XQuery 式 668 を参照してください。

メモ　グリッドビュー内にフィルターとフォーミュラのために式を入力する際に、読みやすいように式を複数行で表示するようにすることが奨励されます。式内に新規のラインを追加するには、「Ctrl+Enter」を押します。

フィルターのセットアップ

フィルターをセットアップするには、フィルターする要素を右クリックして **フィルター** を表示されるコンテキストメニューから選択し Xquery 式を入力して **入力** をクリックします。*フィルターセル*はアイコン により示されています（*上のスクリーンショットを参照*）。*フィルターアイコン*をクリックしてフィルターを有効化/無効化することができます。

フィルターに関する以下の点に注意してください：

- オブジェクトと配列のみに適用することができます。
- フィルターの XQuery 式のコンテキストノードは現在のノードです。上のスクリーンショットでは、例えば、フィルターの XQuery 式のコンテキストノードは `Tracks` ノードです。

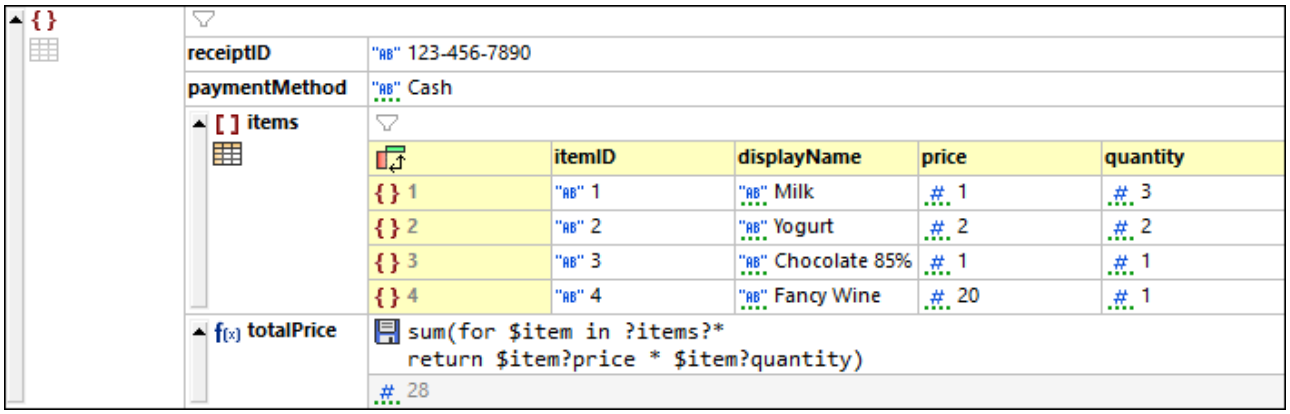
- フィルターはJSON ドキュメント内でJSON コンテンツとしては保管されませんが、次回の使用のために特別なメタデータとして保管することができます。フィルターはドキュメントのメタデータとして保管され、JSON グリッドビュー内でドキュメントが開かれると対応するオブジェクトや配列に適用されます。
- フィルターをネストすることが可能です。ネストされたフィルターは親フィルターのフィルターされたコンテンツに適用することができます。
- 式内に新規のラインを追加するには、「Ctrl+Enter」を押します。

13.5.7 フォーミュラ

フォーミュラはJSON グリッドビュー内の表示のために出力（ノードセットまたは計算）を生成するXQuery 3.1 式です。下のスクリーンショットでは、例えば、アイテム1 から4 の総額が計算され、出力（28）が個別のラインで表示されます。各フォーミュラは個別に実行され他のフィルターまたはフィルターにより影響されません。

JSON のためのXQuery 式ドキュメントの作成に関する詳細は、JSON のためのXQuery 式 668 を参照してください。

メモ　グリッドビュー内にフィルターとフォーミュラのために式を入力する際に、読みやすいように式を複数行で表示するようにすることが奨励されます。式内に新規のラインを追加するには、「Ctrl+Enter」を押します。



ドキュメントがJSON5 またはJSONC ドキュメントである場合:

- フォーミュラはJSON コメントとしてドキュメントに書き込まれます。
- 任意で、フォーミュラの出力はJSON ドキュメント内の（コメントを持たない）コードとして保管されることができます。これを行うには、フォーミュラのディスクアイコンをクリックします（*下のスクリーンショットを参照*）。JSON コンテンツとしてフォーミュラが保管されない場合、グリッドビュ内で表示されるドキュメントメタデータとして保管されます。JSON5 とJSONC ドキュメント内でのみでディスクアイコンを使用することができ、JSON ドキュメントの他の型では表示されません。

保管されたフォーミュラとフォーミュラの結果を含む上のスクリーンショット内で示されているJSON ドキュメント

注意点：フーミュラはJSON コメントとして保管されますが、フォーミュラの結果はストレートなJSON コードとして保管されます。下の（フォーミュラの結果が保存された）コードはクリックされたフォーミュラのディスクアイコンの結果です

```
{
	"receiptID": "123-456-7890",
	"paymentMethod": "Cash",
	"items": [
		{
			"itemID": "1",
			"displayName": "Milk",
			"price": 1,
```

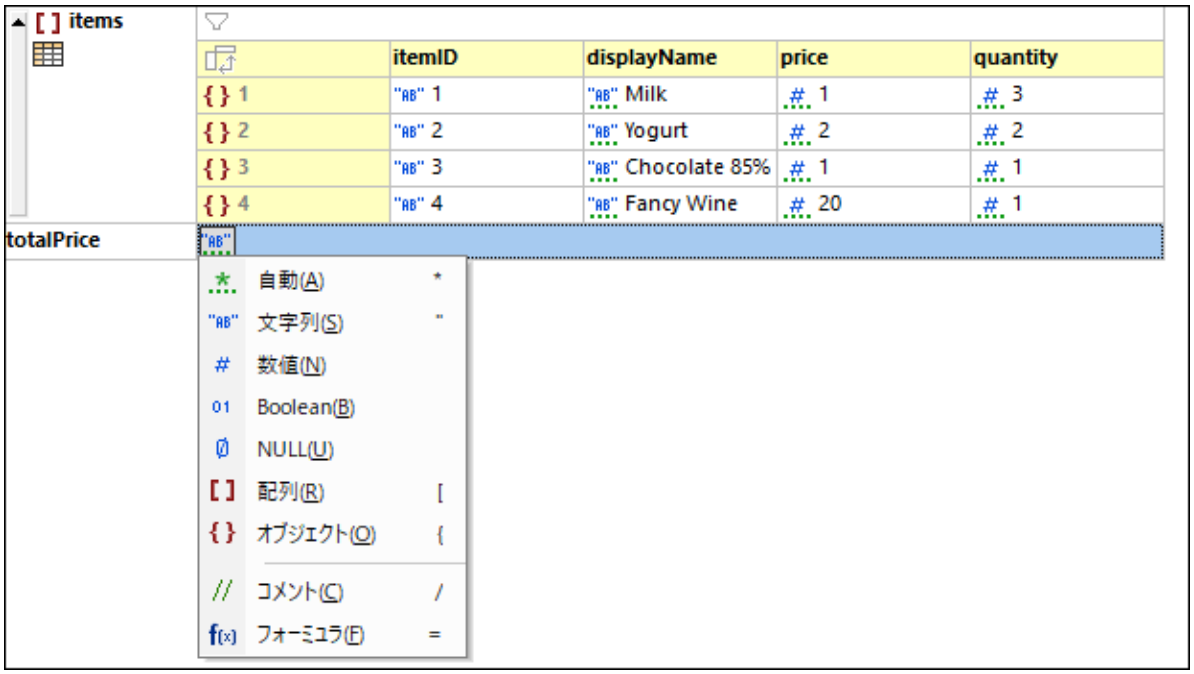
```
                "quantity": 3
            },
            {
                "itemID": "2",
                "displayName": "Yogurt",
                "price": 2,
                "quantity": 2
            },
            {
                "itemID": "3",
                "displayName": "Chocolate 85%",
                "price": 1,
                "quantity": 1
            },
            {
                "itemID": "4",
                "displayName": "Fancy Wine",
                "price": 20,
                "quantity": 1
            }
        ],
        //(:altova_xq_embed:)totalPrice(:altova_xq_key:)sum(for $item in ?items?*
return $item?price * $item?quantity)
        "totalPrice": 28
        //(:altova_xq_end:)
}
```

上のスクリーンショットでは、フォーミュラはシーケンスのメンバーの合計を計算します。メンバーとは **items** 配列内に含まれる各オブジェクトの **price** と **quantity** 値の製品のそれぞれです。各オブジェクトを選択し、**$item** 変数に割り当てるための反復は以下により指定されます: **for $item in ?items?*** 。フォーミュラの親である、したがって **items** ノードの親である、コンテキストノードをメモすることはこの時点で重要です。各製品は **$item** 変数内に現在存在するオブジェクトの **price** と **quantity** 子ノードをルックアップし、これらの値を乗算することにより取得されます。この方法で取得された製品は総額を生成するために合計されたシーケンスのメンバーです。

フォーミュラの作成

フォーミュラを作成するには、以下を行います:

1. フォーミュラに兄弟または子として追加するノードを選択します。右クリックし、兄弟または子を追加します。フォーミュラが(兄弟または子として) 表示される個所を決定する場合、フォーミュラの XQuery 式のコンテキストノードはフォーミュラの親ノードになります。例えば、上のスクリーンショット内で表示される XQuery 式内では、コンテキストノードはフォーミュラ(**totalPrice**) とその兄弟、 **items** 配列の親ノードです。上のスクリーンショット内でフォーミュラを作成する場合、兄弟ノードは **items** 配列に追加されます *(下のスクリーンショットを参照)*。
2. フォーミュラに対するノードの型を変更します *(下のスクリーンショットを参照)*。



3. `f(x)` アイコンを含むセルをダブルクリックし、フォーミュラの名前を入力します *(下のスクリーンショットを参照)*。ドキュメントが JSON5 または JSONC ドキュメントの場合、ディスクアイコンが表示されます。このアイコンをクリックして、フォーミュラの出力をドキュメントに保存することができます。

4. デフォルトでは、XQuery 式は文字列 `'XQuery'` ですが、出力は(式の下のセル内に表示されている)文字列 `XQuery` です。XQuery 式を式のセルをダブルクリックして入力し、式を入力し、「Enter」をクリックします。これによりフォーミュラが評価されその結果が表示されます。他のフォーミュラは変更されません。
5. ドキュメント内に複数のフォーミュラが存在する場合メニューコマンド「JSON | 全てを再評価する」をクリックして全てのフォーミュラの結果を更新します。ドキュメント内のフォーミュラは(例えば、為替などの)動的に変更するデータをルックアップする場合このコマンドは特に役に立ちます。

フォーミュラに関する以下の点に注意してください:

- フォーミュラの XQuery 式のコンテキストノードはフォーミュラノードの親ノードです。
- フォーミュラはドキュメントの(特別なアプリケーション全体の JSON メタデータファイル内にある)メタデータ内で保管されます。ドキュメントが JSON5 または JSONC の場合、フォーミュラは追加で JSON コンテンツ内のコメントとして保管されます。
- フォーミュラにより生成された出力はフォーミュラの XQuery 式の下のセル内に表示されます。JSON5 と JSONC の場合、XQuery 式の横のディスクアイコンをクリックして出力をドキュメント内に保管することができます。ディスクがクリックされているかに関わらず、フォーミュラの出力は計算されドキュメントのメタデータに保管されます。

	JSON コンテンツに保存されていないフォーミュラ出力。クリックして保存します。*JSON5 と JSONC 内のみ。*
	JSON コンテンツに保存されているフォーミュラ出力。*JSON5 と JSONC 内のみ。*

- 出力が計算でコンテンツに保管されている場合、フォーミュラに割り当てた名前を持つプロパティとして保管されます。例えば、上記の `totalPrice` フォーミュラ内では、出力は以下のようになります: `"totalPrice": 28`.
- 以下の違いに注意してください: JSON5 と JSONC ドキュメント内では、JSON コメントとしてフォーミュラは保存されます。

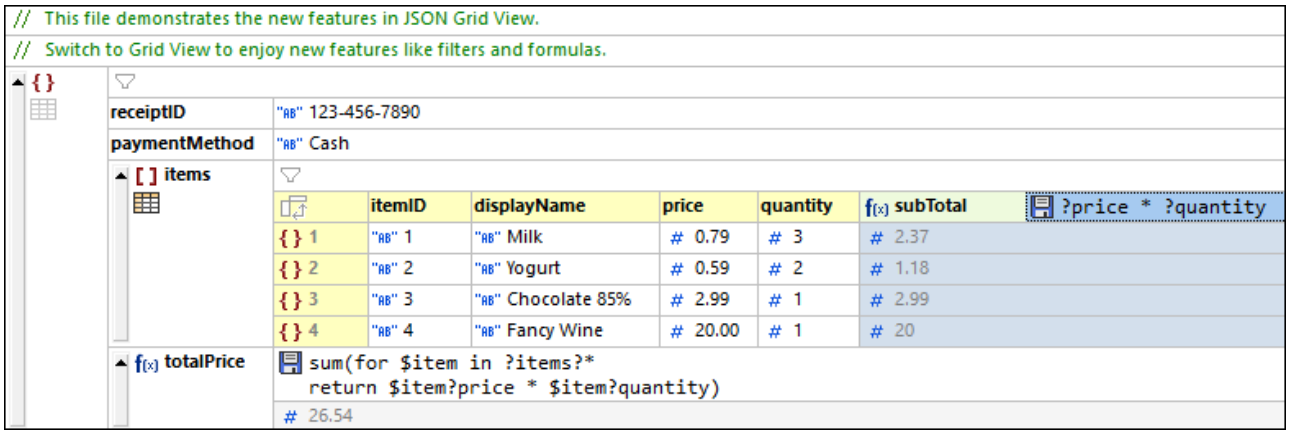
メモ JSON グリッドビュー設定ダイアログ[627] によりフォーミュラがドキュメント内に保管されるかを指定することができます。フォーミュラを保管するオプションはデフォルトで選択されます。

メモ JSON グリッドビュー設定ダイアログ[627] 内で以下を行うことができます：(i) (メタデータファイル内に自動的にフォーミュラを保管するか、または保管しないかなどの) メタデータオプション

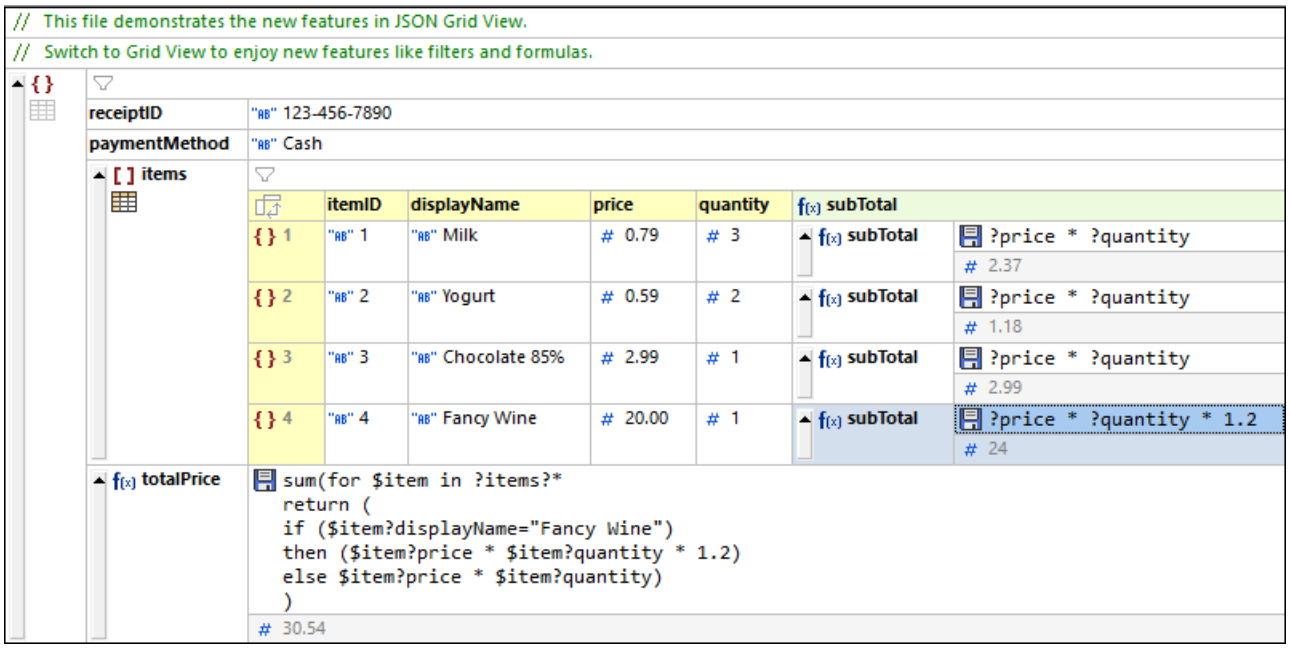
テーブル内のフォーミュラ

(テーブルディスプレイ[613]) 内のテーブル列のすべてのセルに同じフォーミュラが含まれている場合、フォーミュラは列のヘッダー内で一度のみ表示されます (*下のスクリーンショットを参照*)。フォーミュラ計算の*結果*はしかしながら対応するセル内に表示されます。

列ヘッダー内のフォーミュラはグリッドビュー表示です。(テキストビュー内の) JSON ドキュメントコンテンツ 内では、フォーミュラは各テーブル行アイテムのために繰り返されています。



(下のスクリーンショットのハイライトされたセル同様) 単一のフォーミュラが異なる場合でも各フォーミュラは対応するセル内に表示されます。



全てのフォーミュラの再評価
ドキュメント内のすべてのフォーミュラの結果を更新するには、メニューコマンド「JSON | 全てを再評価する」をクリックします。

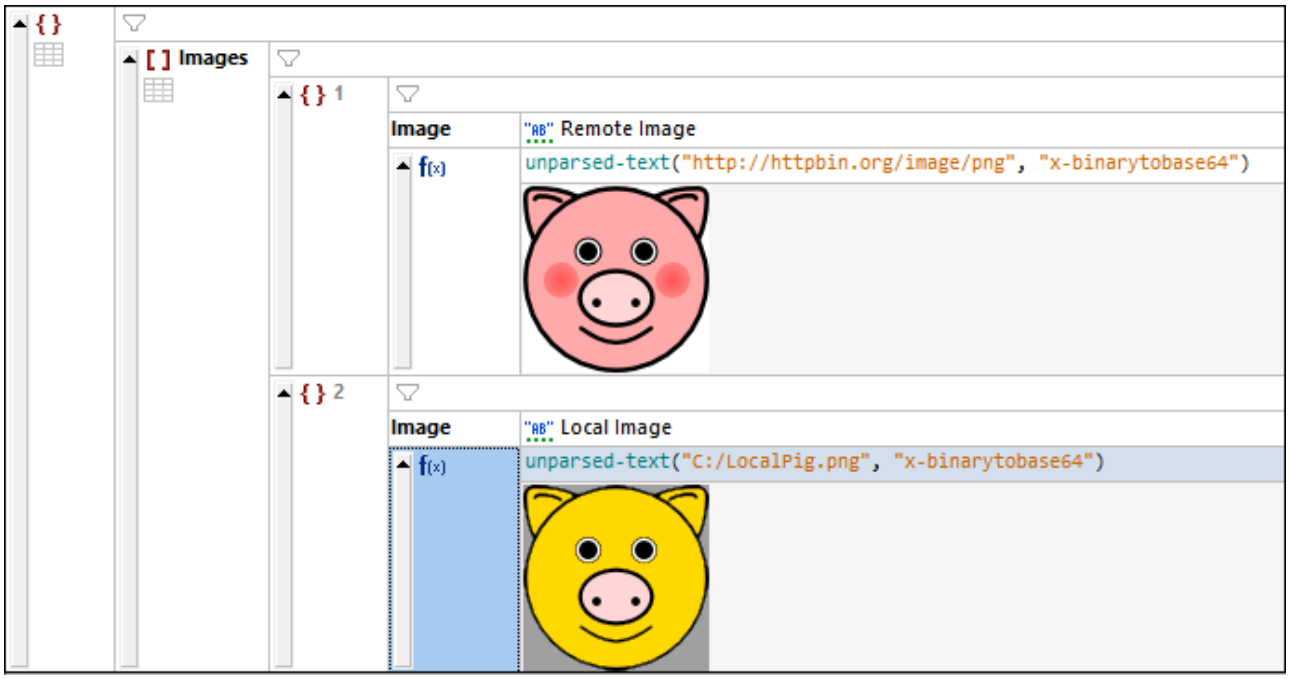
関連したアクションのためのプロシージャ

- 以下のようにテーブルに新規の空の列を追加します: テーブルディスプレイからリスト表示に切り替え、リスト表示内の `key:value` ペアのいずれかをクリックし、新規の `key:value` ペアをアイテムのコンテキストメニューを使用して追加します。テーブルディスプレイに切り替える場合、新規の列が追加または挿入された新規の `key:value` ペアのために作成されます。グリッドビューでこの列を編集することができます。
- フォーミュラがヘッダーに表示されるようにテーブル列のすべてのフォーミュラが同じの場合、および個別のセルのために異なるフォーミュラを作成する場合、リスト表示に切り替え、セルのフォーミュラを編集します。テーブルディスプレイに切り替えると、フォーミュラは全てのセルのために個別のセルのために表示されます。
- 式内に新規のラインを追加するには、「Ctrl+Enter」を押します。

13.5.8 イメージ

イメージを JSON グリッドビューで表示することができます *(下のスクリーンショット参照)*。イメージは Base64-エンコード済みイメージに自動的に変換され表示されます。

イメージをロードするには式を使用する際に以下の式を入力します: **`unparsed-text("<Image-URL>", "x-binarytobase64")`**



URL を相対的パスを含む以下の書式で提供することができます:

- `http://httpbin.org/image/png`
- `file:///c:/LocalPig.png`
- `C:/LocalPig.png`
- `LocalPig.png`

13.5.9 チャート

Altovaext:ahart という名前の Altova XPath/XQuery 拡張子を使用して JSON グリッドビュー内でチャートを作成することができます（*下のスクリーンショットを参照*）。拡張子は下で説明されています。これに関しては[チャート関数](2053)のセクション内で他のチャート共に説明されています。下のスクリーンショットで表示されている **altovaext:chart** 拡張子は[JSON グリッドビューフォーミュラ](618)内で定義されている XQuery `Let` 式内で使用されています。チャートはフォーミュラの下でイメージとして表示されています。（このチャートサンプルは[ドキュメント](33)の *Examples フォルダー*内にあるファイル **Chart.jsonc** 内にあります。または、[*サンプルプロジェクト*](110)からアクセスすることもできます。）

{ }

[ ] Temperatures

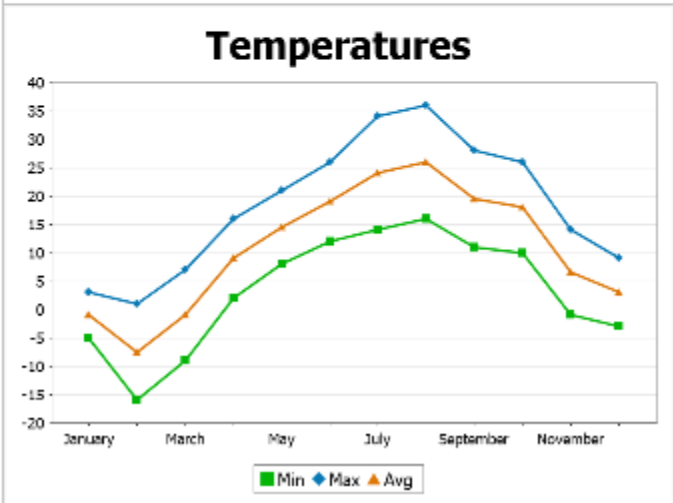
	Month	Min	Max
{ } 1	"AB" January	# -5	# 3
{ } 2	"AB" February	# -16	# 1
{ } 3	"AB" March	# -9	# 7
{ } 4	"AB" April	# 2	# 16
{ } 5	"AB" May	# 8	# 21
{ } 6	"AB" June	# 12	# 26
{ } 7	"AB" July	# 14	# 34
{ } 8	"AB" August	# 16	# 36
{ } 9	"AB" September	# 11	# 28
{ } 10	"AB" October	# 10	# 26
{ } 11	"AB" November	# -1	# 14
{ } 12	"AB" December	# -3	# 9

{ } ChartConfig

// Title is optional, you can remove it	
title	"AB" Temperatures
// Try modifying the kind of this chart.	
kind	"AB" LineChart
width	# 800
height	# 600

f(x) Chart

```
let $temps := ?Temperatures?* return
altovaext:chart(?ChartConfig,
(
    (:name, X-axis,       Y-axis   :)
    ['Min', $temps?Month, $temps?Min],
    ['Max', $temps?Month, $temps?Max],
    (: Calculate average per each min/max
       using mapping operator ! :)
    ['Avg', $temps?Month, $temps ! avg((?Min, ?Max))]
))
```



(C) 2015-2021 Altova GmbH

上のスクリーンショット内で表示されているチャートを生成するための XQuery 式のリスティング

```
let $temps := ?Temperatures?* return
altovaext:chart(?ChartConfig,
(
    (:name, X-axis,       Y-axis   :)
    ['Min', $temps?Month, $temps?Min],
    ['Max', $temps?Month, $temps?Max],
    (: Calculate average per each min/max
       using mapping operator ! :)
    ['Avg', $temps?Month, $temps ! avg((?Min, ?Max))]
))
```

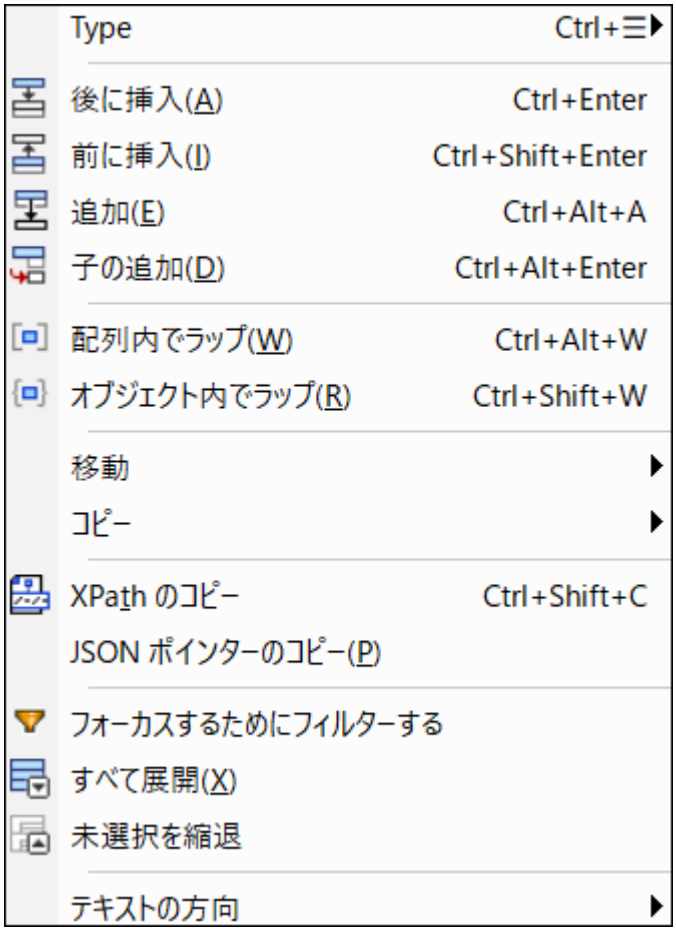
Altova チャート拡張子の使用

- チャート拡張関数 `altovaext:chart` は名前空間プレフィックス `altovaext:` を使用する必要があります。
- `altovaext:chart` は2個の引数を取ります: (I) チャート構成情報と(ii) チャートデータ系列情報。
- チャート構成情報は `altovaext:chart` の最初の引数です。4個のキー値ペアの順序指定されていない系列です。これらのペアは以下のために存在します: (i) チャートのタイトル(キーは `title`)、(ii) 円グラフ、折れ線グラフなどのチャートの種類(キーは `kind` 使用可能な種類に関してはチャート関数 2053 を参照してください。)、(iii) ピクセルで表示されたチャートの幅(整数のみ。キーは `width` です)。および(iv) ピクセルで表示されたチャートの高さ(整数のみ。キーは `height` です)。上のサンプルのスクリーンショット内では `altovaext:chart` 関数内で参照されている `ChartConfig` オブジェクト内に構成情報が保管されています。
- チャートデータ系列は `altovaext:chart` の2番目の引数です。各データ系列はサイズ3の配列です: (I) 系列の名前 (ii) x-軸の値 (ii) y-軸の値(例えば、上のサンプルのように各系列がラインを表示する場合など) 複数の系列を作成する場合は複数の配列のシーケンスを作成します。上のサンプルには最低、最高、および平均機能のための3個のデータ系列が存在します。XとY 軸のためのデータは `Temperatures` という名前の配列から参照されます。

詳細に関しては、チャート関数 2053 のセクションを参照してください。

13.5.10 グリッドビュー内のコンテキストメニュー

JSON グリッドビュー内でコンポーネントを右クリックすると、ドキュメントの表示と編集を変更するためのコマンド表示するコンテキストメニューが表示されます (*下のスクリーンショット*)。コンテキストメニューはマイナスキーを押してアクセスすることができます。コンテキストメニューのコマンドは下で説明されています。



型
型 コマンドにマウスをポイントするとコンポーネントの型[607] を選択することのできる表示されるサブメニューを表示します。「Ctrl+キーメニュー」を使用して型サブメニューにアクセスすることができます。

子を前/後に挿入、子の付加、子の追加
「**挿入**」と「**追加**」コマンドにより同じレベルでアイテムを追加することができます。**後に挿入** と **前に挿入** の場合新規のアイテムがそれぞれ選択されたアイテムの前後に追加されます。**付加** の場合、選択されたアイテムの最後の兄弟として追加されます。「**子の追加**」コマンドは新規のアイテムを子として追加します。1つまたは複数の子アイテムが既存の場合、新規アイテムの型は最後の子と同じになります。子が存在しない場合、新規のアイテムは空の `key:value` ペアになります。例えば、選択されたアイテムがオブジェクト子を持つ配列の場合、新規の子アイテムは最後の子オブジェクトとして追加されます。

配列内でのラップ、オブジェクト内でのラップ
テーブルの選択されている部分は配列またはオブジェクト内にラップされます。

上/下/左/右への移動
コンポーネントを現在の位置からグリッド内で上下左右に移動しすることが可能になり、対応するコマンドが有効化されます。移動を実行するために対応するコマンド選択してください。

コピー
これらのコマンドは現在の選択項目をクリップボードにコピーするために使用されます。そのコンポーネントに許可されているオプションのみを使用することができます。

コマンド	説明
JSON テキストとしてコピーする	現在の選択範囲はJSON マークアップとしてシリアル化されています。
タブで区切られたテキストとしてコピー	現在のテーブル選択項目がTSV（タブ区切り値）としてシリアル化されます。
イメージとしてコピーする	現在のイメージセルがイメージとしてコピーされます

以下の点に注意してください：

- *イメージとしてコピー*は選択されたイメージのBase64 エンコード文字列をコピーします。（例えば、テーブルの表示内の他のセルなど）Base64-エンコード済み文字列がイメージとして表示されるドキュメントに文字列が張り付けられると文字列は標示されます。それ以外の場合は文字列として解析されます。
- テキストをセルに挿入するには、テキストをコピーしてセルに張り付けます。
- イメージをファイルからインポートするには、[編集 | 挿入 | エンコードされた外部ファイル]1154コマンドを使用します。Base64 エンコード文字列としてイメージファイルを挿入し、この文字列をJSON グリッドビューで表示します。インポートのためにサポートされているイメージファイルフォーマットは以下の通りです: PNG、JPEG、BMP、GIF、TIFF。

XPath のコピー
選択されたノードをロケートするドキュメントルートから開始するPath 3.1 ロケーター式をクリップボードにコピーします。

JSON ポインターをコピーする
このコマンドは選択されたノードをロケートするJSON Pointer 式をクリップボードにコピーします。例：`/Artists/1/Albums/1/Tracks`. 詳細に関してはXPointer/JSON-Pointer のコピー1154 を参照してください。

フィルター、フォーカスするフィルター
フィルター コマンドは切り替えコマンドです。選択されたオブジェクトにフィルターを追加または*フィルター*を無効化にします。このコマンドを使用してフィルターを追加後、フィルター式を入力する必要があることに注意してください。それ以外の場合フィルターは自動的に削除されます。フィルターが追加された後、コマンドをもう一度選択またはフィルターをクリックして無効化することができます。フィルターを削除するにはツールバー内で *全てのフィルターを削除する*をクリックします。

フォーカスするフィルター コマンドは（i）選択されたノードをロケートするXPath 式を含むルート要素上にフィルターを作成します。そして（ii）選択されたノード（とその子孫）を直接ルート要素の下で表示するためにグリッドビューをフィルターします。選択されたノードの表示のみに集中する場合この機能は役にたちます。フィルターを削除するには、ルート要素のコンテンツセル内のフィルターアイコンをクリックします。

フルに展開する
このコマンドは選択されているコンポーネントまたはその子孫が折りたたまれている場合に有効化されます。コンポーネントと子孫コンポーネントが展開されます。

選択されていない範囲を折りたたむ
このコマンドは選択されているコンポーネントとその祖先コンポーネント以外のすべてのコンポーネントを折りたたみます。

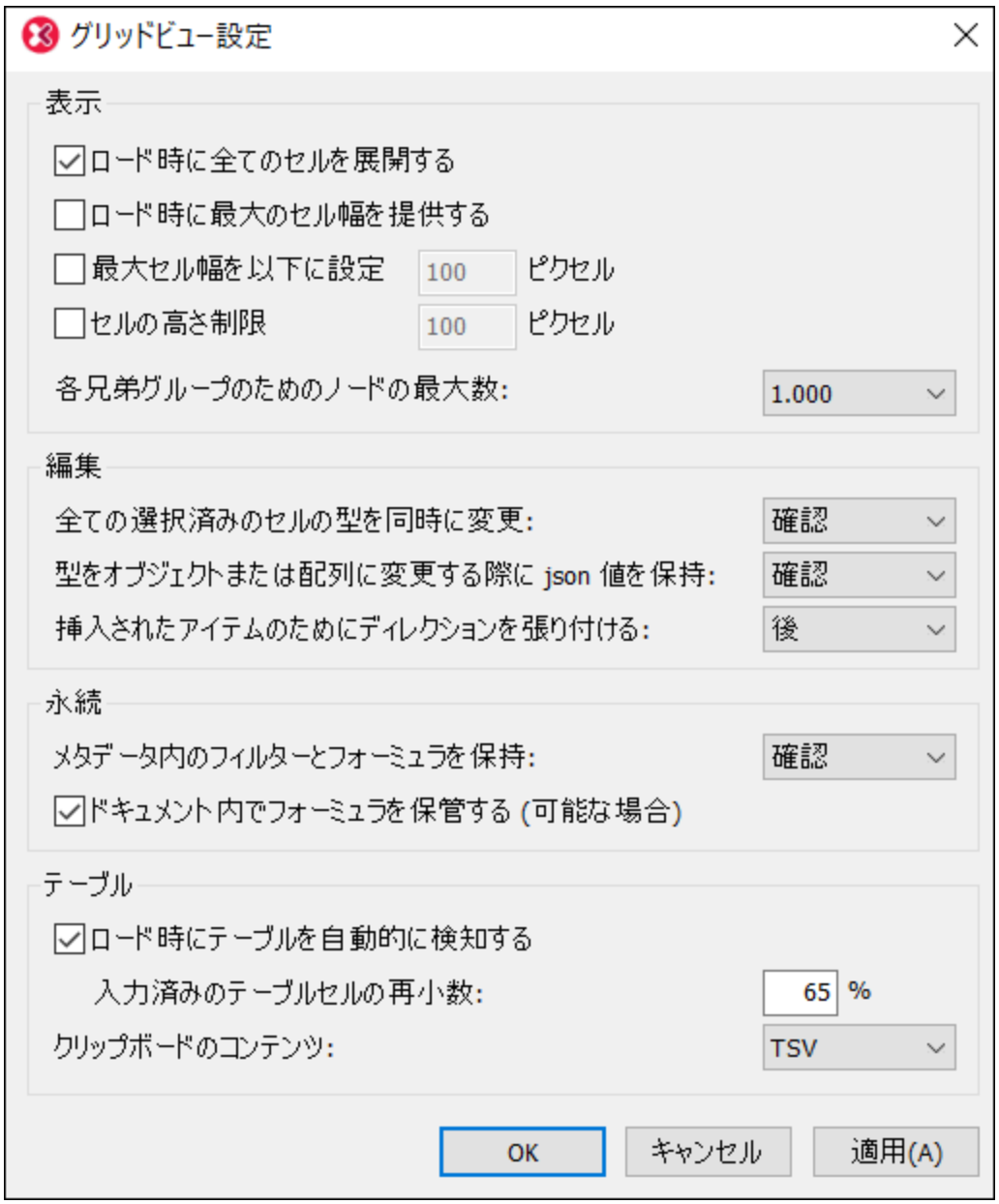
テキストの向き
このコマンドはString 型を含むノードのために使用することができ、読み取りの順序をセルの左側方向または右側方向に切り替えます。この機能はアラブ語とヘブライ語などの言語を使用する際に役に立ちます。

13.5.11 グリッドビューの設定

グリッドビューの設定はグリッドビュー設定ダイアログ（下のスクリーンショット）はグリッドビューツールバー602内の設定アイコンを使用してアクセスすることができます。

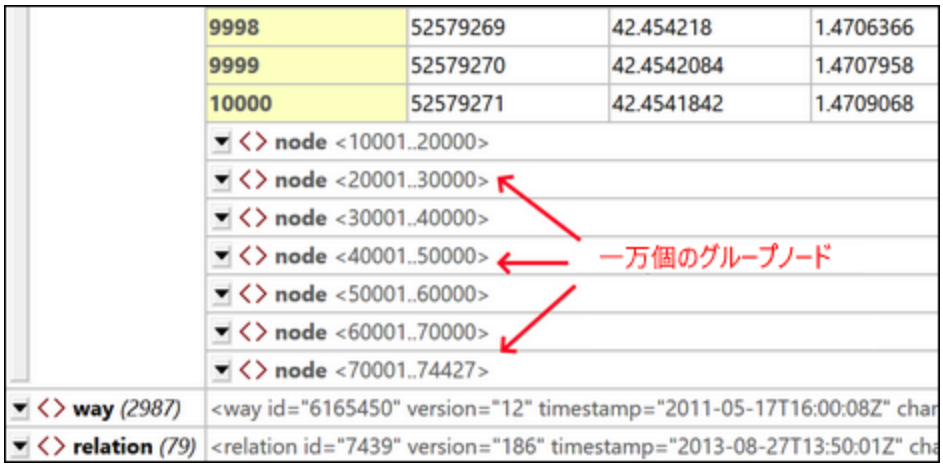
	グリッドビューツールバー602内のグリッドビュー設定アイコン

グリッドビューの設定は以下で説明されています。全てのドキュメント（XML、JSON、DTD）のグリッドビューに適用することができることに注意してください。



表示
表示 セクション内のチェックボックスは自明です。以下は更に説明が必要な場合のメモです。

- ロードする際に全てのセルが展開されていない場合ルートノードとそのすべての子孫は折りたたまれています。ドキュメントを更に深く移動するためには各ノードを展開する必要があります。
- 最適な幅がオンに切り替えられている場合グリッドの幅全体が表示されます。これを達成するには、セルの一部のテキストがラップされます。
- 兄弟ノードは100、1000 または10000 ノードの兄弟グループで表示することができます（*下のスクリーンショットを参照してください*）。多数の兄弟ノードを持つファイル内でスペースの表示を省略する場合



編集

表示 セクション内のチェックボックスは自明です。以下は更に説明が必要な場合のメモです。

- 複数の選択されたセルの型を変更する場合、設定のアクションを継続するかを決定する以下のオプションが与えられます: (ユーザーにより決定される) *常に*、*なし* または *確認* 。
- JSON 型を動的な型からオブジェクトまたは列挙に変更する場合、設定のアクションを継続するかを決定するかを決定する以下のオプションが与えられます: (i) *確認* (動的な型の値が名前を持たない子 `key:value` ペアの値として保持される、または破棄されるかを確認) (ii) *常に* (名前を持たない子 `key:value` ペア内の値を保持する) (iii) *なし* (値を保持する。すなわち値を破棄する)。
- *ディレクションの貼り付け* オプションはクリップボード内の選択範囲が選択されたセルの上または下に張り付けられるかを決定します。

永続

フィルターとフォーミュラを XML と JSON ドキュメントのメタデータ内に保管することができます。フォーミュラは XML ドキュメント内の処理命令として、そして JSON ドキュメント内のコメントとして保管されます。これらの設定はフィルターとフォーミュラの保管の方法を指定します。フォーミュラの場合 *可能な場合* という用語はフォーミュラが実際にすべての JSON ドキュメント (JSON5 と JSONC ドキュメント内でのみ) のコメントとして保存できないことを意味します 。

テーブル

以下の設定を使用することができます:

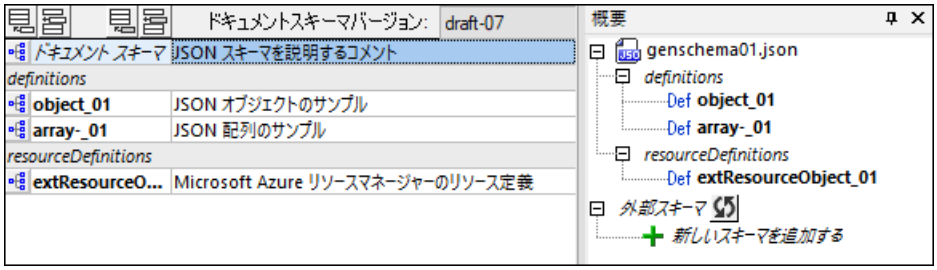
- 設定がロード時に JSON グリッドビューテーブルを自動的に検知する場合、テーブルをテーブルとして検知するために評価する記入されたテーブルセルの最小のパーセントを選択することができます。記入されたテーブルセルの数値がこのレベルを越さない場合、構造はそれぞれが下に表示されている繰り返し要素を持つ通常のグリッドとして表示されます。JSON グリッドビューのみにこの設定は適用されることに注意してください。XML グリッドビュー内では、テーブルセルの記入されている率に関わらずテーブルは自動的に検知されます。
- クリップボードコンテンツがタブにより区切られた値 (TSV) または (ドキュメントの型に従い) XML/JSON として保管するかを選択することが可能です。これはとても役に立つ機能です: クリップボードからテーブルを他のドキュメントに張り付ける場合、この設定により TSV としてまたはマークアップとしてテーブルがコピーされるかを選択できるようになります。(違いを確認するには、テーブルをそれぞれのフォーマットでクリップボードにコピーしテキストエディターにテーブルを張り付けてください。)

13.5.12 JSON スキーマビュー

JSON スキーマビューは、JSON スキーマドキュメントを閲覧し編集するために使用することができます。JSON スキーマビューウィンドウの主な部分は以下のとおりです:

- 定義概要グリッド[632] とデザインビュー[638] 間を切り替えるメインウィンドウ
- 3 つの入力ヘルパーウィンドウ[634] (メインウィンドウの右側にデフォルトで表示されます): 概要、詳細、制約
- メッセージウィンドウ (メインウィンドウの下にデフォルトで表示されます)
- 情報ウィンドウ[662] (アプリケーションウィンドウの左側にデフォルトで表示されます)

下のスクリーンショットは、メインウィンドウおよび概要入力ヘルパーを表示しています。

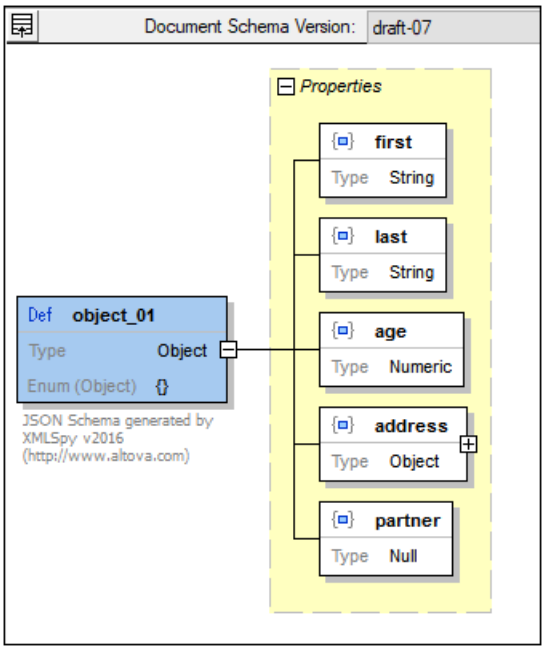


メインウィンドウ

メインウィンドウは、定義概要グリッド[632] (*上のスクリーンショットに表示*) とデザインビュー[638] (*下のスクリーンショット*) を切り替えます。定義概要グリッドは、現在のドキュメントの (*"ドキュメントスキーマ"としてリストされている*) メインスキーマを表示し、またスキーマに追加する定義を表示します。(定義は、JSON データ構造の詳細です。上のスクリーンショットでは、`object_01` および `array_01` がそれぞれオブジェクトと配列の定義です。) 定義は、概要入力ヘルパーでもリストされています (*上のスクリーンショット参照*)。

定義概要グリッドは、JSON スキーマのハイレベルな表示を行いますが、概要内にリストされている定義の内容を表示しません。デザインビュー内の定義を確認して編集するには (*下のスクリーンショット*)、定義のアイコン (*上のスクリーンショット参照*) をクリックするか、または概要入力ヘルパー内の定義をダブルクリックします (*上のスクリーンショット参照*)。

(C) 2015-2021 Altova GmbH



デザインビューから定義概要グリッドに切り替えるには、デザインビューの左上にある「**定義グリッドに切り替え**」アイコンをクリックします（*上のスクリーンショット参照*）。デザインビューを構成するには、メニューコマンド スキーマデザイン | 構成ビュー[657] をクリックします。

入力ヘルパー

スキーマビューの2つのモード（定義概要グリッド とデザインビュー）には、3 つの入力ヘルパーがあります: 概要、詳細、制約。これらの入力ヘルパーは以下のためのメカニズムが搭載されています: (i) スキーマとその定義に関する情報を表示 (ii) 定義に関連する情報と値の入力。詳細に関しては、入力ヘルパー: 概要、詳細、制約[634]を参照してください。

13.5.12.1 JSON スキーマバージョン

JSON スキーマは、JSON 構文で書かれており、`.json` 拡張子を持つファイルに通常保存されています。以下の値を持つトップレベルのオブジェクトの最初のキーワードである **`$schema`** キーワードにより JSON スキーマとして表示されています: **`"http://json-schema.org/draft-N/schema#"`** または **`"http://json-schema.org/schema#"`**.

```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
   ...
}
```

or

```
{
   "$schema": "http://json-schema.org/schema#",
   ...
}
```

上記の最初のサンプルでは、スキーマバージョンは **`draft-04`**、**`draft-06`**、などの書式を持つバージョン名と共に明示的に名前が付けられています。上の2番目のサンプルの通り、バージョン名が省略されている場合、これは使用されるスキーマバージョンは最新のバージョンであることを示しています（現在は **`<%JSON-SCHEMA-LATEST%>`**）。

JSON スキーマビュー内で、メインウィンドウの上のバー内のコンボボックス内でバージョンを変更することができます（*下のスクリーンショットを参照してください*）。

他の古いバージョンで定義されていない新規スキーマバージョンの機能

新しいスキーマバージョンの機能を使用し、この機能をサポートしない古いバージョンに切り替える場合、以下が発生します:

- 機能を削除/変換、または、機能を保管するかを問うメッセージが表示されます。
- 保管する場合、コンポーネントまたは詳細に対応する新規機能がオレンジ色のテキスト色で表示されます。 例えば、値が `const` キーワードに設定されており（`draft-06` の新規機能）`draft-04` にスキーマバージョンを変更すると、`const` キーワードの値がオレンジ色で表示されます。

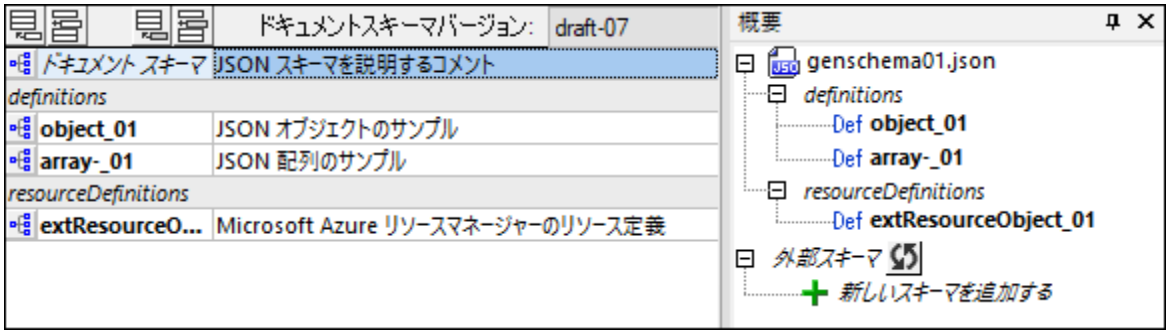
JSON スキーマバージョン

特に各バージョンに関する追加情報についてのJSON スキーマ仕様の詳細に関しては、下のリンクを参照してください:

JSON スキーマ仕様: http://json-schema.org/specification.html
Draft-06 リリースノート: http://json-schema.org/draft-06/json-schema-release-notes.html
Draft-07 リリースノート: http://json-schema.org/draft-07/json-schema-release-notes.html

13.5.12.2 グローバル定義の追加

メインウィンドウ内の**定義概要グリッド** 630（*下のスクリーンショット*）は、スキーマのグローバル定義を表示します。これらのグローバル定義は、以下のとおりです: (i) **メインドキュメントスキーマ** 定義、(ii) **オブジェクト**、配列、文字列などの、JSON スキーマ型である **グローバルな JSON 型**の定義、(iii) **外部、または、カスタムに定義された JSON 型** の定義。`resourceDefinitions` という名前のコンテナー内で発生する定義のみを使用することができます。これは、JSON 定義のためのMicrosoft Azure Resource Manager により使用されているコンテナーです。グリッドツールバーの **定義セクションの追加** または **定義セクションの挿入** アイコンを使用して、新しいリソース定義を追加します（*下のスクリーンショットを参照してください*）。

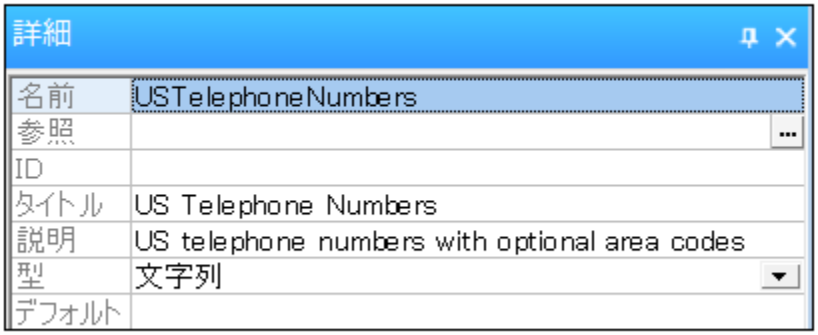


JSON 型[593] とグローバルに定義することは、同じスキーマおよび他のスキーマ内でその型を使用する際にとても便利です。例えば、1つの JSON スキーマ内で JSON 文字列の型を米国の電話番号で使用すると仮定します。 同じスキーマと他の JSON スキーマからもこの定義を参照することができます。

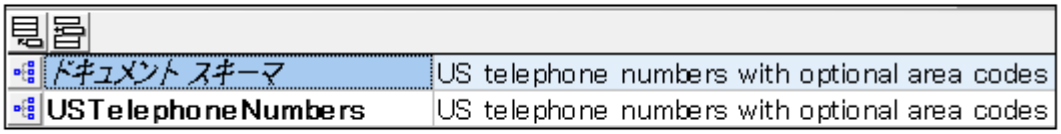
定義と関連したアクションの追加

定義概要グリッド内で定義を追加および編集するために次のアクションを使用することができます。

- *定義を追加する:* 定義概要グリッドの左上の「**名前を付けられたスキーマ定義を追加**」または「**名前を付けられたスキーマ定義を挿入**」をクリックします(*上のスクリーンショットを参照*)。新しい空の定義がグリッド内に作成されます。新規の定義は、概要入力ヘルパーにも *Def* としてリストされます(*上のスクリーンショットを参照*)。
- *定義の型の変更:* `Any` の型の新しい定義が作成されます。詳細入力ヘルパーで型を変更することができます(*下のスクリーンショットを参照。型が 'String' である箇所*) または、デザインビュー内の定義を編集。



- *定義の名前の変更:* 名前をダブルクリックして編集します。 または、詳細入力ヘルパー内の*名前* フィールドを編集します。
- *定義の説明の入力:* 詳細入力ヘルパー内の*詳細* フィールドを編集します。定義の名前の横の定義概要グリッド内に詳細が表示されます(*下のスクリーンショットを参照*)。詳細を編集するために定義概要グリッドの詳細フィールドをダブルクリックすることもできます。



- *定義の参照:* 詳細に関しては、概要入力ヘルパー[634] の説明とトピックグローバルとローカル定義[637]を参照してください。
- *定義の編集:* 定義概要グリッド内の定義アイコンをクリック、または概要入力ヘルパーの定義をダブルクリックします。これにより定義を編集することのできるデザインビュー内の定義が開かれます。

13.5.12.3 入力ヘルパー: 概要、詳細、制約

JSON スキーマビュー入力ヘルパーは、デフォルトで、アプリケーションウィンドウの右側に表示されます。メインウィンドウの両方のモード[630]で使用することができます: (i) 定義概要グリッド、(ii) デザインビュー。入力ヘルパーウィンドウのタイトルバーをドラッグしてスクリーンの他の場所に移動することができます。入力ヘルパーのタイトルバーを代わりにダブルクリックして、入力ヘルパーをドッキングまたはドッキング解除することができます。詳細に関しては、入力ヘルパー[113]のセクションを参照してください。

概要入力ヘルパー

概要入力ヘルパー(*下のスクリーンショット*)は、現在のスキーマの定義および現在のスキーマのすべてのグローバル定義をリストします。定義をダブルクリックすることにより、デザインビュー[638] 内でその定義を開き編集することができます。ことができます。外部スキーマからの定義を使用するには、最初に外部スキーマを追加して、定義を再利用します。

外部スキーマの追加

概要入力ヘルパー内の「**新しいスキーマの追加**」アイコンクリックして、追加するスキーマを参照し、外部スキーマを追加します。スキーマが追加されると、スキーマが追加されると、その定義は概要入力ヘルパーに表示されます。下のスクリーンショットは、例えば、スキーマ `TelNumbers.json` が追加されたこと、また、このスキーマには、`USTelephoneNumbers` という名前の1つの定義があること表示しています。外部スキーマは任意の数追加することができます。

外部定義の再利用

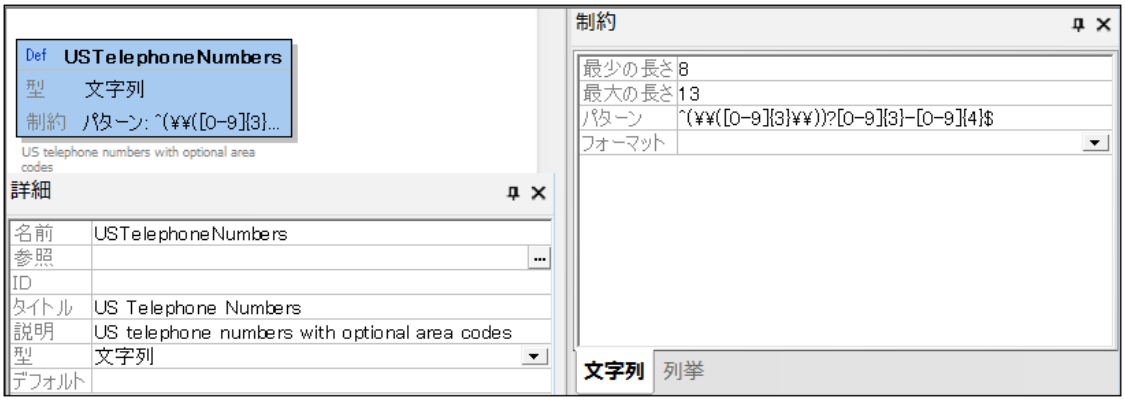
外部スキーマが追加された後、インポートされるスキーマの定義内で再利用することができます。定義の1つが他の定義を(参照して)使用する場合、その定義のプロパティを取ります。参照は以下の2つの方法で行うことができます:

- *デザインビュー内:* 概要入力ヘルパーから定義を定義が使用される箇所にドラッグします。
- *定義概要グリッド または デザインビュー内:* 他の定義を再利用する定義の詳細入力ヘルパーの参照フィールドを使用して。この点に関しては、詳細入力ヘルパー[635]で説明されています。

メモ T概要ウィンドウ内の外部スキーマの横の「**更新**」アイコンは、追加されたすべての外部スキーマを更新します。追加された外部スキーマから定義が再利用されない場合、そのスキーマはリストが更新されると削除されます。

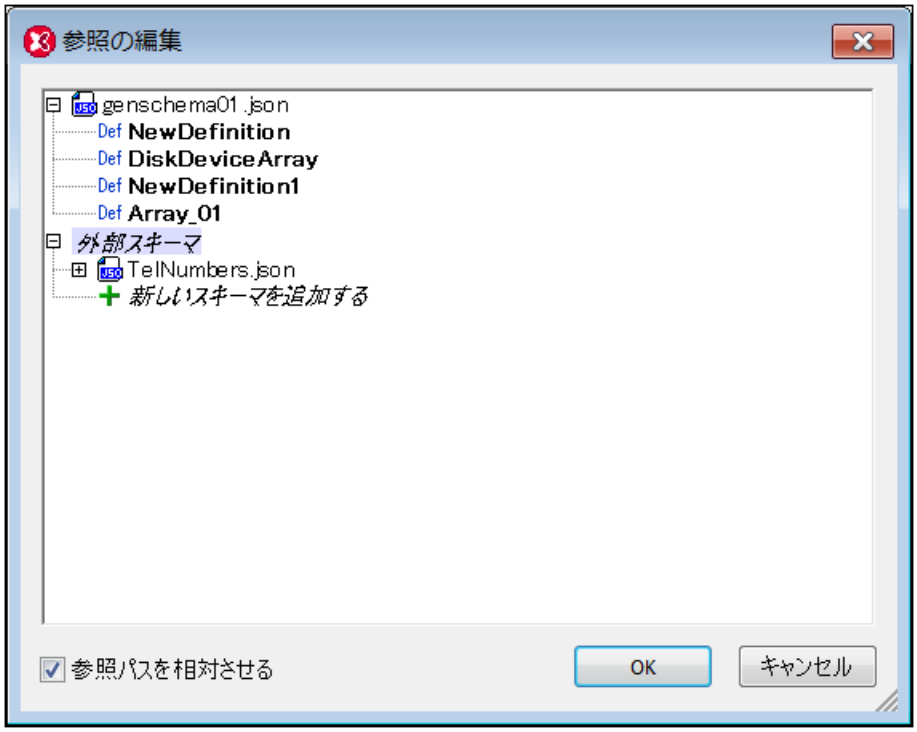
詳細入力ヘルパー

定義のプロパティは、メインウィンドウのいずれかのモード内で定義が選択されると、詳細と制約ウィンドウ内に入力することができます: 概要グリッド、または、デザインビューの定義[630]. 下のスクリーンショットは、詳細と制約入力ヘルパーと共にデザインビュー内の `USTelephoneNumbers` の定義を表示しています。2つの入力ヘルパー内の情報はデザインビュー[638]の定義(青)ボックスに表示されていることにご注意してください。 2つの入力ヘルパーに設定することのできるプロパティは下にリストされています。



詳細入力ヘルパーに次の詳細を入力することができます:

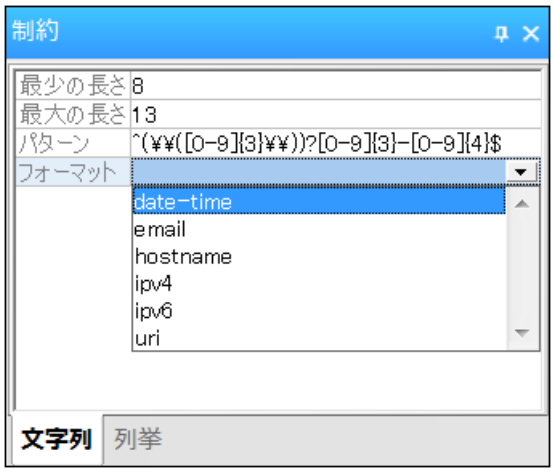
- *名前:* 定義の名前.
- *参照:* 定義で他の定義を再利用する場合、参照フィールドの「**追加ダイアログ**」ボタンをクリックします。これにより使用可能なすべての(現在のスキーマおよび外部スキーマからの)定義をリストする参照の編集ダイアログが表示されます(*下のスクリーンショット*)。再利用する定義を選択して、相対パスを使用する場合、相対パスのオプションを選択して、「OK」をクリックします。



- *型:* コンボボックスのドロップダウンリストから定義のデータ型を選択します。前の定義に戻るには、元に戻す (Ctrl-Z) を押します。これらの型は JSON 型[590] と型セレクタ (Any、Multiple、など)[652] で説明されています。
- *ID:* これはオプションのキーワードで、(親スキーマ内のサブスキーマとして考えられる) 現在の定義のソリューションスコープを変更するために使用されます。ID 値は、URI である文字列である必要があります。Altova JSON バリデーターは、正規の逆参照のみを使用することに注意してください。詳細に関しては、JSON 仕様を参照してください。
- *タイトル、説明:* これらの 2 つのキーワードはエンドユーザーにより読み取られることのできる詳細の目的のために使用されます。
- *コメント (draft-07 の新規機能):* エンドユーザーのための詳細に対して、スキーマの管理者のためのメモを想定しています。
- *制約 (draft-06 の新規機能):* 1 つの値の列挙のような定数です。
- *デフォルト:* 定義のデフォルトの値。
- *読み取り専用、書き込み専用 (draft-07 の新規機能):* これらは、それぞれ、読み取りと書き込み専用のフィールドを示しています。書き込み専用のフィールドはパスワードフィールドです。

制約入力ヘルパー

定義の制約は、型により異なります。各型の制約は、以下に説明されています (次も参照: アトミック型[651])。



型が下のリストに表示されない場合、制約を定義することはできません。しかしながら、すべての型に対して列挙を定義することはできます:

- *文字列:* 文字列の長さと文字列のパターン。パターンは正規表現を使用して指定することができます。フォーマットフィールドから仕様内で定義されているフォーマットの1つを選択することができます(*上のスクリーンショットを参照*)。*フォーマット* フィールド内で、, 仕様内で定義されている文字列のフォーマット の一つを選択することができます(*draft-04 内で使用することのできるフォーマットを表示する上のスクリーンショットを参照してください*)。追加フォーマットは以降のバージョンで定義されています。(draft-07 での新規機能である) *コンテンツメディア型とコンテンツエンコード* はJSON 文字列でエンコードされた非-JSON データのメディア型とエンコードを選択します。
- *数値:*許可されている数値の範囲
- *配列:*許可されている配列内のアイテムの数
- *オブジェクト:*許可されているプロパティの数

すべての型のための制約入力ヘルパーには列挙タブがあります。その定義の型で許可されているアイテムのリストを指定することができます。更に、*禁止された* 以外のすべての型のためにサンプルタブを使用することができます (draft-06 内では新規)。これは検証効果の無いサンプルの配列です。`default` の値は、このキーワードの下で繰り返しの無いサンプルとして使用することができます。

13.5.12.4 グローバルとローカル定義

JSON スキーマ定義 グローバルまたはローカルで設定することができます。

- **グローバル定義** は、定義を追加し、プロパティを指定することにより[632] メインウィンドウの 定義概要グリッド[630] 内で作成されます。グローバル定義は、同じスキーマまたは他のスキーマ内の定義により参照されることができます。これによりプロジェクト全体での定義の再利用が有効化されます。現在のスキーマのすべてのグローバル定義は、スキーマの 定義概要グリッド[630] に表示されています。他のスキーマからのグローバル定義は、(概要入力ヘルパー内の) 外部スキーマの追加[634] により再利用のために使用可能になります。
- **ローカル定義** は、グローバル定義に子孫 または 兄弟定義 (プロパティまたはアイテム) を追加することにより、グローバル定義内で作成されます。

グローバル定義の再利用

グローバル定義を再利用するには、以下の1つを行います:

- デザインビューで、グローバル定義を概要入力ヘルパーからドラッグして使用する定義の上にドロップします。

- デザインビュー内の定義概要グリッドから、再利用する定義を選択します。詳細入力ヘルパーの参照フィールド内で、再利用するグローバル定義を選択します。詳細に関しては、詳細入力ヘルパー[635] を参照してください。

メモ 同じスキーマ内の他の定義で参照された後に、グローバル定義の名前を変更する場合、参照内で名前も変更されます。他のスキーマからの参照は、しかしながら、名前変更を反映するために手動で編集される必要があります。

ローカル定義 を グローバル定義に変換する

ローカル定義を変換するには、デザインビュー内で定義を右クリックし、「グローバルにする」を選択します。グローバル定義が作成され、ローカル定義上に参照が作成されます。グローバル定義の名前は、自動的に生成されるため、変更するとその変更はローカル定義の参照にパスされます。、

グローバル定義の参照をローカル定義に変更する

グローバル定義への参照は、ローカルおよびグローバル定義上に存在することができます。参照を削除して、プロパティをローカルにするには、デザインビュー内のローカルまたはグローバル定義を右クリックして、「ローカルにする」を選択します。グローバル定義のプロパティは、定義上にローカルで作成されます。

13.5.12.5 デザインビュー

デザインビューでは、構造および許可されているグローバル定義の値を指定することができます。定義は次の GUI コンポーネント または メカニズムを使用して指定することができます:

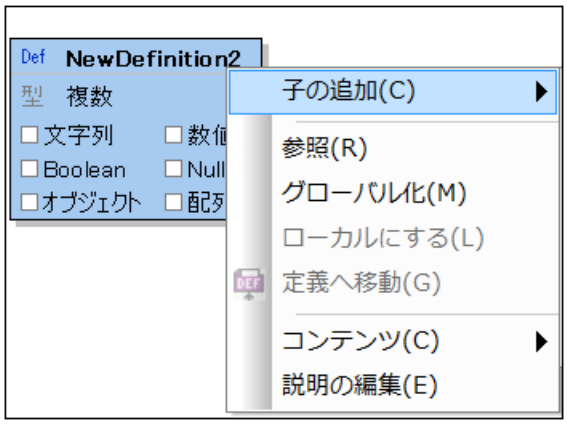
- 詳細入力ヘルパー[635] (定義概要グリッド内で使用することもできます)
- 入力ヘルパーの制約[636] (定義概要グリッド内で使用することもできます)
- 定義のコンテキストメニュー(メインウィンドウの定義ボックスを右クリックしてアクセスすることができます)

詳細と入力ヘルパーの制約により指定することのできる定義は、入力ヘルパー: 概要、詳細、制約[634] のセクションで説明されています。これらのプロパティの一部は、メインウィンドウの定義ボックスで指定することもできます。このセクションでは、次の3 つのセクションで、メインウィンドウで使用することのできるグラフィカルなメカニズムについて説明を行います。

メモ 誤った操作および希望しない変更を元に戻すには、「Ctrl+Z」を押してください。

コンテキストメニュー

(*下のスクリーンショットの青のボックス*) のコンテキストメニューにより、定義の構造のデザイン、およびプロパティの編集を有効化することができます。

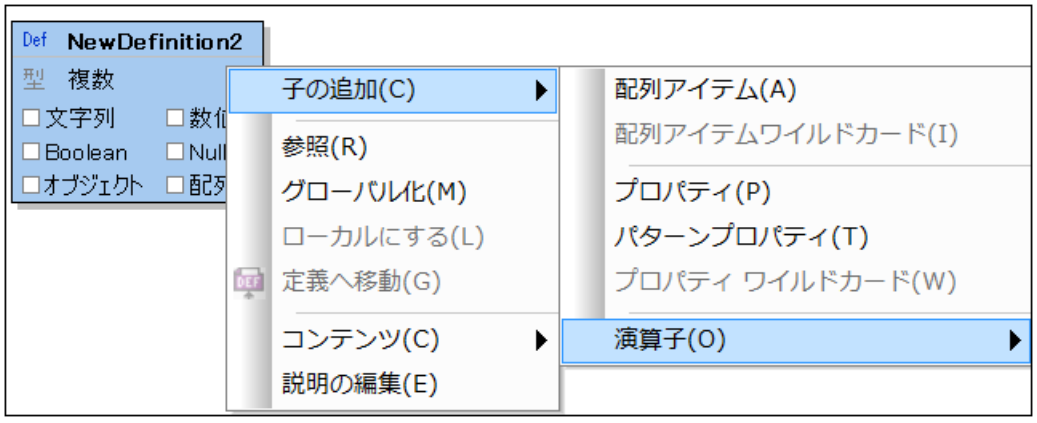


以下のコマンドを使用することができます:

- **子の追加**: 定義の型により追加することのできる子が決定されます(下の[子の追加: 構造の作成](639)を参照してください)。
- **参照**: 定義がグローバル定義を参照することを有効化し、そのグローバル定義のプロパティを取ります。コマンドが開く参照の編集ダイアログは、[詳細入力ヘルパー](635)を使用してアクセスすることのできるダイアログと同じです。詳細は[入力ヘルパー: 概要、詳細、制約](635)を参照してください。
- **グローバルに設定**: このコマンドは、[ローカル定義](637)が定義の場合に有効化されます。現在選択されている定義を[グローバル定義](637)とし、現在の選択にそのグローバル定義の参照を追加します。 I
- **ローカルに設定**: このコマンドは、[グローバル定義](637)が定義の場合に有効化されます。現在選択されている[ローカル定義](637)を元のグローバル定義への参照を作成して変換します。
- **定義に移動**: 選択された定義がグローバル定義を参照する定義に含まれる場合、このコマンドは有効化されます。クリックするとグローバル定義に移動します。
- **コンテンツ**: コンテンツ コマンドは、選択された定義のコンテンツを切り取り、コピーおよびリセットするを含むサブメニューを表示します。
- **詳細の編集**: 編集する定義の詳細フィールドを有効化します。

子の追加: 構造の作成

定義の構造は、複数のレベルの子孫を追加することにより作成されます。これらのレベルは、コンテキストメニューの**子の追加** コマンドにより作成されます。定義に追加する子はその型により異なります:



- *オブジェクト*[640]: プロパティと演算子を取ります

- *配列*[649]: 配列 アイテムと演算子を取ります
- *アトミック型 (文字列、数値、boolean、null)*[651]: 演算子を取ります
- *Any*[652]: プロパティ、配列 アイテム、と演算子を取ります
- *Multiple*[652]: 含まれる型によります。選択された方のための許可された子の和集合を取ります。
- *演算子*[655]: 論理演算子を構造を決定するために使用することを有効化します

各型のために作成される構造は、上のリストにリンクされているセクションで説明されています。

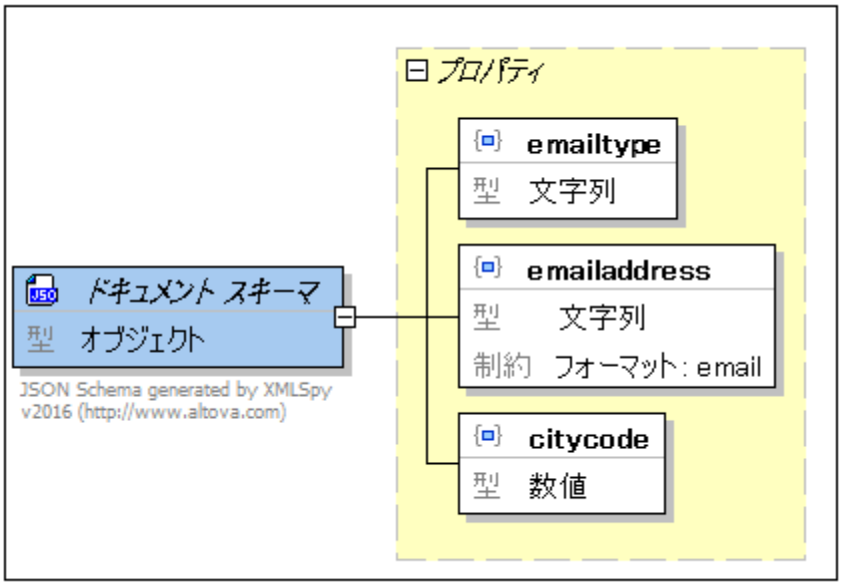
13.5.12.6 オブジェクトとプロパティ

オブジェクトは、中かっこで区切られており、値にキーを以下のようにマップします: `"MyKey": Value`。キーは常に文字列である必要があり、そのために引用符で囲まれている必要があります。値は、JSON データ型[590] 任意のであることができます。各 `key:value` ペアはオブジェクトのプロパティとして知られています (*下のスクリーンショット参照*)。

以下は、3 つのプロパティを持つインスタンス化されたオブジェクトの例です:

```
{
  "emailtype": "home",
  "emailaddress": "contact01.home@altova.com",
  "citycode": 22
}
```

オブジェクトのためのスキーマは、デザインビューで以下のように表示されます。



以下の点に注意してください:

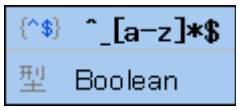
- それぞれのプロパティは、インスタンス内に存在する必要があります。これは、プロパティの境界により示されています。プロパティがオプションである場合、境界は点線で表示されています。プロパティが必要であるかまたはオプションであるかをプロパティのコンテキストメニューまたは詳細入力ヘルパーで設定することができます。
- プロパティがインスタンス内で発生する順序は、スキーマ内で定義することはできません。これは、スキーマ内で定義されるプロパティの順序は、無関係であるからです。

- かっこ内の青い角のシンボルは、(他のシンボルにより示されている)プロパティワイルドカード または プロパティワイルドカードに対して プロパティを示しています。下を参照してください。641.
- プロパティの型は、ダイアグラム内の型をダブルクリックして編集することができ、表示されるドロップダウンリストからオプションを選択することができます。または、型は詳細入力ヘルパー内で選択することができます。
- `emailaddress` プロパティの制約の値は、入力ヘルパーの制約で定義することができます。

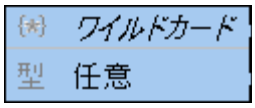
プロパティ、パターンプロパティ、プロパティワイルドカード、およびプロパティ名スキーマ

オブジェクトは、プロパティ、パターンプロパティ、プロパティワイルドカード、およびプロパティ名スキーマ(draft-07 の新規機能631)、を持つことができます。これらは、オブジェクト コンテキストメニューを使用して追加することができます: (i) オブジェクト、(ii) 黄色いプロパティボックス(ボックスのプロパティタイトルを右クリック)、(iii) それぞれのプロパティ。プロパティは上で説明されています。ここでは、パターンプロパティとプロパティワイルドカードについて説明をおこないます。

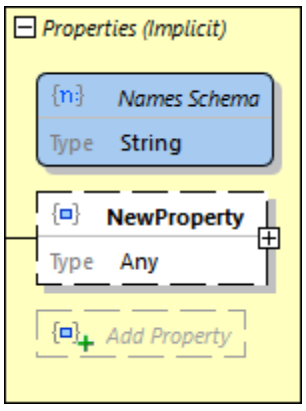
パターンプロパティ (*下のスクリーンショット*) は、プロパティの名前を正規表現として定義します。下の*スクリーンショット*では、例えば、正規表現は以下を指定します: (i) プロパティがアンダースコアで始まる名前を有する、そして、(ii) ブール値を値として有する。ワイルドカードのための制約条件はありません。パターンプロパティは任意の数追加することができます。パターンプロパティのためのアイコンに注意してください。



プロパティワイルドカード (*下のスクリーンショット*) は、オブジェクトのプロパティセットの他のプロパティに加え発生するプロパティを指定します。ワイルドカードはしかしながら、これらの発生のための型を定義します。下の*スクリーンショット*は、数値以外の値を持つ名前を持つプロパティを定義するプロパティワイルドカードを表示しています。各オブジェクトにプロパティワイルドカードは1つのみ存在することができます。ワイルドカードが*Any* 型、に設定されている場合は、入力ヘルパーの制約内のそれぞれの方の制約を設定することができます。プロパティワイルドカードのためのアイコンに注意してください。



プロパティ名スキーマ (*下のスクリーンショット*) はそのメンバーのプロパティの名前を制約します(これはdraft-07 の新規機能631です。)例えば、下の*スクリーンショット*では、プロパティの名前は文字列である必要があることがわかります。更に、制約入力ヘルパーを使用してプロパティ名のために更に制約を指定することができます: 例えば、プロパティの名前はプロパティ名は特定の文字の長さにおさまる、または、特定のパターンを持ちます。



メモ プロパティワイルドカード または プロパティワイルドカードの最小および最多の発生設定はありません。更に深い理解のためにプロパティの検証642を参照してください。

プロパティの検証方法

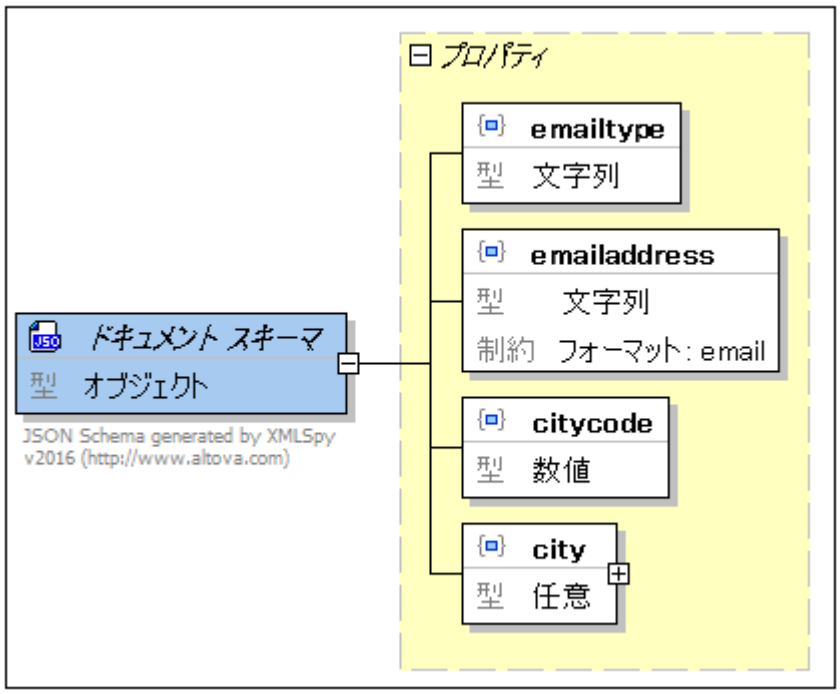
プロパティがインスタンスに遭遇した場合、以下の方法で検証することができます:

1. スキーマ内のプロパティの名前は、オブジェクトのすべての名前付けられたプロパティに対してチェックされます。
2. 一致が検出されない場合、名前はオブジェクトのプロパティセット内のすべてのパターンプロパティに対してチェックされます。
3. まだ一致が検出されない場合、存在する場合はワイルドカードが呼び出されます。
4. まだ名前との一致が検出されない場合、有効性エラーが報告されます。プロパティまたはパターンプロパティに一致する名前、またはワイルドカードが存在する場合、値がプロパティ定義に対応するに値に対してチェックされます。
5. インスタンス値が対応するプロパティ定義の型と制約に一致する場合、プロパティは有効です。それ以外の場合は無効です。

サンプル

下のスクリーンショットは、以下の条件を持つオブジェクトを定義しています:

- `emailtype`、`emailaddress`、および`citycode` という名前を持つ3 つのプロパティが必要です。
- アンダースコアで始まる名前を持ち、ブール値を持つ1 つまたは複数のプロパティを持つことができます(下のスクリーンショットのパターンプロパティを参照)。
- 任意の名前と値を持つ1 つまたは複数の追加プロパティを持つことができます。



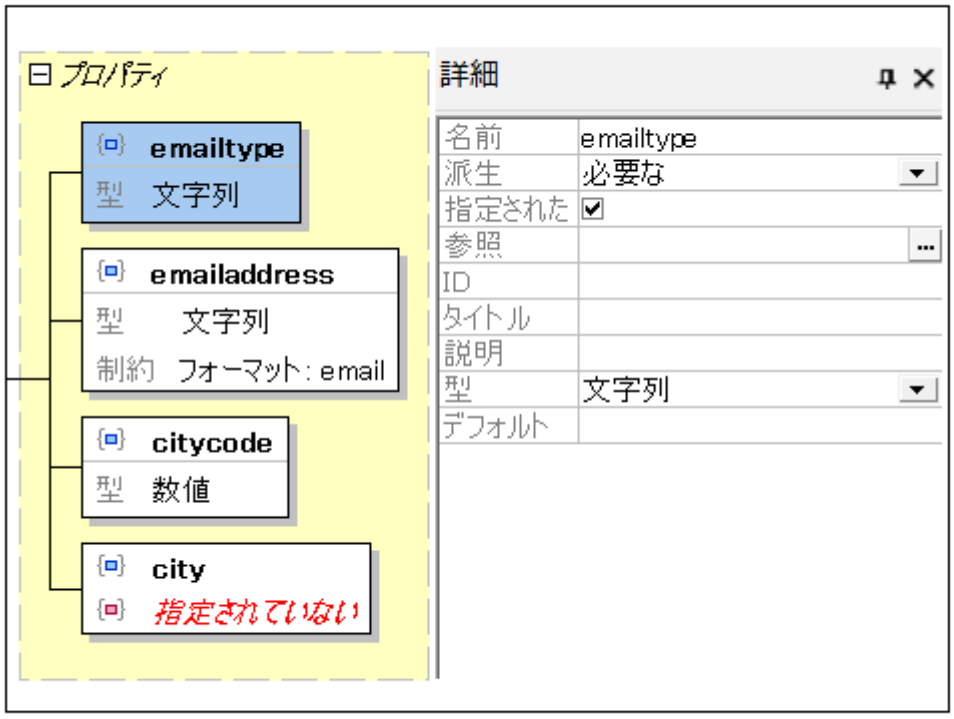
13.5.12.7 指定されていないプロパティ

下のコードのリストでは、`required` キーワードは、4 つのプロパティがこのオブジェクトのために必要なことを指定しています。しかしながら、4 つのプロパティの3 つは定義されています。4番目のプロパティ`city` は定義されていません。定義されているプロパティは指定されなければならず、定義されていないプロパティは、指定されていません。リストの下のスクリーンショットを参照してください。

(C) 2015-2021 Altova GmbH

コードのリスト: 指定されているおよび指定されていないプロパティ

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "description": "JSON Schema generated by XMLSpy v2016 (http://www.altova.com)",
    "type": "object",
    "properties": {
        "emailtype": {
            "type": "string"
        },
        "emailaddress": {
            "type": "string",
            "format": "email"
        },
        "citycode": {
            "type": "number"
        }
    },
    "required": [
        "emailtype",
        "emailaddress",
        "citycode",
        "city"
    ],
    "additionalproperties": false
}
```



(C) 2015-2021 Altova GmbH

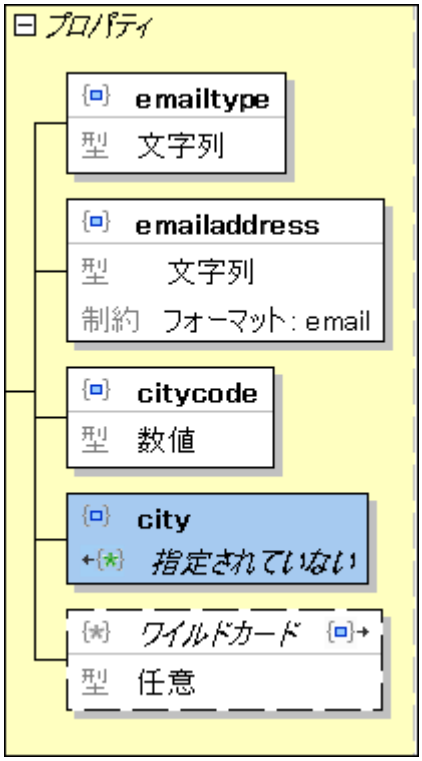
デザインビュー内では、スキーマで必要とされているにもかかわらず定義されていないため、指定されていないプロパティは赤でフラグ付けされています。JSON スキーマ自身は、有効ですが、検証に使用されるインスタンスドキュメントが有効ではありません。これには以下の理由があります: (i) `city` プロパティが存在しない場合、`city` プロパティが必要なため、ドキュメントは無効になります。`city` プロパティが存在する場合、`city` プロパティが定義されておらず、このプロパティのためのプロパティワイルドカードが許可されていないため、ドキュメントは無効になります (*下の**暗黙的にプロパティを指定する*642 *を参照*)。

指定されていないプロパティのための定義を作成するには、以下を行います:

1. デザインビュー内の指定されていないプロパティを選択します。
2. 詳細入力ヘルパー内の指定されたチェックボックスをチェックします (*上のスクリーンショット参照*)。または、指定されているフラグをコンテキストメニューを使用して削除します。
3. 必要に応じてプロパティの定義を変更します。

暗黙的にプロパティを指定する

プロパティは、適切なパターンプロパティまたはプロパティワイルドカードを追加することにより暗黙的に指定することができます。下のスクリーンショットは、追加されたプロパティワイルドカードを表示しています。`city` と名前の付けられたインスタンスプロパティは、このワイルドカードに一致します。スキーマ内では、ですから、`city` プロパティは、ワイルドカードにより暗黙的に指定されたということができます。`city` プロパティを含むインスタンスファイルはこのスキーマに対して有効になります。



暗黙的に指定されているプロパティ内とプロパティワイルドカード内のそれぞれのアイコンに注意してください。各アイコンは他のプロパティへのリンクです。アイコンをダブルクリックするとの他のプロパティを選択します。

13.5.12.8 オブジェクトと依存性

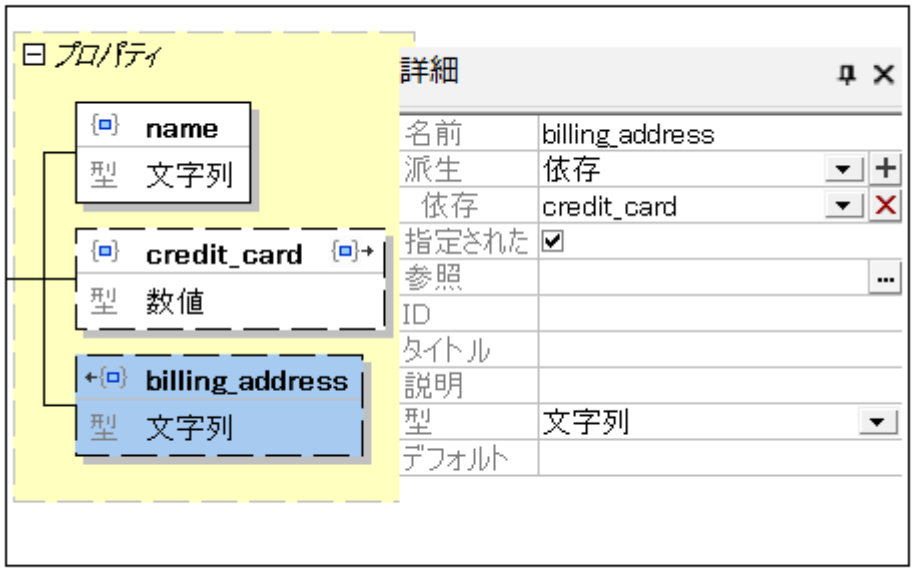
オブジェクトの定義では、特定のプロパティが他のプロパティが存在する場合のみ存在するように指定する必要がある場合があります。最初のプロパティは、2番目のプロパティの依存性となります。このセクションでは依存性を含むシナリオについて説明を行います。オブジェクト（例えば `member`）には `credit_card` という名前のプロパティがあるとします。オブジェクトの`billing_address` プロパティは、`credit_card` プロパティの依存性に設定することができます。これは、`credit_card` プロパティが存在する場合のみ、`billing_address` プロパティが存在することを意味します。

この種類の依存性は2通りの方法で指定することができます:

- プロパティ依存性として（依存性構造はプロパティです）
- スキーマ依存性として（依存性構造はスキーマです）

プロパティ依存性

下のスクリーンショットは、`name` プロパティ（必須）、`credit_card` プロパティ（任意）、および`billing_address` プロパティ（依存性）を持つオブジェクトを表示しています。`billing_address` プロパティは、依存性 `credit_card` プロパティにあります。JSON オブジェクト定義のコードは、下のスクリーンショットにリストされています。また、プロパティ依存性の作成方法は下に説明されています。



プロパティ依存性を持つ JSON オブジェクトのコードのリスト

```
{
   "type": "object",
   "properties": {
      "name": {
      "type": "string"
      },
      "credit_card": {
         "type": "number"
      },
      "billing_address": {
         "type": "string"
      }
   },
   "required": [ "name" ],
   "dependencies": {
      "credit_card": [ "billing_address" ]
   },
   "additionalProperties": false
}
```

プロパティ依存性を作成するには、以下を行います:

1. 依存性がベースにされるプロパティを右クリックします。(サンプルでは、`credit_card` プロパティです。)
2. 表示されるコンテキストメニューから [依存性の追加 | 依存性プロパティ] を選択します。新しいプロパティが依存性の値の発生に追加されます。
3. このプロパティの名前と値を定義し、追加の詳細または制約を追加します。

他のプロパティの依存性としてプロパティを指定するには、以下を行います:

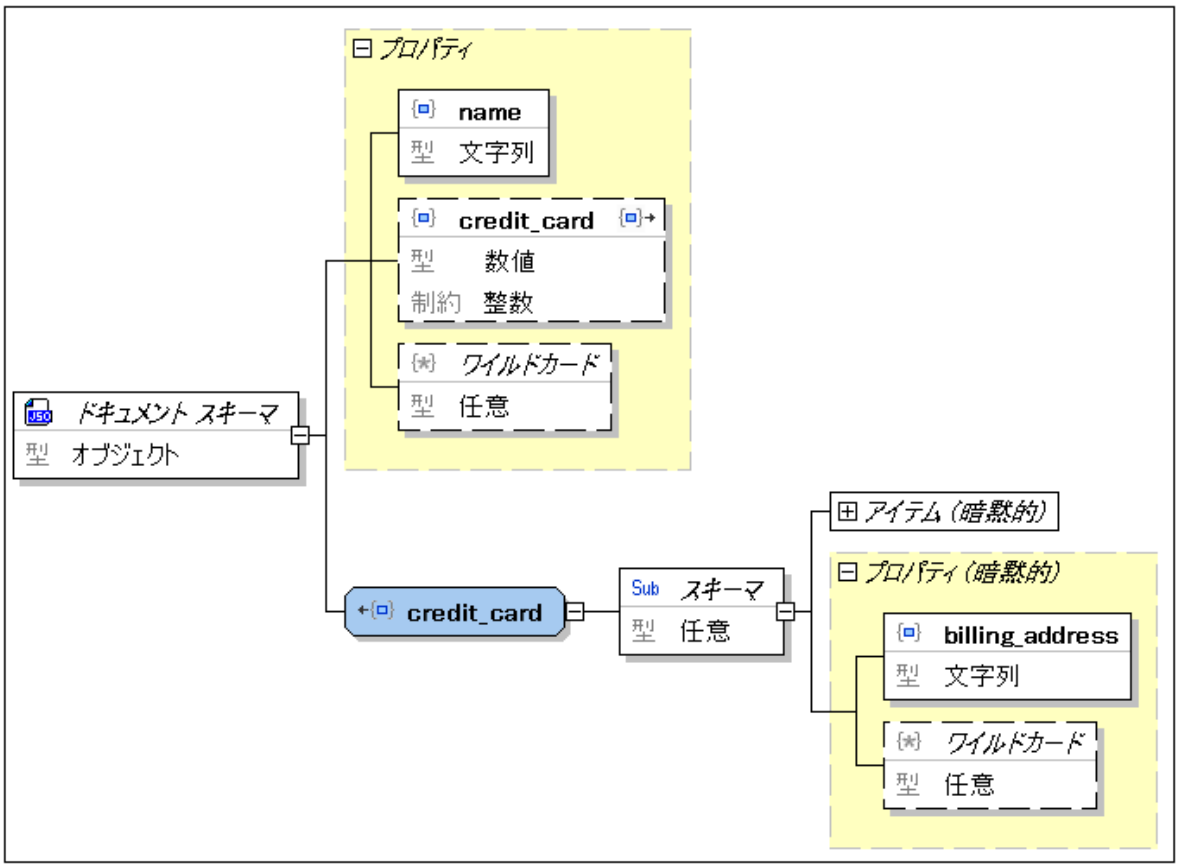
1. 他のプロパティの依存性とするプロパティを右クリックします。(サンプルでは、`billing_address` プロパティ.)
2. 表示されるコンテキストメニューから、[依存性] を選択します。または、詳細入力ヘルパーで、出現の入力に移動して、*依存性* を選択します(*上のスクリーンショット参照*)。
3. 詳細入力ヘルパーで、入力の依存性のドロップダウンリストのアイコンをクリックします。ドロップダウンリストはオブジェクトのすべての他のプロパティを表示します。現在のプロパティが依存するプロパティを選択します。

メモ 依存性に含まれる両方のプロパティのボックスが表示されますアイコン(*上のスクリーンショット参照*)。プロパティのアイコンをダブルクリックすると他のプロパティに移動します。

メモ プロパティは、複数の依存性 プロパティを持つことができます。

スキーマ依存性

下のスクリーンショットは、前のセクションで同じインスタンスデータ構造をオブジェクトとして説明するオブジェクトについて説明しています。を表示します。2つのオブジェクトの定義は、しかしながら、異なります。前の定義は`billing_address` プロパティを`credit_card` プロパティ上の依存性と定義するためにプロパティ依存性を使用していますが、現在の定義は、スキーマ依存性をこの依存性を定義するために使用しています。後者のJSON オブジェクト定義のコードは下のスクリーンショットにリストされています。どのようにスキーマ依存性を作成するかは下に説明されています。



- *スキーマ依存性を持つJSON オブジェクトのコードリスト*

スキーマ依存性を作成するには、以下を行います:

1. 依存性がベースにするプロパティを右クリックします（サンプル内では `credit_card` プロパティです）。
2. 表示されるコンテキストメニューから「**依存性の追加 | スキーマ依存性**」を選択します。新しいオブジェクト定義が作成されます。オブジェクトは、依存性のプロパティと同じ名前（サンプル内では `credit_card`）を持ち、子サブスキーマを持ちます。
3. サブスキーマを定義して、追加詳細または制約を追加します。

メモ 依存性に関連するプロパティおよびオブジェクトのボックス内にアイコンが表示されます（*上のスクリーンショット参照*）。ボックス内のアイコンをダブルクリックすると他のボックスに移動することができます。

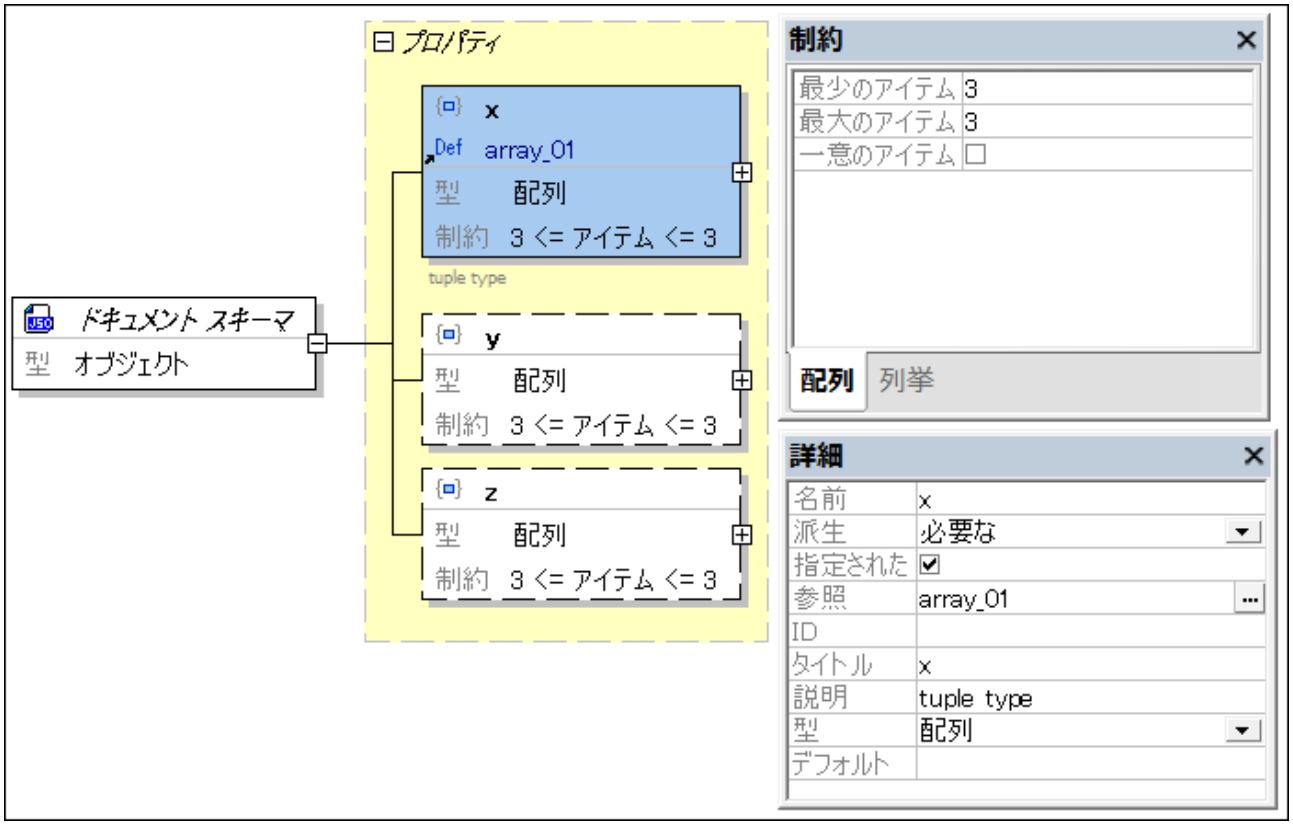
メモ 複数の依存性を設定するには、依存性 サブスキーマ 内で行います。（*上のスクリーンショット参照*）。

13.5.12.9 配列

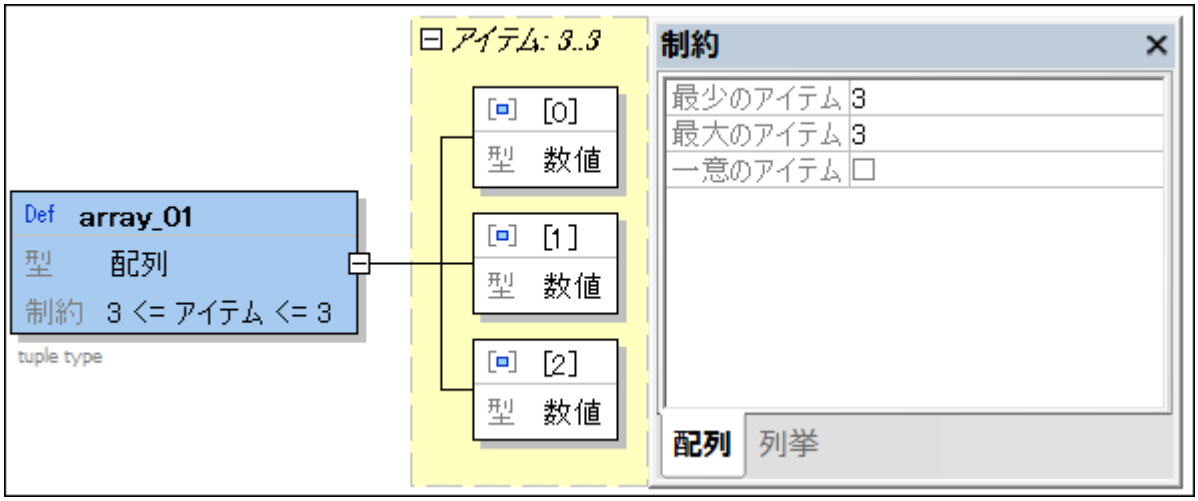
配列は、順序付けされたゼロまたは複数のアイテムで、角かっこにより区切られています。リスト内の各アイテムには、型が割り当てられています。下のインスタンスのリストは、3 つのプロパティを持つオブジェクトです。各プロパティの値は（黄色のハイライトで区切られている）配列です。

```
{
      "x": [ 1, 2, "abc" ],
      "y": [ 3, 4, "def" ],
      "z": [ 5, 6, "ghi" ]
}
```

上のリスト内の 3 つすべての配列は、同じ定義を持ちます。それぞれ、3 つの順序付けされたアイテムを次の順番で含んでいます: (i) 数値アイテム、(ii) 数値アイテム、(iii) 文字列アイテム。このオブジェクトのスキーマの詳細は、下のスクリーンショットに表示されています。3 つの配列すべての定義は同じなため、定義は、`array_01` という名前のグローバル配列内で作成されています。3 つの配列（`x`、`y`、および `z`）のそれぞれは、以下のグローバル配列を参照しています: 637 `array_01`。



上のスクリーンショットでは、(青いハイライトで表示されている) 配列 x が選択されています。詳細および制約は、それぞれ入力ヘルパーに表示されています (*上のスクリーンショット参照*)。許可されているアイテムの制約の数に注意してください。数は、制約入力ヘルパーで編集することができ、ダイアグラム内で表示されています。配列 アイテムは、配列自身の定義内で定義することができます。この場合は、グローバル定義 `array_01` (*下のスクリーンショット*) です。



以下の点に注意してください:

- `unique` 制約は、配列内のすべてのアイテムが一意であることを指定します。

- アイテムの番号付けは 0 から開始されます。
- ダイアグラム内のフレーズ、*3 <= アイテム <= 3* および *アイテム: 3..3* (*上のスクリーンショット参照*)、は、許可される最小および最大のアイテムを表しています。この場合、3つのアイテムが存在する必要があります。

配列アイテム、配列アイテムワイルドカード、contains キーワードの追加方法

配列アイテム、配列アイテムワイルドカード、および配列の `contains` キーワードは定義、または、配列アイテムのコンテキストメニューを使用して追加することができます。配列ワイルドカードにより、配列内に含まれるオブジェクトの範囲を広くすることができます。`contains` キーワードは一方、`contains` キーワードの値が有効な JSON スキーマであること、および、配列の要素の少なくとも1つが与えられたスキーマに対して有効であることを指定します。

13.5.12.10 アトミック型

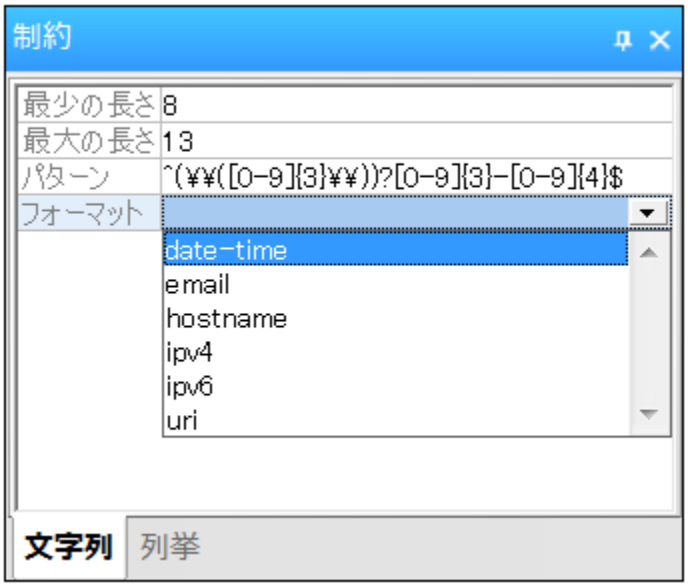
5つの JSON アトミック(単純型 または プリミティブとして知られる)型があります: (i) `string`、(ii) `number`、(iii) `integer`、(iv) `boolean`、および(v) `null`。これらのアトミック型のうちの1つの定義を指定するには、、以下のうち1つを行います:

- 定義のボックス内の*型* 値フィールドをダブルクリックして、型を選択します。
- 詳細入力ヘルパー内で、*型* フィールド内のドロップダウンリストから型を選択します。

各アトミック型の制約は、下に説明されています。

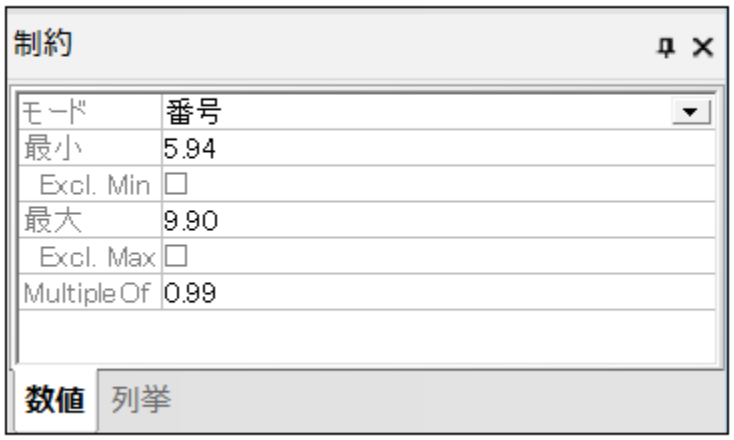
文字列

文字列 型に関しては、以下の制約を指定することができます: (i)文字列の長さ、(ii) 文字列のパターンを説明する正規表現、(iii) 仕様からの定義済みのフォーマット。



数値

数値 型は、2つの型（数値 および 整数。 *下のスクリーンショット参照*）のための総合的な名前です。実際の型は、モードフィールド（ デフォルトは数値）で設定されます。2つの型の間の違いは、数値型は小数点を許可しますが、整数型は許可しない点です。*MultipleOf* フィールド内に値が存在すると、*インスタンス*値は、*MultipleOf* 値の整数である必要があります。



上の*スクリーンショット*で定義されている 数値 型の有効な値は、以下のとおりです: `5.94`、`6.93`、`7.92`、および`8.91`.

Boolean とNull

`boolean` 型は、値として`true` または`false` を取ります。 `null` 型 は、値として`null` を取ります。Neither 型 は任意の制約を取ります。

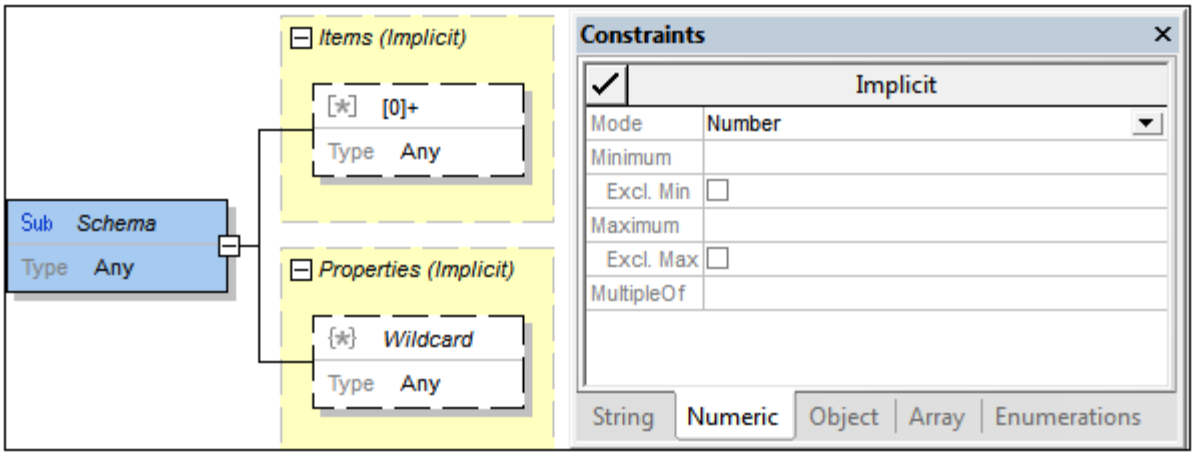
13.5.12.11 型セレクター（Any、Multiple）

JSON *スキーマビュー*の*型コンボボックス*の*ドロップダウン*には、JSON 型: ではない、以下の2つの"型"があります: **`any`** および **`multiple`**。をリストします。これらは、*型セレクター*です。

- **`any`** 型*セレクター*は、任意のJSON 型を選択します。これは、*インスタンス*内で任意のJSON 型 がその特定の定義に対して有効であることを意味します。
- **`multiple`** 型*セレクター*は、1つまたは複数のJSON 型を選択します。これは、*インスタンス*が、*スキーマ*内で選択されたJSON 型の1つである場合、*インスタンス*はその特定の定義に対して有効であることを意味します。
- (draft-06 内で新規の[631]) **`unconstrained`** 型*セレクタ* はJSON 型に制約を設定しません。これは、*インスタンス*内で、全てのJSON 型はその名前を持つ定義に対して有効です。
- (draft-06 内で新規の[631]) **`forbidden`** 型*セレクタ* はその名前を持つ定義の存在を効果的に許可せずJSON 型を禁止します。

any 型セレクター

any 型セレクターは、型が選択できるすべての箇所で選択されることができます。定義がスキーマに追加されると、 any がデフォルトの多々の選択になります。また、JSON 型の1つが有効であることを指定します。これは、インスタンス型はオブジェクト、配列またはいずれのアトミック型（string、number、integer、boolean および null）である可能性があります。



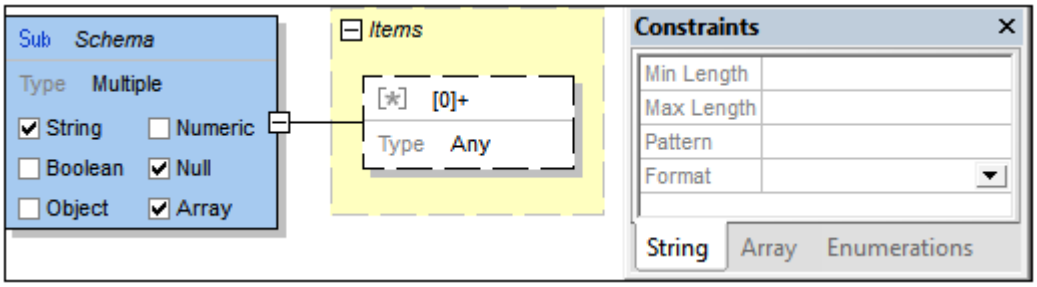
上のスクリーンショットでは、サブスキーマは Any の型を有します。ですから、すべてのJSON 型 は、この定義に対して有効です。以下はユーザーインターフェイスに従い暗黙的に実装されています:

- オブジェクトが許可されているため、プロパティボックスは自動的に作成されます（*上のスクリーンショット参照*）。プロパティボックスは、デフォルトで、（Any の型のプロパティワイルドカードを使用して）任意の型の任意の数のプロパティを許可するように定義されています。プロパティの定義を自由に変更することができます。
- 配列が許可されているため、アイテムボックスは自動的に作成されます（*上のスクリーンショット参照*）。配列ボックスは、デフォルトで、（Any の型のアイテムワイルドカードを使用して）任意の型の任意の数のプロパティを許可するように定義されています。アイテムの定義を自由に変更することができます。
- 文字列と数値（数字と整数）は、許可されているため、これらのアトミック型のための制約は、制約入力ヘルパー内で定義することができます。

これらの型すべては、ですから、暗黙的に、 Any 型セレクターにより定義されています。特定の型に型を変更する場合はその型を選択してください。オブジェクトと配列を指定するには別の方法があります: オブジェクトまたは配列を右クリックして、**明示的に設定**を選択します。この方法は、選択された型を型とし、他の型を削除、またはオブジェクト/配列型を非アクティブにします。

multiple 型セレクター

multiple 型セレクターは、型が選択される箇所すべてで選択されることができます。許可する型をチェックすることにより、1つまたは複数のJSON 型の選択を許可します（*下のスクリーンショット参照*）。選択された制約入力ヘルパー内の型のための制約を指定することができます。



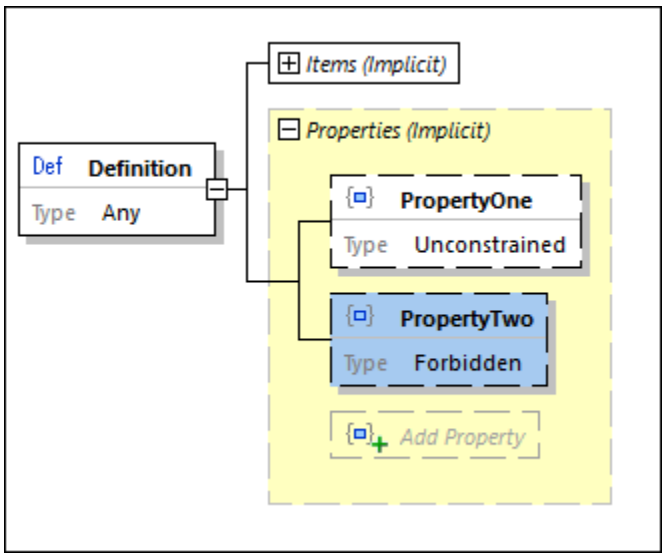
上のスクリーンショットでは、サブスキーマは、`string`、`null`、および`array` の型を許可します（*スクリーンショット参照*）。

- 文字列の制約は入力ヘルパーの制約で定義されています。
- null 型は、furr 制約をとりません。
- 配列アイテムボックスが自動的に作成されます。許可された配列アイテムの数値と型を定義することができます。

インスタンスドキュメントでは、選択された型はサブスキーマに対応する箇所に許可されています。

制約がなく禁止されている型セレクタ

`unconstrained` と`forbidden` 型セレクタは型を選択できる箇所で選択することができます。これにより、全ての型のオブジェクトが許可されている、または、その名前のオブジェクトが許可されていないかをそれぞれ指定することができます。

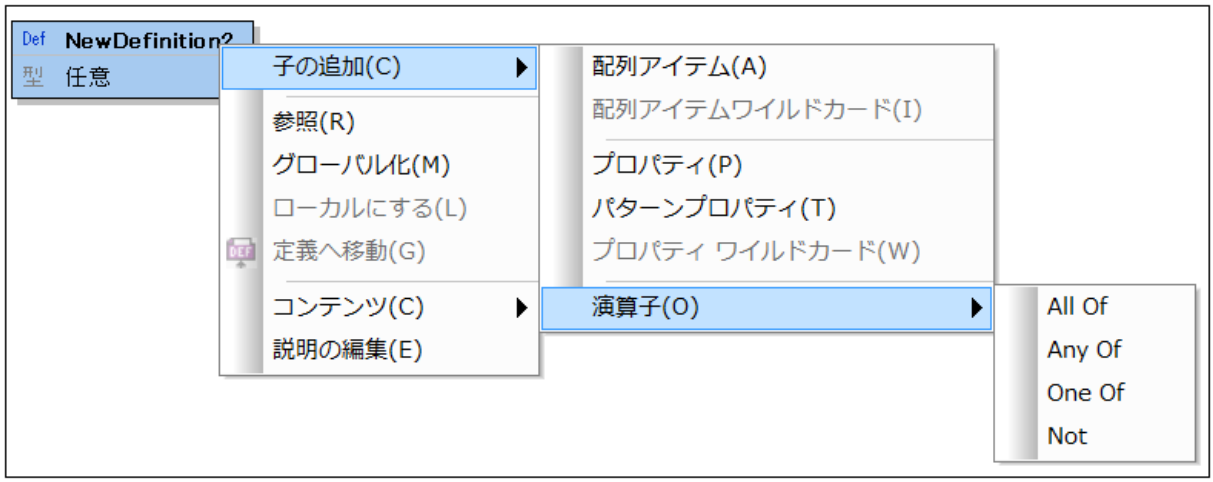


上のスクリーンショットでは、定義には2つのプロパティが存在します。**PropertyOne** はすべての型を持つことができますが、**PropertyTwo**という名前のプロパティは許可されていません（*スクリーンショットを参照してください*）。テキストの書式では、これは以下のようなコードリスティングを作成します。

```
"Definition": {
	"properties": {
		"PropertyOne": true,
		"PropertyTwo": false
	}
}
```

13.5.12.12 演算子

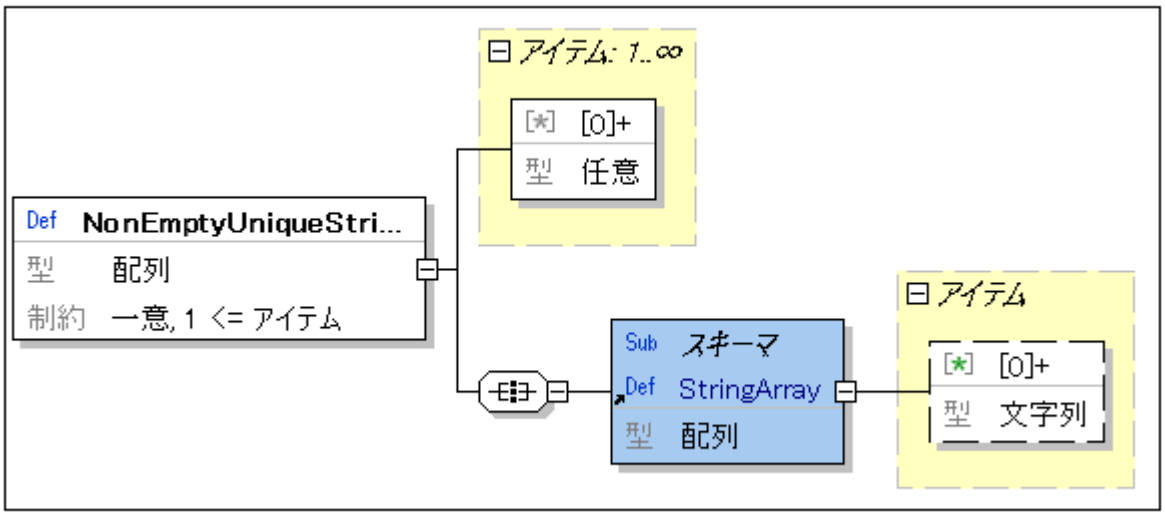
以下の4つの演算子が存在します: (i) `allOf`、(ii) `anyOf`、(iii) `oneOf`、および(iv) `not`。演算子は、下に説明されるように有効性の条件を指定します。任意の定義に演算子を追加することができます。演算子 サブメニューにアクセスするには、演算子を追加する定義を右クリックして、**「子の追加 | 演算子」**を選択します(*下のスクリーンショット参照*)。



これらの演算子は、検証を成功するための条件を以下のように定義します:

演算子	アイコン	詳細
	All Of	値は、演算子の子として追加された1つまたは複数のサブスキーマ(定義)の配列です。*インスタンス*は、配列内のすべての定義に対して有効な場合有効です。
	Any Of	値は、演算子の子として追加された1つまたは複数のサブスキーマ(定義)の配列です。*インスタンス*は、配列内の1つまたは複数の定義に対して有効な場合有効です。
	One Of	値は、演算子の子として追加された1つまたは複数のサブスキーマ(定義)の配列です。*インスタンス*配列内の1つの定義に対して有効な場合有効です。
	Not Of	値は、演算子の子として追加されサブスキーマ(定義)たです。*インスタンス*は、与えられた定義に対して無効な場合有効です。

下の*スクリーンショット*は、3つのサブスキーマ(定義)の配列を子として持つ*All Of* 演算子を表示しています。*インスタンス*を有効にするには、(ドキュメント構造のこの箇所で) 3つのサブスキーマ定義の1つに一致する1つのJSON データ構造を持つ必要があります。

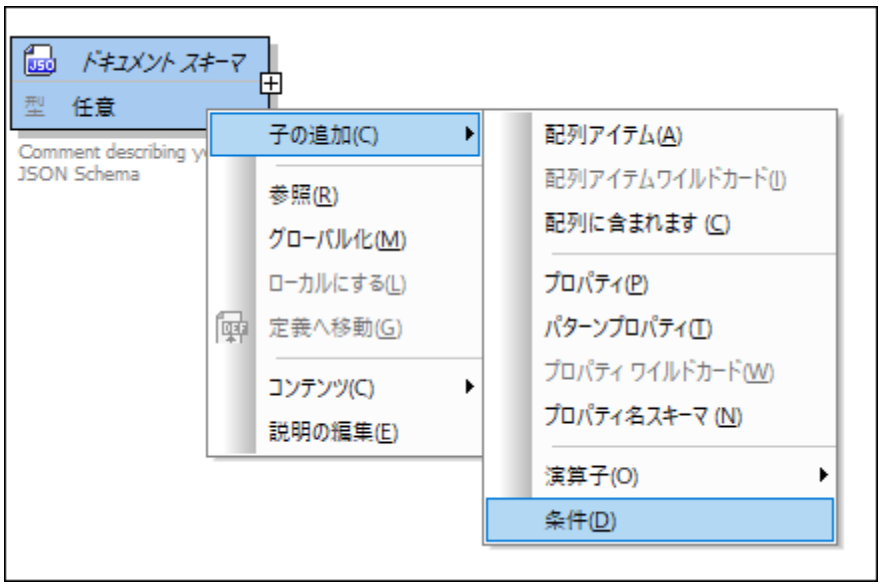


13.5.12.13 条件

条件は[draft-07](631) の新規機能です。これにより、型および/または追加の型固有の制約などオブジェクトの特定のアスペクトにより制約を異なるものに指定できるようになります。

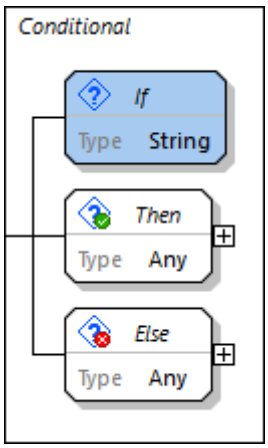
条件の追加

定義のコンテキストメニューを介して条件を定義に追加することができます (*下のスクリーンショットを参照してください*)。条件のサブメニューにアクセスするには、条件に追加する定義を定義を右クリックし、「**子の追加 | 条件**」を選択します。



条件の検証のセットアップ

条件は3つの要素を持つボックスとして追加されます: `If-Then-Else` (*下のスクリーンショットを参照してください*).

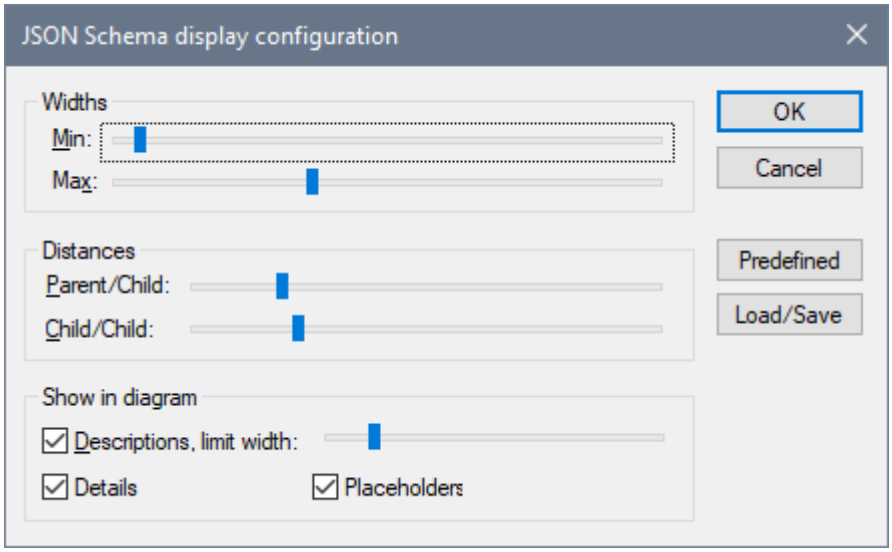


条件の検証をセットアップするには、以下を行います:

1. *If* ボックス内の条件を詳細入力ヘルパー内で最初に型を選択し、条件入力ヘルパー内の型ベースの条件を選択してセットアップします。
2. *Then* ボックス内で、(*If* ボックス内で指定されている) 条件が満たされていないイベント内の検証の必要条件をセットアップします。
3. *Else* ボックス内で、(*If* ボックス内で指定されている) 条件が満たされていないイベント内の検証の必要条件をセットアップします。

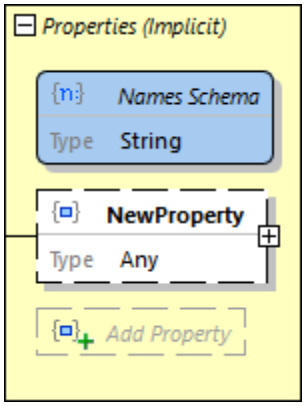
13.5.12.14 デザインビューの構成

メインウィンドウがデザインビューモードに設定されている場合、構成の表示ダイアログ (*下のスクリーンショット*) をメニューコマンド「**スキーマデザイン | ビューの構成**」を使用してアクセスすることができます。ここでデザインビューの外観を構成することができます。



デザインビューの次のアスペクトを構成することができます:

- *幅:* 2つのスライダーはそれぞれ、デザインビュー内のボックスの最小および最大の幅を決定します。また、ボックスの許可される幅の範囲を決定します。
- *親/子 の距離:* は、階層構造の各レベル間の左右の間隔を設定します。
- *子/子 の距離:* は、各ボックス間の上下の間隔を設定します。
- *詳細の幅:* は、詳細ラインの幅を設定します。テキストが幅をはみ出した場合、次のラインに折り返します。
- *詳細の表示:* 詳細ボックスのオプションをチェックまたはチェックを解除することにより、定義の詳細を表示または非表示に切り替えることができます。対応するツールバーアイコンが存在します。
- *プレースホルダの表示:* プレースホルダは定義されていないアイテムです。これらは可能性のあるアイテムを示しています。このオプションは、プレースホルダがオンに切り替えられているかを設定します。対応するツールバーアイコンが存在します。例えば、*プロパティの追加* アイテムは下のスクリーンショットではプレースホルダです。



メモ 「構成ビュー」メニューコマンドはオブジェクトの詳細な定義を表示するデザインビューモード[630]内でのみ有効化されています。定義概要グリッド[630]では使用することができません。

13.5.12.15 JSON スキーマドキュメンテーションの生成

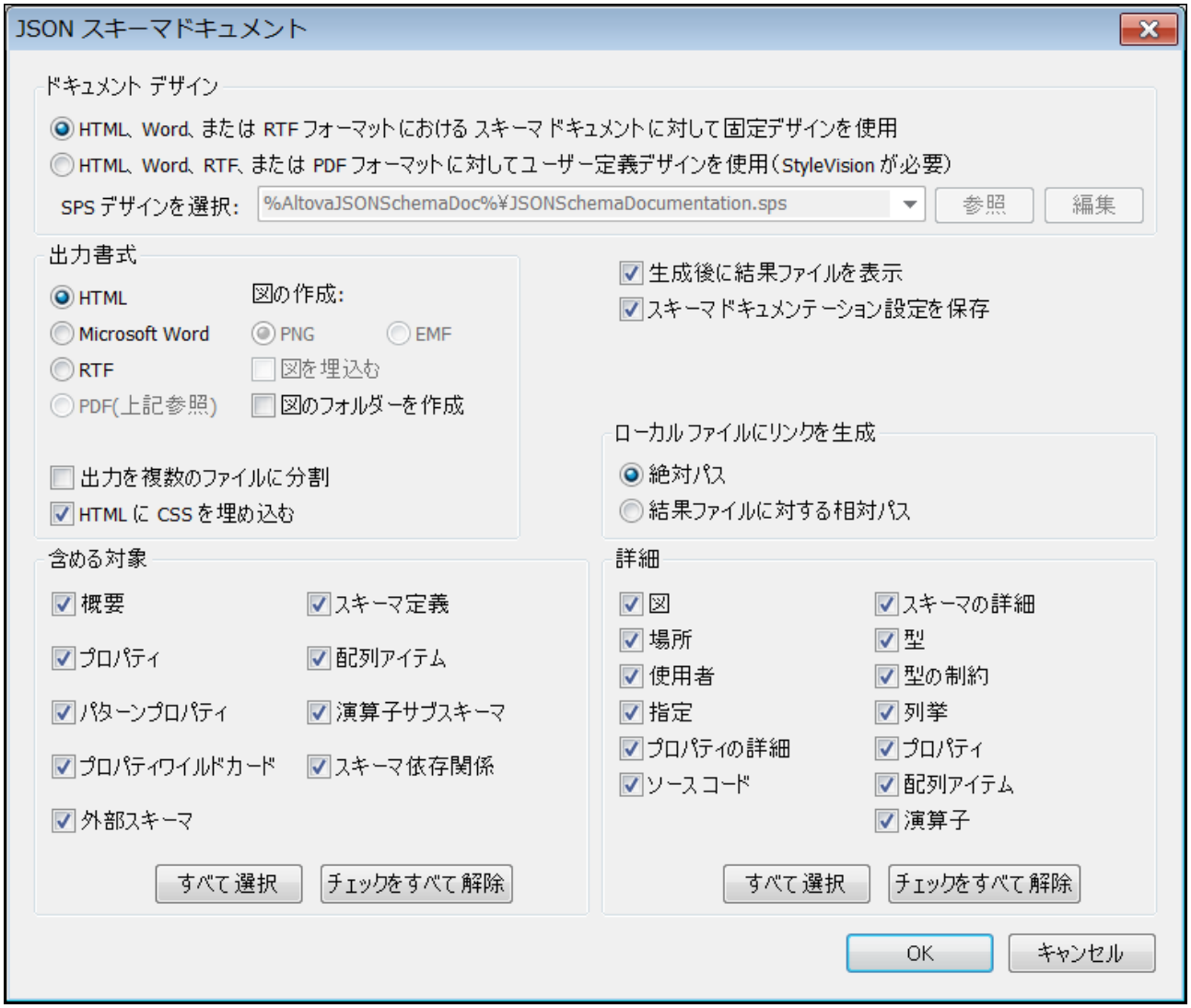
IJSON スキーマアクティブなドキュメントの場合、「**スキーマデザイン | ドキュメントの生成**」コマンドをクリックすることでドキュメントを生成することができます。ドキュメントの出力は、HTML、MS W ord、またはRTF ファイルで生成することができ、含まれるコンポーネントを指定することができます。関連するJSON コンポーネントは、生成されたドキュメント内でハイパーリンクが付けられ、簡単に移動することができます。

メモ MS W ord フォーマットでドキュメントを生成するためには、MS W ord (バージョン2000 またはそれ以降) がインストールされている必要があります。

JSON スキーマドキュメンテーション生成のステップ

JSON スキーマファイルのためにドキュメントを生成するためには、以下を行います:

1. JSON スキーマがアクティブなドキュメントであることを確認します。
2. スキーマビューに切り替えます。
3. メニューコマンド「**スキーマデザイン | ドキュメントの生成**」を選択します。これにより、JSON スキーマドキュメンテーション ダイアログボックスが開かれます (*下のスクリーンショット*)。
4. 生成する出力の型を次から選択します HTML、MS W ord、またはRTF。
5. ドキュメントに含まれる特別なコンポーネントと詳細を選択し、他のオプションを設定します (*下の JSON スキーマドキュメンテーションオプションを参照*)。



6. 「OK」をクリックして、JSON スキーマドキュメンテーションファイルの名前を、表示される名前をつけて保存ダイアログボックス内に入力します。

JSON スキーマドキュメンテーションオプション

次のドキュメントオプションから選択することができます:

- デザインテンプレート XMLSpy デザイン内でビルドイン(固定された)と指定することができ、またSPS ファイル内に保存されるユーザー定義デザインであることもできます。ユーザー定義 デザインの使用方法に関しては、ユーザー定義デザイン[1263] のセクションを参照してください。
- 出力フォーマットペイン内で必要なフォーマットを指定することができます: HTML、Microsoft W ord、またはRTF。ドキュメントは単一のファイルまたは複数のファイルに生成されることができます。複数のファイルが生成される場合、それぞれのファイルは、コンポーネントに対応します。どのコンポーネントが出力に含まれるかは、含むペインのチェックボックスを使用して指定します。
- ダイアグラムの埋め込みオプションは、オMS W ord とRTF 出力 プションのために有効化することができます。このオプションがチェックされていると、ダイアグラムはPNG またはEMF フォーマットとして結果ファイル内に埋め込まれます。それ以外の場合は、ダイアグラムは、オブジェクトリンクを介した結果ファイル内にPNG またはEMF ファイルとして作成されます。出力がHTML の場合、全てのダイアグラムは、ドキュメント外部 PNG ファイルとして作成されます。

- 含むペイン内で、ドキュメントに含むアイテムを選択することができます。概要オプションは、ファイルの上にコンポーネント型別に整理された全てのコンポーネントをリストします。スキーマ定義が選択されていない場合、(外部スキーマ以外のすべての) 子コンポーネントは無効化されます
- 詳細ペインは、各コンポーネントに含まれる可能性のある詳細をリストします。スキーマ定義が選択されていない場合、全ての詳細は無効化されます。ドキュメントに含む詳細を選択します。
- 結果ファイルの表示オプションは全ての出力オプションのために有効化されています。このオプションがチェックされていると、結果はブラウザービュー(HTML 出力)、MS W ord (MS W ord 出力)、.rtf ファイルのデフォルトのアプリケーション(RTF 出力)に表示されます。

13.6 JSON ドキュメントの検証

XMLSpy は、以下を行うことのできる JSON 検証エンジンを搭載しています:

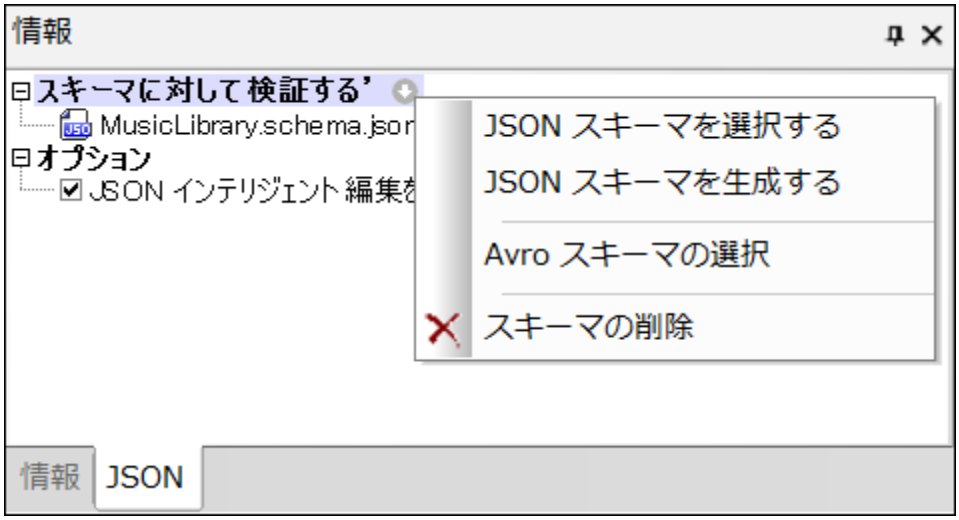
- *JSON スキーマがアクティブなドキュメントの場合:* JSON スキーマを適切な JSON スキーマ仕様に対して検証します(このため、追加のスキーマの割り当ては必要ありません)。スキーマバージョンは `$schema` [592] キーワードにより示されています。検証は、以下の3種類のビューで行うことができます(テキスト [595]、グリッド [600]、および JSON スキーマ [630])。
- *JSON インスタンスがアクティブなドキュメントの場合:* JSON インスタンスを JSON スキーマに対して検証します。スキーマは、JSON インスタンスに下で説明されるように割り当てられます。JSON インスタンス検証は、テキストビュー [595] とグリッドビュー [600] で実行されます。
- *JSON5 インスタンスがアクティブなドキュメントの場合:* JSON インスタンスを JSON スキーマに対して検証します。JSON5 インスタンスにスキーマが以下に説明されるとおり割り当てられます。JSON5 インスタンス検証はテキストビュー [595] とグリッドビュー [600] で実行することができます。

Avro 検証 (Enterprise Edition のみ)
Avro データと Avro スキーマドキュメントを JSON ドキュメントとして行われる検証は、テキストビュー [595] とグリッドビュー [600] で実行されます:

- *JSON フォーマットの Avro データインスタンスがアクティブなドキュメントの場合:* Avro インスタンスを Avro スキーマに対して検証します。スキーマは下で説明されるようにインスタンスに割り当てられます。
- *Avro スキーマがアクティブなドキュメントの場合:* Avro スキーマを Avro スキーマ仕様 に対して検証します(スキーマの割り当ては必要ありません)。検証は、テキストビュー [595] またはグリッドビュー [600] で実行されます。

JSON インスタンスに JSON または Avro スキーマを割り当てる

JSON インスタンスを JSON スキーマ、または Avro スキーマ スキーマに対して検証するには、アクティブなインスタンスドキュメントにスキーマを割り当てる必要があります。アクティブな JSON インスタンスドキュメントの情報ウィンドウ(下のスクリーンショット。)。または プロジェクトプロパティ [1194] ダイアログを使用して、割り当てを入力します(オプションつきの検証)。JSON5 インスタンスドキュメントは、JSON スキーマに対してのみ検証されることに注意してください。



情報ウィンドウの JSON タブで、*スキーマに対する検証*の横のアイコンをクリックします、そして、表示されるメニューから、「JSON スキーマ、または、Avro スキーマの選択」 をクリックします(下のスクリーンショット。)。スキーマを参照して、「OK」をクリックします。スキーマがアクティブな JSON インスタンスドキュメントに割り当てられ、スキーマのファイル名が情報ウィンドウに入力されます。JSON インスタンスドキュメントが空の場合、JSON または Avro スキーマのインスタンスへの割り当ては、自動的に JSON インスタンスをスキーマをベースにしたサンプルデータで埋めます。

割り当てを解除するには、同じメニューから「**スキーマの削除**」を選択します（*上のスクリーンショット参照*）。

JSON *インスタンス*からの JSON *スキーマ*の生成に関する詳細は、JSON インスタンスからの JSON スキーマの生成[670] のセクションを参照してください。

インスタンス および スキーマドキュメントの検証

アクティブな JSON ドキュメント（JSON *インスタンス/スキーマ* または、Avro *スキーマ*）を検証するには、コマンド「XML | XML の検証 (F8)」を選択、または、ツールバー内の「**検証**」(F8) アイコン をクリックしてください。*インスタンス*ドキュメントが検証される場合、スキーマドキュメントが*インスタンス*に割り当てれる必要があります（*上を参照*）。検証の結果はメッセージウィンドウに表示されます。エラーは、番号の付けられている余白内でもフラグが付けられます。エラーのために*スマートフィックス*を使用することができる場合、電球アイコンがエラーを生成するライン内に表示されます。アイコンにマウスをポイントすると、使用することのできるスマートフィックスのリストのポップアップが表示されます。適用するフィックスを選択します。

メモ　上記の検証エラーインジケーターとスマートフィックスは **XML | 「検証」** (F8) コマンドが実行された場合のみ更新されます。これらは背景で更新されます。このため、エラーを修正した後、**検証** (F8) コマンドをもう一度実行し、エラーがフィックスされていることを確認する必要があります。

*インスタンス*ドキュメントからスキーマドキュメントに移動するには、情報ウィンドウ内のスキーマをダブルクリックします（*上のスクリーンショットを参照*）または、コマンド **DTD/スキーマ | スキーマに移動** を選択します。JSON キーワードのスキーマ定義、または、オブジェクトに直接移動するには、*インスタンス*ドキュメント内のキーワードまたはオブジェクトを選択し、**DTD/スキーマ | 定義に移動**を選択します。

「**検証**」コマンドを使用して、JSON ファイルを含む*プロジェクトフォルダー*[950] を検証することもできます。

変更中に検証

編集中に検証 モードはデフォルトでオンに切り替えられています。オンに切り替えられていると、JSON グリッドビュー内でのドキュメントの変更中に整形式チェックと検証チェックが実行されます。（整形式のチェックに加え）JSON ドキュメントの検証を実行するには、JSON スキーマが JSON ドキュメントに割り当てられている必要があります。エラーはエラーの存在するテキストを赤色で表示し、エラーの箇所を赤い感嘆符で表示することで表示されています。詳細に関しては JSON ドキュメントの検証[662] を参照してください。

編集中に検証 モードは (i)「XML | 編集中に検証」[1233] メニューコマンド (ii) **編集中に検証** ツールバーボタン または (iii) オプションダイアログの検証設定[1455] の *編集時* オプションによりオンとオフに切り替えることができます。

13.7 JSON フラグメントの挿入

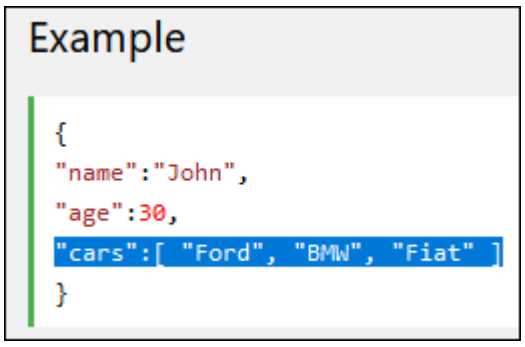
他のアプリケーションとW eb ページからJSON フラグメントを挿入することができます。2つの方法を使用してこれらのフラグメントを挿入することができます:

- ドラッグアンドドロップを使用してテキストビューとグリッドビューに挿入する。グリッドビューにドラッグアンドドロップする場合、ドラッグオーバーレイ内で使用可能なインテリジェントな情報 610 がフラグメントをドロップする箇所を決定する手助けをします。
- コピーして貼り付けを使用してテキストビューとグリッドビューに挿入する。

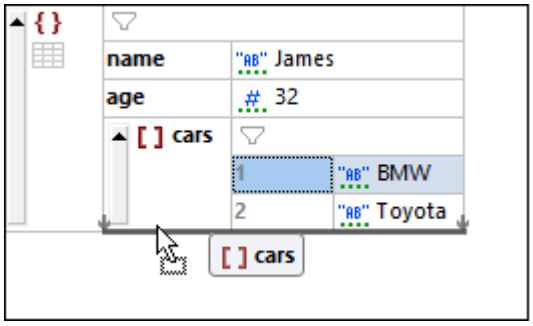
サンプル

次のサンプルはフラグメントをJSON ドキュメント内の正確な箇所に素早く追加する方法を示しています。

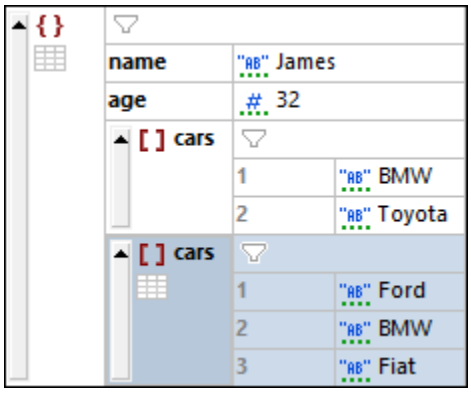
1. 下でハイライトされているフラグメントが(w3schools.com でのJSON チュートリアルから) 選択されています。cars という名前の配列



2. 下のスクリーンショットは類似しているcars 配列を含むJSON ドキュメントのグリッドビューを表示しています。W eb ページからのフラグメントが既存のcars 配列にドラッグされる場合 ドラッグされたJSON フラグメントが既存の配列の下にcars という名前の新規の配列としてドロップされる情報を含むドラッグオーバーレイ 610 が表示されます。



3. フラグメントがドロップされると希望する箇所に配置されます(下のスクリーンショット)。


{ }
name
"AB" James
age
32
[ ] cars
1
"AB" BMW
2
"AB" Toyota
[ ] cars
1
"AB" Ford
2
"AB" BMW
3
"AB" Fiat

13.8 XSLT/XQuery を使用した JSON 変換

XPath/XQuery 3.1 式を使用して JSON マップ、配列、および、オブジェクトをターゲットとすることができます。結果、JSON ドキュメントを、XSLT 3.0、XQuery 3.1、および XQuery Update 3.0 ドキュメントと共に XMLSpy の内蔵のエンジンを使用して変換することができます。

以下の機能を使用することができます:

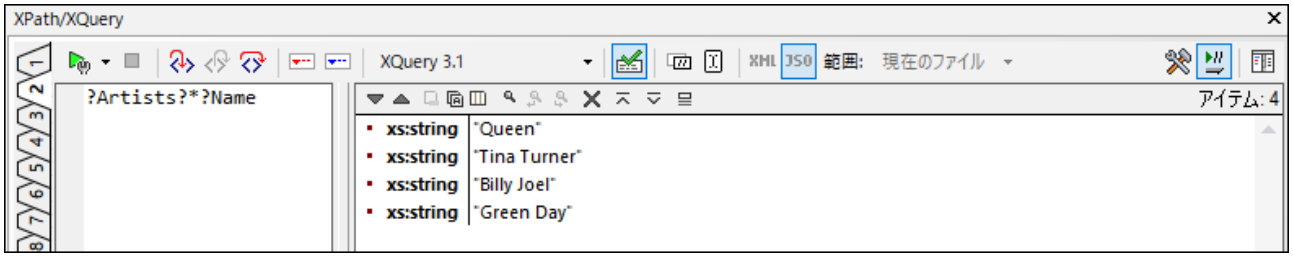
- XPath/XQuery 出力ウィンドウ[113] からアクティブな JSON ドキュメントを XPath/XQuery 3.1 式を使用してクエリ[666] することができます。
- アクティブな JSON ドキュメントをユーザーにより選択されている XSLT、または、XQuery ファイルを使用して変換[667] することができます。
- アクティブな XSLT、または、XQuery ドキュメントをユーザーにより選択されている JSON ソースファイル上で実行[667] することができます。

これらの機能は下記で詳細に説明されています。JSON のための XQuery 式作成ドキュメントに関する詳細は JSON のための XQuery 式[668] を参照してください。

メモ アプリケーションフォルダー内の **Examples** プロジェクトの *JSON サンプル* フォルダー内の JSON、XSLT、および XQuery ファイルを使用して JSON を変換することができます: C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント\Altova\XMLSpy2021\Examples.

XPath/XQuery 出力ウィンドウを使用して JSON ドキュメントをクエリする

JSON ドキュメントを XPath/XQuery 出力ウィンドウ[113] 内に XPath/XQuery 3.1 クエリ式を入力してクエリすることができます。(*下のスクリーンショットを参照*). **XPath 3.1** アイコン、または、**XQuery 3.1** アイコンを選択し、JSON 評価モード内にウィンドウが存在することを確認してください (*下記で説明されています*)。



下記の情報は JSON 評価モード内の JSON ドキュメントの評価に関しての情報です (XPath/XQuery ウィンドウの概要と使用の詳細に関しては次のセクションを参照してください: 出力ウィンドウ XPath/XQuery[113])。

JSON 評価モード
JSON 評価モードの以下の点に関して説明されています:

- XPath/XQuery ウィンドウは XML 評価モード、または、JSON 評価モードのどちらかです。現在アクティブな モードは、アクティブなモードのボタンがハイライトされることにより示されています。上のスクリーンショット内の XML/JSON 評価モード ボタンを参照してください。スクリーンショットでは、ウィンドウは JSON 評価モードに設定されています。
- 上のスクリーンショットでは、ボタンが無効化されていることを示す XML と JSON ボタンがグレーアウトされていることに注意してください。ボタンが無効化されると、有効化、または、無効化の状態を変更することはできません。反対にボタンが有効化 (グレーアウトされていません) されていると、ウィンドウの評価モードを変更することができます。
- XML/JSON 評価モード ボタンの*有効化/無効化*状態は評価のスコープにより異なります (*Where フィールドの値、上のスクリーンショット参照*)。評価スコープの値は、*有効化/無効化* 状態を決定するために 2 つのグループに分割されています: (i) 単一ファイル (*現在のファイル*)、(ii) 複数のファイル (*開かれているファイル、プロジェクト、フォルダー*)。

- (*上のスクリーンショットに表示されているとおり*) 評価スコープのために単一のファイル (*現在のファイル*) が選択されている場合、ウィンドウのモード (JSON、または、XML) はファイルの拡張子[1458] をベースにして決定されます。ファイルが JSON 適合[1458] の場合、JSON 評価モードは有効化されています。または、ファイルが JSON に適合しない場合[1458]、および、XML 評価モードがオンに切り替えられている場合。単一のファイルの型が既知なため、適切な評価モードが有効化されており、両方のボタンが無効化されているので、モードを変更することはできません。
- (*開かれているファイル、プロジェクト、フォルダー*) 複数のファイルオプションが選択されている場合、両方の評価モードボタンが有効化されており、ユーザーは有効化するモード (JSON、または、XML) を選択することができます。複数のファイルスコープのデフォルトの評価モードは XML です。
- XML 評価モードでは、XML 適合ファイル[1458] が処理され、JSON ファイルはスキップされます。
- JSON 評価モードでは、JSON 適合ファイル[1458] が処理され、XML ファイルはスキップされます。
- JSON 式をデバッグモード内でクエリすることもできます。

XSLT/XQuery を使用した JSON ドキュメントの変換

XSLT 3.0, XQuery 3.1、または、XQuery Update 3.0 ドキュメントを使用してアクティブな JSON ドキュメント[108] を変換するには、、以下を行います:

- *XSLT 3.0 変換:* 「XSL/XQuery | XSL 変換」をクリックし、を参照し、XSLT 3.0 ファイルを参照し、「OK」をクリックします。
- *XQuery 3.1、または、XQuery Update 3.0 変換:* 「XSL/XQuery | XQuery/Update の実行」をクリックし、XQuery 3.1、または、XQuery Update 3.0 ファイルを参照し、「OK」をクリックします。

変更済みのドキュメントが生成され、XMLSpy 内で直接確認することができます。

メモ　XSLT/XQuery デバッガー[467] を JSON ドキュメントから開始することができますが、ブレークポイントとトレースポイントは XSLT、または、XQuery ドキュメント内でのみで設定します。

XSLT/XQuery ドキュメントのための JSON ソースの提供

アクティブな XSLT、または、XQuery ドキュメント[108] を JSON ソースファイル上で実行するには、以下を行います:

- *アクティブな XSLT 3.0 ドキュメント:* 「XSL/XQuery | XSL 変換」をクリックし、JSON ファイルを参照し、「OK」をクリックします。
- *アクティブな XQuery 3.1、または、XQuery Update 3.0 ドキュメント:* 「XSL/XQuery | XQuery/Update の実行」をクリックし、JSON ファイルを参照し、「OK」をクリックします。

変更済みのドキュメント 変更済みのドキュメントが生成され、XMLSpy 内で直接確認することができます。

メモ　XSLT/XQuery デバッガー[467] を XSLT、または、XQuery ドキュメントから開始することができ、JSON ドキュメントをデバッグセッションのための入力として割り当てることができます。ブレークポイントとトレースポイントは XSLT、または、XQuery ドキュメント内でのみで設定します。

13.9 JSON のための XQuery 式

JSON のための XQuery 3.1 式

JSON データ構造は通常オブジェクトと配列を使用するため、 XQuery 3.1 lookup operator `?` が（基本的に XQuery パースペクティブからマップする） JSON オブジェクトと JSON 配列内のノードを検索するために使用されます。ノードをロケートするこの方法は XML ドキュメント内でノードをロケートするために パス式が作成される方法とは異なります。これらの中では、スラッシュ演算子 `/` が パス式内でステップを接続するために使用されています（例: `items/*`）。JSON のための式 XQuery 内では、スラッシュ演算子はノードをロケートするために使用されません。

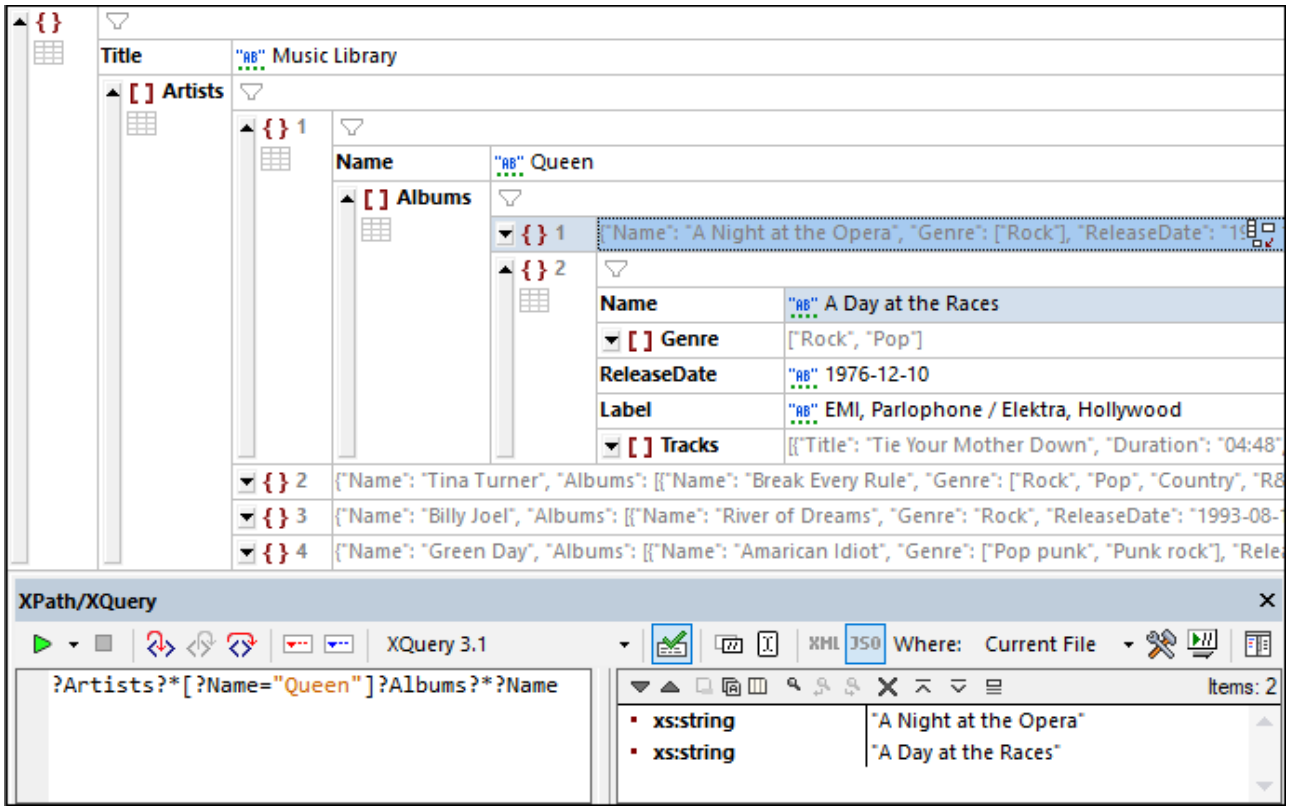
JSON のための XQuery 式のサンプル

`?items?*`
意味: 子ノード `items` をルックアップし、自信の子ノード全てをルックアップします。`items` はコンテキストノードの子ノードであることが期待されることにご注意してください。

`?Artists?1?Albums?2?Name`
意味: 子ノード `Artists` をルックアップし、最初の子ノードをルックアップします。そのノードの内部で子ノード `Albums` をルックアップし、2 番目の子ノードをルックアップします。2番目の`Name` ノードが返されます。

`?Tracks?*[contains(?Writer, 'Brian')]`
意味: 子ノード `Tracks` をルックアップし、自信のすべての子をルックアップします。子をルックアップ中に、各 `Writer` 子ノードの子をルックアップし、文字列 `'Brian'` を含むものを選択します。この式内には3 つのルックアップ演算子が存在します。ノードセットがルックアップされる個所でそれぞれが新規ステップ内で使用されています。

`?Artists?*[?Name="Queen"]?Albums?*?Name`
意味: ルートオブジェクト内で lookup 子ノード `Artists` をルックアップし、`"Queen"` の値を持つ`Name` ノードを持つすべての子をルックアップします。コレアのノード内で、全ての子 Albums ノードをルックアップし、その子もルックアップします。これらの子内で、対応する`Name` ノードをルックアップ（して返）します。下のスクリーンショットでは、この式は XPath/XQuery ウィンドウ[506] 内でターゲット JSON ドキュメントの JSON グリッドビュー[600] 表示と共に表示されています。


Title
Music Library
Artists
1
Name
Queen
Albums
"Name": "A Night at the Opera", "Genre": ["Rock"], "ReleaseDate":
2
Name
A Day at the Races
Genre
["Rock", "Pop"]
ReleaseDate
1976-12-10
Label
EMI, Parlophone / Elektra, Hollywood
Tracks
[{"Title": "Tie Your Mother Down", "Duration": "04:48"
{"Name": "Tina Turner", "Albums": [{"Name": "Break Every Rule", "Genre": ["Rock", "Pop", "Country",
{"Name": "Billy Joel", "Albums": [{"Name": "River of Dreams", "Genre": "Rock", "ReleaseDate": "1993-08-
{"Name": "Green Day", "Albums": [{"Name": "Amarican Idiot", "Genre": ["Pop punk", "Punk rock"],
XPath/XQuery
XQuery 3.1
Where: Current File
?Artists?*[?Name="Queen"]?Albums?*?Name
Items: 2
xs:string "A Night at the Opera"
xs:string "A Day at the Races"

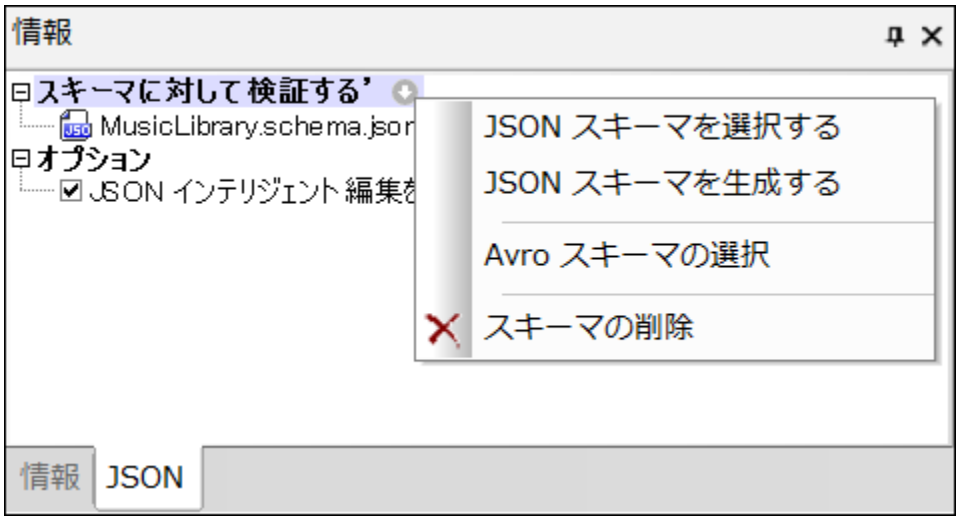
13.10 JSON インスタンスからの JSON スキーマの生成

XMLSpy は、(JSON5 *インスタンス*を含む) JSON *インスタンス*ドキュメントから JSON スキーマを生成することができます。この機能は、既存の JSON *インスタンス*をベースにしたスキーマを素早く提供し、スキーマをはじめから手動で作成する手間を省くため、とても役に立ちます。必要に応じて作成されたスキーマを修正することもできます。

スキーマの生成

JSON *インスタンス*から JSON スキーマを生成するには、以下を行います:

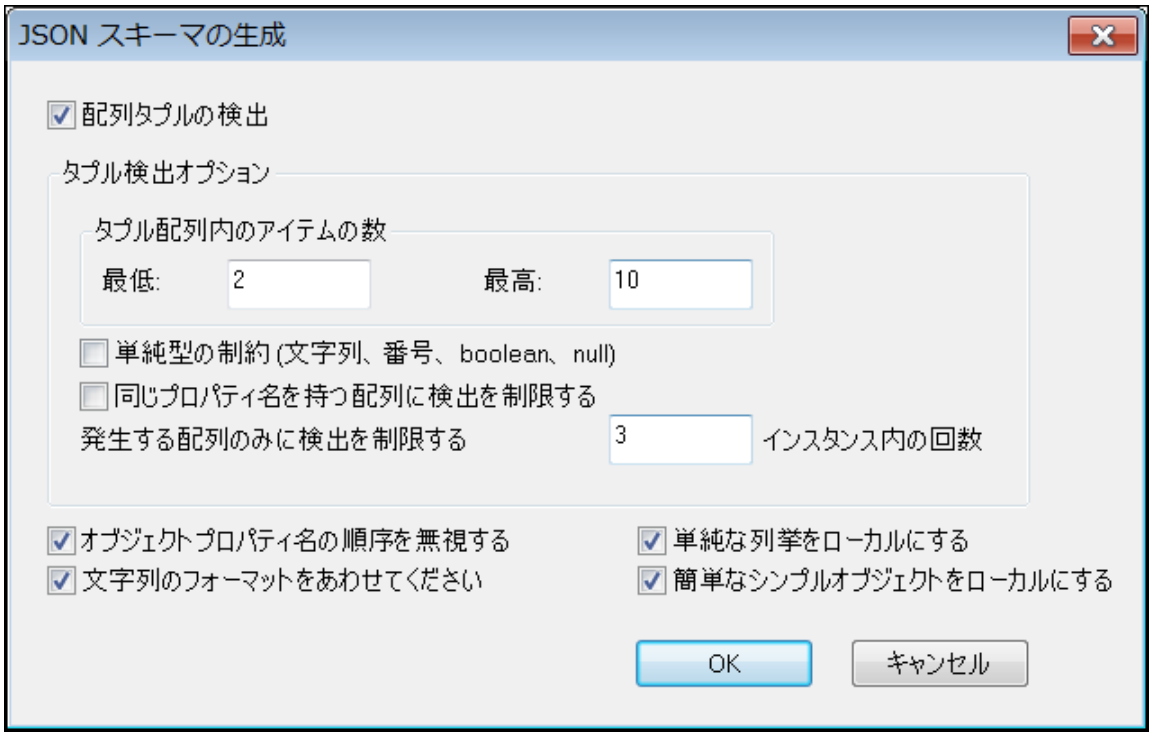
1. JSON *インスタンス*ドキュメントをアクティブなドキュメントにする。
2. 情報ウィンドウの JSON タブで、スキーマに対して検証するの横の矢印のアイコンをクリックすると、メニューが表示されます。「JSON スキーマの生成」をクリックまたは「DTD/スキーマ | DTD の生成/スキーマ」を選択します。



3. 表示された JSON スキーマの生成 ダイアログ(*下の次のセクションを参照*) 内で設定を変更します(*詳細は下を参照*)。編集が完了すると「OK」をクリックします。
4. 生成された JSON スキーマのための パスとファイル名を与えるようにプロンプトされます。「保存」をクリックすると、スキーマが生成されアクティブドキュメントになります。インスタンスドキュメント内で、生成されたスキーマファイルは検証で使用されるスキーマとして割り当てられます。以前に割り当てられたスキーマは上書きされます。割り当てを変更するには、コンテキストメニューの「JSON スキーマの選択」[662] コマンドを使用します(*上のスクリーンショット参照*)。

スキーマの生成のための設定

JSON スキーマの生成 ダイアログのスキーマの生成を指定することができます(*下のスクリーンショット*)。このダイアログにアクセスする方法の詳細に関しては、前のセクション[670]を参照してください。



配列タプルの検出
配列 タプルは配列内のアイテムのシーケンスです。例えば、次の配列には3つのアイテムを持つタプルがあります: `[ 1, 2, "abc" ]`。配列の検証のためには、スキーマは配列(タプル)の順序とデータ型を考慮するか指定することができます。*配列タプルの検出* オプションがチェックされていると、(*上のスクリーンショット参照*)、アイテムの順序とデータ型が検出されます。検出された内容に従い、スキーマ内に対応する定義が作成されます。この設定のオプションは以下のとおりです:

- *タプル アイテムの数:* 指定することのできる最少および最大のタプルアイテムの数。インスタンス内のタプルがこの範囲内のアイテムの数を持つ場合、配列は検出され定義されます。
- *単純型のみ:* 単純型アイテム(アトミック型 `string`、`number`、`integer`、`boolean`、および`null`) を持つタプルのみが検出対象となります。
- *同一の名前を持つ配列:* 同じ名前を持つプロパティの値として定義された配列のみが検出対象となります。例えば、下の JSON データフラグメント内では、赤でハイライトされたかっこでマークされた配列は(青でハイライトされた) `a1` という名前のプロパティの値です: `{"object1": [ { "a1": [ 1, 2, "abc" ] }, { "a1": [ 3, 4, "def" ] }, { "a1": [ 5, 6, "ghi" ] } ] }`。
- *配列の最少の数値:* 指定することのできる配列の検出の最少の数です。

他の設定

- *オブジェクト プロパティ名前の順序を無視する:* 選択されると、プロパティのオブジェクトの順序がチェックされ、元に限りなく近く再作成されます。
- *文字列のフォーマットに一致させる:* 文字列データ型を指定するスキーマは特別なフォーマットを持つ必要があります。このオプションが選択されていると、XMLSpy は可能な箇所で文字列フォーマットを検出およびフォーマット定義を追加します。
- *単純配列をローカルに設定する:* 単純配列は、同じ単純データ型のすべてのアイテムの1つです。選択されると、すべての単純配列は、ローカルで参照されているグローバル定義を使用する代わりにスキーマ内でローカルに定義されます。
- *単純オブジェクトをローカルに設定する:* 単純オブジェクトは、すべてのプロパティの値が同じ単純データ型をもつものです。選択された場合、ローカルで参照されているグローバル定義を使用するかわりに、すべての単純オブジェクトはスキーマ内で定義されます。

メモ JSON スキーマが生成されると、個々のオブジェクトと配列の定義のローカル定義をグローバルにまたその反対に設定することができます。詳細に関しては、グローバルとローカル定義 637 のセクションを参照してください。

13.11　JSON インスタンスを JSON スキーマから生成する

テキストビュー、グリッドビューまたはスキーマビューで JSON スキーマがアクティブなファイルの場合、JSON スキーマから JSON インスタンスを生成することができます。「DTD/スキーマ | サンプル XML/JSON ファイルを生成する」[1251] をクリックします。このコマンドは、JSON5 ドキュメントではなく JSON ドキュメントを生成することご注意してください。

13.12 JSON と XML の間の変換

以下の変換オプションを使用することができます:

- XML インスタンスを JSON に変換 [1352]: XML インスタンスドキュメントがアクティブなドキュメントの場合、JSON または JSON5 インスタンスドキュメントの生成を選択することができます。コマンド「変換 | XML インスタンスを JSON へ/から変換」[1352]を使用します。
- JSON インスタンスを XML に変換 [1352]: JSON/JSON5 インスタンスドキュメントがアクティブなドキュメントの場合、XML インスタンスドキュメントは JSON インスタンスから「変換 | XML インスタンスを JSON へ/から変換」[1352]をクリックすることで生成されます。
- XML スキーマを JSON スキーマに変換 [1355]: XML スキーマドキュメントがアクティブなドキュメントの場合、JSON スキーマドキュメントは XML スキーマから「変換 | XML スキーマを JSON スキーマへ/から変換」[1355]をクリックすることで生成されます。
- JSON スキーマを XML スキーマに変換 [1355]: JSON スキーマドキュメントがアクティブなドキュメントの場合、XML スキーマドキュメントは JSON スキーマから「変換 | XML スキーマを JSON スキーマへ/から変換」[1355]をクリックすることで生成されます。

これらの変換はテキストビューとグリッドビューで有効化されます。上のリンクをクリックするとそれぞれの機能を確認することができます。

14 Avro、Avro スキーマ

Apache Avro™ は、データをコンパクトなバイナリフォーマットにシリアル化するためのシステムです。Avro データ構造は、 JSON フォーマットで書かれている Avro スキーマで定義されており、実際のデプロイのシナリオでは、Avro ドキュメントは通常、Avro データ構造だけではなく、構造を定義する Avro スキーマを含む、バイナリファイルとしてシリアル化されています。Avro バイナリはですから、データとデータの構造の定義 (Avro スキーマ) を含んでいます。Avro データは、しかしながら、JSON としてシリアル化することができ、(JSON ファイル内の) Avro データは、外部の Avro スキーマを参照することができます。

XMLSpy は Apache Avro™ 1.8.1 をサポートします。

XMLSpy では次の Avro サポートをご利用いただけます:

- テキスト ビュー[595] と グリッドビュー[600] で、Avro データ (`.json` JSON ドキュメントとして) を編集することができます。両方の種類のビューでインテリジェントな編集機能を使用することができます。データドキュメントに Avro スキーマを割り当て、検証することができます。
- (`.avsc` Avro スキーマドキュメントとして) Avro スキーマをテキスト ビュー[595] と グリッドビュー[600] 内で編集することができます。Avro スキーマを Avro スキーマ仕様に対して検証することができ、ビューにはインテリジェントな編集機能が搭載されています。
- テーブルビューで Avro データブロックを表示する Avro ビュー[846] で Avro バイナリインスタンス (`.avro` ファイル) を確認することができます。
-

Altova の RaptorXML エディション は、Avro を更にサポートします:

- (Avro スキーマに対しての) Avro データ (JSON-シリアル化された `.json` ファイル) 検証
- Avro データ (バイナリ-シリアル化された `.avro` ファイル) 検証
- Avro スキーマ (通常 `.avsc` ファイル) の (Avro スキーマ仕様 に対して) 検証
- Avro バイナリからの Avro スキーマの抽出

既存の Avro ドキュメントを開き、新規ファイルを作成する

「オプション | ファイルの型」[1458] タブから、(*下のスクリーンショット*)、異なる型の Avro ドキュメント (JSON データフォーマット、Avro スキーマ、Avro バイナリ) が開かれるデフォルトのビューを設定することができます。使用することのできるビューを随時切り替えることができます。

ドキュメントの型	ファイル拡張子	適合性	使用することのできるビュー
JSON フォーマットのAvro データ	.json	JSON 適合 | JSON	テキストビュー、グリッドビュー
Avro スキーマ	.avsc	JSON 適合 | Avro スキーマ	テキストビュー、グリッドビュー
バイナリファイル内のAvro データ	.avro	Avro 適合| Avro バイナリ	Avro ビュー

次の点に注意してください:

- 選択された型の既存のドキュメントと新規ドキュメントは、ファイル型タブで選択されているデフォルトのビューで開かれます。
- 読み取り専用のビューであるAvro バイナリは、Avro ビュー[681]のみで確認することができます。ファイル型がAvro-適合と定義されている場合、使用することのできるビューはAvro ビュー[681]のみです。

(C) 2015-2021 Altova GmbH

- 上にリストされている Avro ドキュメントの 1 つとして、特定のファイル拡張子のファイルを XMLSpy に読み取らせる場合、子の新規ファイル拡張子を追加し、対応する適合に割り当てます。
- 新規ドキュメントを作成するには、「**ファイル | 新規作成**」クリックして、ドキュメントの型を選択します。Avro バイナリは、バイナリであるため、この方法で作成できず、Avro ビュー[681] のみで読み取ることができます。

14.1 Avro スキーマ

Avro スキーマは、Avro データブロックを指定します。どのデータフィールドが期待され、どのように値が示されるかを指定します。Avro スキーマに関する情報、とその仕様はここで確認することができます。

Avro スキーマに関する以下の点に注意してください:

- Avro スキーマを JSON フォーマット内で作成
- Avro スキーマは以下であることができます: JSON 文字列、JSON オブジェクト、または、JSON 配列
- Avro スキーマは、以下の4つの属性を含むことができます: **name**、**namespace**、**type**、と**fields**
- 以下の8つのプリミティブなデータ型が存在します: **null**、**boolean**、**int**、**long**、**float**、**double**、**bytes**、と**string**
- 以下の6つの複合型が存在します: **records**、**enums**、**arrays**、**maps**、**unions**、と**fixed**
- プリミティブな型には属性が存在しません。また、各複合型には、自身の属性のセットが存在します。

Avro スキーマに関する更に詳しい情報は、Avro スキーマ仕様を参照してください。

サンプル

以下に、Avro スキーマの簡単なサンプルが表示されています。それぞれが JSON フォーマットの Avro データスニペットに対応しています。スキーマは特定の構造を定義することに注意してください。一部の場合、定義された構成が複数回インスタンス化される場合があり、結果の出力は、有効ではない JSON になる可能性があります。例えば、スキーマが JSON オブジェクトの構造を定義する場合があります。JSON オブジェクトが複数回インスタンス化されると、各オブジェクトは、個別に、Avro スキーマに対して有効ではあるが、コンテナーオブジェクトが存在しないため、ドキュメント全体は、有効な JSON ではない場合があります。有効な JSON が必要な場合、JSON オブジェクトの配列を検証するために Avro スキーマを書き直す必要があるかもしれません。サンプル 4 と 5 を比べて、上記の点を確認してください。

01: JSON 文字列としての Avro スキーマ

このスキーマは、単一の文字列で、データブロックが Avro (int) プリミティブデータ型 **"int"** の値を持つように指定します。
有効な Avro: **2016**
無効な Avro: **"2016"**

02: JSON オブジェクトとしての Avro スキーマ

このスキーマは、前のスキーマと同じ内容を指定しますが、JSON オブジェクトです。データブロックは Avro (int) プリミティブ データ型の値であるアイテムを1つ含む必要があります。

```
{
   "type": "int"
}
```

有効な Avro: 2016
無効な Avro: "2016"

03: JSON オブジェクトとしての Avro スキーマ: 整数の配列

このスキーマは、整数の配列を指定する JSON オブジェクトです:

```
{
   "type": "array",
   "items": "int"
}
```

有効な Avro: `[2016, 2017]`
有効な Avro: `[2016]`
有効な Avro: `[2016]`
無効な Avro: `2016, 2017`

04: JSON オブジェクトとしての Avro スキーマ: レコード

このスキーマは、単一のレコードを指定する JSON オブジェクトです:

```
{
   "type": "record",
   "name": "ages",
   "fields" : [
   {"name": "name", "type": "string"},
   {"name": "age", "type": "int"}
   ]
}
```

有効な Avro: `{"name":"John", "age":35}`

05: JSON オブジェクトとしての Avro スキーマ: 複数のレコード

このスキーマは、レコードの配列を指定する JSON オブジェクトです。それぞれは JSON オブジェクトである必要があります:

```
{
   "type": "array",
   "items": {
   "type": "record",
   "name": "ages",
   "fields" : [
   {"name": "name", "type": "string"},
   {"name": "age", "type": "int"}
   ]
    }
}
```

有効な Avro: `[{"name":"Mary", "age":34}, {"name":"John", "age":35}]`

Avro スキーマ ファイル型

XMLSpy の機能を Avro に関連した、編集と検証で使用する場合、XMLSpy は、ファイルを Avro スキーマとして認識する必要があります。ファイルは、ファイルの拡張子が XMLSpy のオプションダイアログ(「ツール | オプション | ファイル型」1458)で定義されているとおり Avro スキーマとして識別されます。XMLSpy のデフォルトの設定は、1 つのファイル拡張子675 — `.avsc` 拡張子 — を Avro スキーマとして指定しています。Avro スキーマドキュメントを指定する他のファイル拡張子を作成する場合、675 これらのファイル拡張子を Avro スキーマ拡張子として、オプションダイアログ1453 内のリストに追加します。

Avro スキーマの作成と編集

XMLSpy 内では、Avro スキーマ ファイル拡張子をファイルの型と指定することにより、Avro スキーマとして、新規のファイルを作成することができます1123。XMLSpy は、インテリジェントな編集ヘルプを入力時に提供します。この機能にはコンテキストに敏感なキーワードの提案、自動かっこ、引用符の入力、構文の色分け、キーワードの自動補完などが含まれています。更に、3 つの入力ヘルパーが搭載されています: JSON プロパティ、JSON 値、および JSON エンティティ。使用することのできるエントリは、コンテキストに敏感です。挿入するエントリをカーソルが現在ある場所でダブルクリックします。「**検証** | XML の検証(F8)」メニューコマンドを利用して、ファイルを Avro スキーマ仕様に対して検証 することができます。

14.2 JSON フォーマットの Avro データ

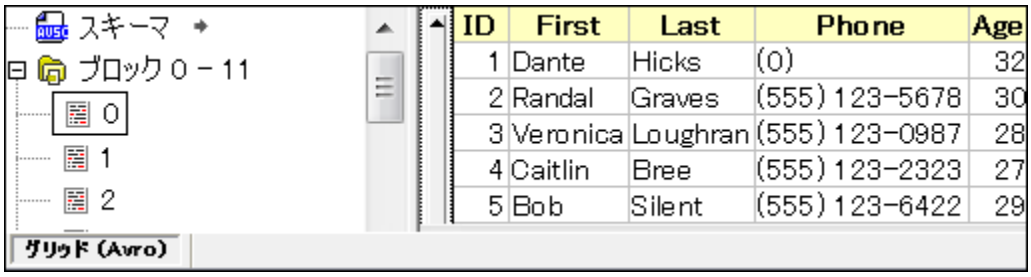
Avro データは、バイナリフォーマット または JSON フォーマットでシリアル化されることができます。以下のポイントは、XMLSpy の Avro フォーマットのサポートについて説明しています。

- JSON フォーマットの Avro データは、通常、`.json` ファイルで保存されます。XMLSpy が追加ファイル拡張子を JSON 適合であると認識[675]することを指定できます。.
- Avro JSON ファイルは、テキスト ビュー[595] とグリッドビュー[600] で開くことができ、これらのビューで編集することができます。
- Avro スキーマ ファイルを Avro JSON ファイル[662] に割り当てることができ、データファイルを Avro スキーマに対して検証することができます[662]。
- JSON ドキュメントのためのインテリジェントな機能は、テキスト ビュー[595] とグリッドビュー[600] の両方で使用することができます。更に、Avro スキーマが JSON フォーマット内の Avro データドキュメント に割り当てられている場合、Avro インスタンス内でスキーマにより定義されたキーワードの自動補完[595] を使用することができます。

(C) 2015-2021 Altova GmbH

14.3 Avro ビュー: Avro バイナリのグリッドビュー

Avro データをバイナリフォーマット または JSON フォーマットにシリアル化することができます。バイナリフォーマットは、Avro データ構造 と スキーマの両方を含み、自動化されたデータ処理の過程で生成されます。Avro バイナリは、簡単に閲覧できるテーブルフォーマット内の Avro データ構造を表示する、グリッドビューの Avro ビュー内で開くことができます（*下のスクリーンショット参照*）。Avro ビューは、ですから、ユーザーが使いやすい Avro バイナリビューアーとしての役割を果たします。

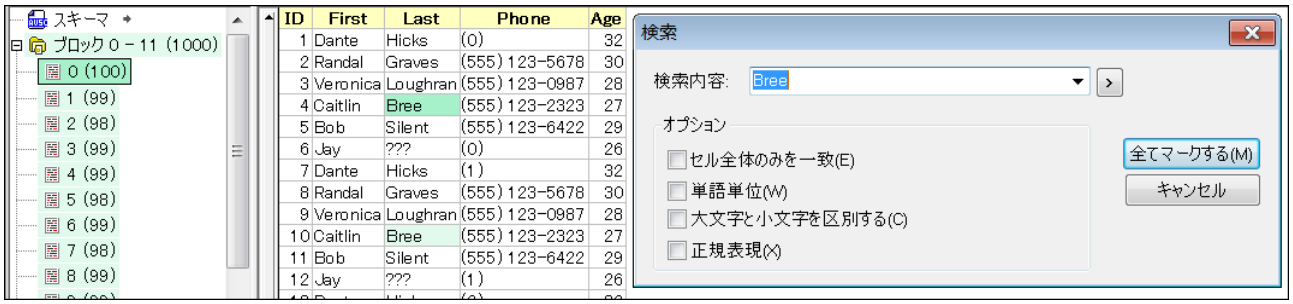


以下の点に注意してください：

- Avro バイナリは、XMLSpy などにより識別される必要があります。これは、「オプション | ファイル型」[1458] タブで Avro バイナリのファイル拡張子を Avro 適合に設定して[675] 行うことができます。デフォルトでは、ファイル拡張子 `.avro` は、Avro に適合するように設定されています。Avro バイナリに適合するファイルの型を追加することもできます。[675] これらのファイルは、Avro ビューで開かれます。
- 2 つのペインから構成される Avro ビュー: (i) ナビゲート用の*ブロック*ペイン。および (ii) ブロックペイン内で選択するデータ構造を表示する *データ*ペイン。
- *ブロック*ペインは、*データブロック*を 1000 *のグループ*に整理します。それぞれのグループは、展開および折りたたむことができます。データ ブロックは、インデックス番号別に表示されています。
- 特定のデータブロックを確認するには、*ブロック*ペイン内で検索し、ダブルクリックします。
- ブロック ペイン は、スキーマと呼ばれるエントリを含みます。エントリの右横のボタンをクリックすると、Avro スキーマが Avro バイナリから抽出され、新しいテキストビュータブで開かれます。Avro スキーマを保存することもできます。

テキストの検索

テキスト文字列を検索するには、メニューコマンド「**編集 | 検索**」(Ctrl+F) を選択します。表示される*ダイアログ*に、検索用語をテキスト文字列、または、正規表現として入力します（*下のスクリーンショット参照*）。（[ここ][1159] で説明されている）適用することのできるオプションを選択し、「**全てをマークする**」をクリックします。



- *ブロック*とデータ ペイン内で一致するエントリは ハイライトされます: 現在選択されている項目は、深緑色で表示され、その他は薄い緑で表示されています。
- *ブロック*ペイン 内では、各ブロック内の数は、エントリの横に表示された数と一致します。

- 一致する項目をブロックごとに確認、ブロック内のフィールドを選択して、または、**F3** (**編集 | 次を検索**) と**Shift+F3** (前を検索) を使用してナビゲートできます。
- Avro ビューは読み取り専用です。Avro バイナリ内のデータを編集することはできません。

15 WSDL と SOAP

Altova ウェブサイト : W SDL エディター
このセクションではXMLSpy のW SDL ならびにSOAP 機能を説明します。

WSDL

W SDL ドキュメントは、ウェブサービスの記述を行うXML ドキュメントです。XMLSpy ではW SDL 1.1 ならびにW SDL 2.0 がサポートされ、W SDL ビューにてW SDL 1.1 ならびに2.0 ドキュメントの作成や編集を行うことができます。W SDL ビューでは、W SDL のバージョンに関係なく、適切な編集環境が提供されます。

XMLSpy のW SDL ビューでは、グラフィカルな構築ブロックを使用しながらW SDL ドキュメントの構築を行うことができ、作成作業の大幅な簡易化を行うことができます。W SDL ビュー231 に関する記述は編集ビュー126 の対応するセクションを参照ください。W SDL ドキュメントの作成方法に関しては、このドキュメント内のW SDL チュートリアル684 を参照ください。W SDL ドキュメントの閲覧や編集はテキストビュー130 ならびにグリッドビュー149 で行うことができます。両方のビューで、W SDL ドキュメントは通常のXML ドキュメント261 として扱われます。

W SDL ビューで表示されるW SDL ファイルに対するXML 署名370 は独立したファイルとして作成することができ、W SDL ファイル内に含めることができます("enveloped")。署名の詳しい使用方法については、XML 署名370 のセクションを参照ください。

SOAP

SOAP はXML のメッセージ交換プロトコルで、アプリケーション間のメッセージ交換に使用されます。XMLSpy では、XML ドキュメント261 の編集機能を搭載したテキストビュー130 やグリッドビュー149 にてSOAP ドキュメントを作成、編集できるだけでなく、W SDL ファイルからSOAP リクエストファイルを作成することもできます。W SDL ファイルからSOAP リクエストファイルを作成する方法はW SDL チュートリアル684 に記載されています。(SOAP メニュー1380 のコマンドを使用することで)XMLSpy はSOAP リクエストの送受信を行うことができます。更に、このセクション内のサブセクションで解説されるXMLSpy のSOAP デバッガー695 を使用することでSOAP リクエストのデバッグを行うこともできます。

15.1 WSDL チュートリアル

このチュートリアルは2つのパートから構成されます:

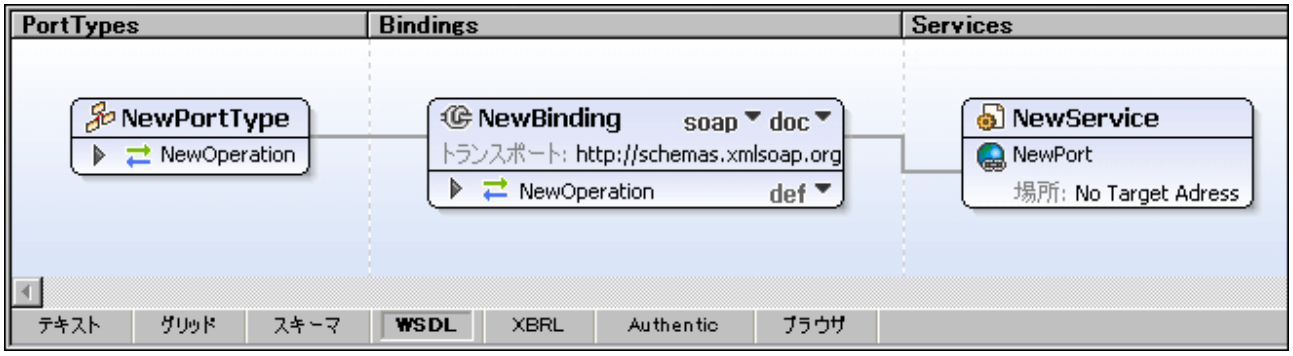
- 最初のパートでは、XMLSpy のグラフィカルな WSDL ビューを使って WSDL 1.1 ドキュメントを作成する方法が記述されます。このパートでは以下のような操作を行ないます、(i)「ファイル | 新規作成」メニューオプションを使った基本的な WSDL 1.1 ドキュメントの作成、(ii) ポートタイプの作成、(iii) バインディングの作成、(iv) サービスとポートの作成、(v) ドキュメントの検証と保存。
- 2番目のセクションでは、ウェブサービスへの接続、WSDL ファイルのローカルでの保存、そしてウェブサービスに SOAP リクエストを送信する方法を説明します。

これらの操作は全てグラフィカルな WSDL ビュー[231]にて行うことができ、テキストビューを使用する必要はありません。ドラッグアンドドロップにより WSDL コンポーネントの操作を行ったり、WSDL ビューの入力ヘルパーを使用したプロパティ値の変更を行うことができます。

関連項目: WSDL ドキュメント処理の情報については、WSDL ビュー[231]ならびにユーザーレファレンス | WSDL メニュー[1367]を参照ください。

15.1.1 新規ドキュメントの作成

新しい WSDL ドキュメントを作成するには「ファイル | 新規作成」コマンドを選択してください。新規ドキュメントを作成ダイアログにて、WSDL (WSDL Web Service Description v1.1) を作成するドキュメントの種類として選択し、「OK」をクリックします。この操作により新たなドキュメントのスケルトンが WSDL ビューにて作成されます (以下のスクリーンショットを参照)。



ターゲット名前空間の割り当て

テキストビューへ画面を切り替えます。ws 要素の開始タグは以下のようになっているはずです:

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
3       xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
4       xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
5       xmlns:xs="http://www.w3.org/2001/XMLSchema"
6       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
7       xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
8       xmlns:tns="http://new.webservice.namespace"
9       targetNamespace="http://new.webservice.namespace">
10
```

ターゲット名前空間属性を `http://mywebservice.namespace` または好みの名前に変更します(このチュートリアルではWSDLドキュメントの作成に重点をおいており、実際にサービスを作成するわけではありません)。その他にも、`tns` 属性の値を `http://mywebservice.namespace` (または、ターゲット名前空間に設定した値)に変更します。

メモ: スタート時点のスケルトンドキュメントでは、WSDL 要素がターゲット名前空間内にあり、WSDL 要素への参照は `tns` プレフィックスを使用して行われます(例: `<wsdl:binding name="NewBinding" type="tns:NewPortType">`)。`tns` プレフィックスをターゲット名前空間にマッチさせるには、その名前空間の値もターゲット名前空間と一致させる必要があります。

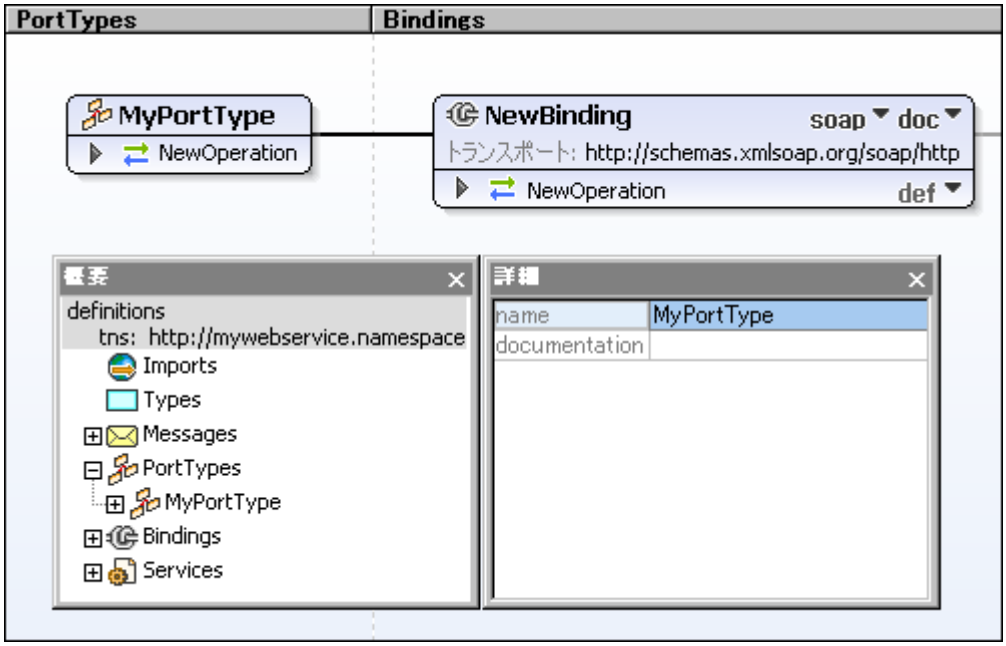
15.1.2 ポートタイプの作成

ポートタイプの作成には以下の操作が含まれます:

- ポートタイプの名前作成
- オペレーションの追加
- 入力ならびに出力メッセージの作成
- メッセージへパラメーターを追加

ポートタイプの名前作成

デザインにある `NewPortType` ボックスのタイトルバーをダブルクリックして、`NewPortType` を `MyPortType` に変更した後、「Return」により確定します。概要ならびに詳細入力ヘルパー内に表示されるポートタイプの名前も変更されていることに注目してください(以下のスクリーンショット参照)。



オペレーションの挿入

`MyPortType` には既に `NewOperation` という名前のオペレーションが存在しているので、このオペレーションを使用することにします。`NewOperation` という名前を、例えば `EchoString` に変更します(名前をダブルクリックした後「Return」により確定)。ポートタイプ

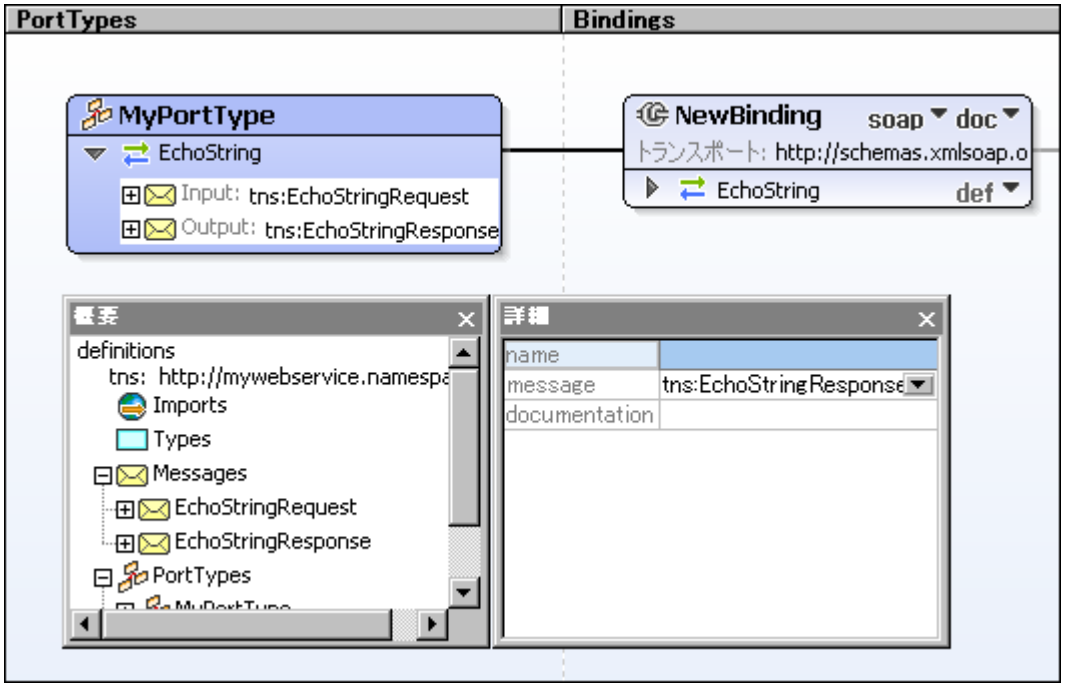
にオペレーションを追加するには、ポートタイプボックスを右クリックして、「**オペレーションを追加**」を選択し、目的の種類に沿ったオペレーションをクリックします。

入力ならびに出力メッセージの追加

オペレーションがポートタイプに追加されていれば、オペレーションの種類を以下のうつの候補から選択することができます:

- 応答要求
- 送信請求/応答
- 一方向
- 通知
- 空のオペレーション

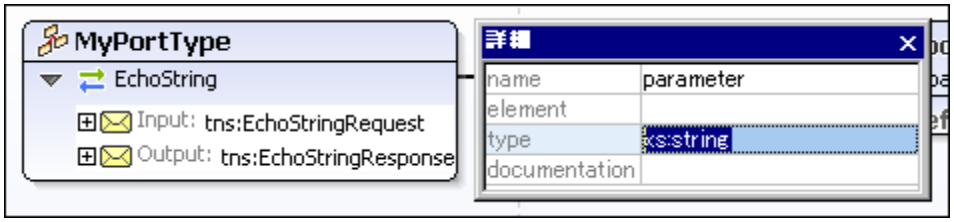
各オペレーションの種類に対して入出力メッセージが自動的に追加されます。「**空のオペレーション**」が指定されたときには、右クリックして挿入するメッセージを選択することができます。メッセージを右クリックして、「**入力/output/fault 要素を削除**」を選択することでメッセージを削除することができます。EchoString オペレーションの例では、入力 メッセージを `EchoStringRequest` へ、Output メッセージを `EchoStringResponse` へそれぞれ変更します。



メッセージへ パラメーターを追加

各 Input または Output メッセージは、デフォルトで追加される単一の `xs:string` メッセージ(または パラメーター)と共に作成されます。パラメーターの追加を行うには、メッセージまたは パラメーターを右クリックし、「**メッセージ パート (パラメーター)**」を追加を選択します。

(C) 2015-2021 Altova GmbH



パラメーターの編集を行うには、(i) テキストをダブルクリックして編集するか、(ii) パラメーターを右クリックして、「編集」を選択するか、(iii) 詳細入力ヘルパーを使用(上のスクリーンショットを参照)します。

15.1.3 バインディングの作成

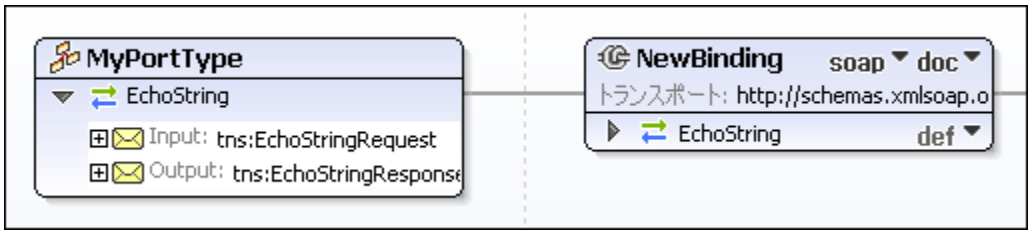
バインディングは特定のポートタイプに対するデータ形式を指定したプロトコルです。従って、バインディングの作成には、以下のような操作が伴います:

- ポートタイプへバインディングを割り当て
- バインディングのプロトコルとデータ形式の指定
- バインディングをポートへ関連付け

バインディングをポートへ割り当てる

(デザイン内の何も無い箇所を右クリックして、「バインディングを挿入」を選択するなどして) 新しいバインディングが作成されたとき、割り当てられているポートはありません(以下のスクリーンショットを参照)。新規 WSDL ドキュメントを作成した場合、デフォルトで作成されたバインディングがデフォルトで作成されたポートタイプに割り当てられ、2つのボックスが線で接続されます。

ポートタイプにバインディングを割り当てるには、(上のスクリーンショットにあるように) 新規バインディングボックスにてポートタイプエントリの下矢印をクリックします。ドキュメント内で定義されているポートタイプのリストが表示されるので、バインディングの割り当てを行ないたいポートタイプを選択します。ポートタイプにバインディングが割り当てられた場合、以下にあるようにポートタイプとバインディングのボックス間を結ぶ線として割り当てが表示されます。



プロトコルとデータ形式の選択

バインディングのプロトコルは、バインディングボックスのタイトルバーにある下向き矢印から選択することができ、4つの中からプロトコルを選択します: `SOAP`、`SOAP 1.2`、`HTTP-GET`、`HTTP-POST`。SOAP 1.1 または SOAP 1.2 プロトコルが選択された場合、データ形式として（プロトコルリストの右側の下向き矢印から表示される）`document` または `rpc` を選択することができます。

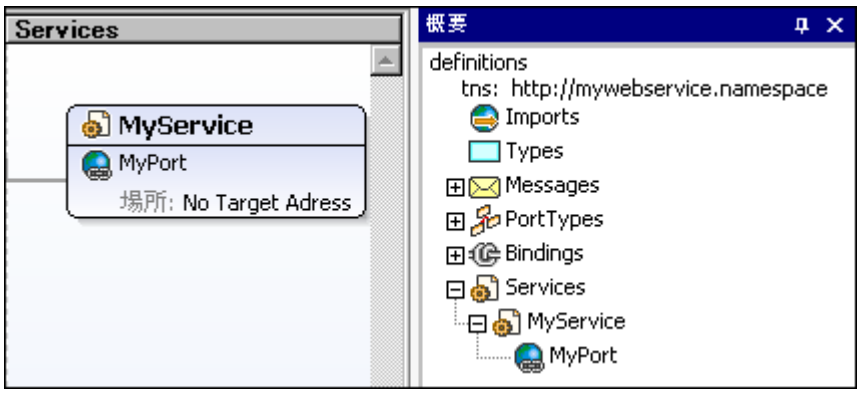
バインディング内部にある各オペレーションは（上のスクリーンショットにあるように）デザインまたはオペレーションを選択時に詳細入力ヘルパーにて定義することができます。

バインディングにポートを割り当てる

バインディングにポートを割り当てるには、まずポートを定義する必要があります。ポートの作成とバインディングの割り当てに関してはサービスとポートの作成 688 セクションにて記述されます。

15.1.4 サービスとポートの作成

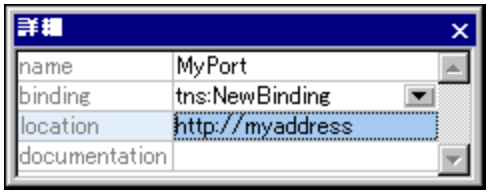
新しいサービスを追加するには、デザインの何も無い箇所を右クリックして、コンテキストメニューから「**サービスを挿入**」を選択します。新規 WSDL ドキュメントを作成した場合、デフォルトでサービスがデザイン内に表示されます。サービス名をダブルクリックし、名前を編集後「Return」を押下することでサービスの名前変更を行うことができます。サービスの名前は概要入力ヘルパーでも変更されることに注目してください（以下のスクリーンショットを参照）。



概要入力ヘルパーにおいて名前の変更を行うには、`NewPort` エントリをダブルクリックして、`MyPort` と変更した後「Return」により確定します。デザイン内のポート名も変更されることに注目してください（上のスクリーンショットを参照）。ポートを追加するには、サービスまたはポートを右クリックし、コンテキストメニューから「**ポートを挿入**」を選択します。

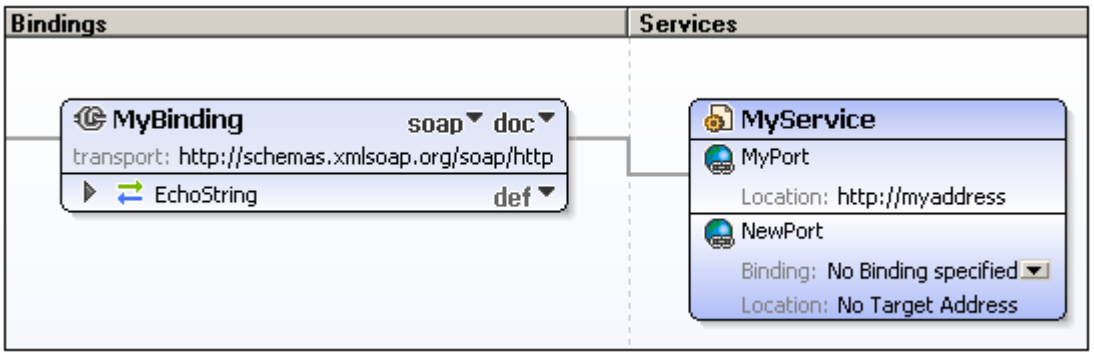
ポートのアドレスを入力

ポートのアドレスは、(i) デザインの場所アイテムから直接、または (ii) (以下のスクリーンショットにあるように) 詳細入力ヘルパーから (Location フィールドをダブルクリックして) 編集することができます。



ポートをバインディングに割り当てる

ポートはバインディングとネットワークアドレスを結びつけるエンドポイントです。ポートのアドレスが一度定義されると、必要な作業はそのポートをバインディングに割り当てるだけとなります。ポートをバインディングに割り当てるには、ポートのバインディングアイテムにある下向き矢印をクリックして、ドキュメント内で定義されているバインディングのリストから選択を行ないます。



注意: ポートが既にバインディングに割り当てられており、他のバインディングへの割り当てを行ないたい場合、まず (ポートの右クリックメニューから) バインディング参照を削除した後、新規バインディング参照を追加する必要があります。

15.1.5 WSDL ドキュメントの検証

WSDL ドキュメントを完成させると、「XML | XML 検証 (F8)」コマンドによりドキュメントの検証を行うことができます。検証の結果はメッセージ ウィンドウ内に表示されます (以下のスクリーンショットを参照)。

検証により、発見されたエラーに関する詳しい情報が表示され、エラーの場所を素早く見つけることができます。

15.1.6 ウェブサービスへの接続とファイルを開く

このセクションでは以下の操作方法について説明します:

- XMLSpy を使ったウェブサービスへのアクセス
- XMLSpy を使って WSDL ファイルを開く
- XMLSpy を使って WSDL ドキュメントに割り当てられた XML スキーマを開く

ウェブサービスへのアクセス

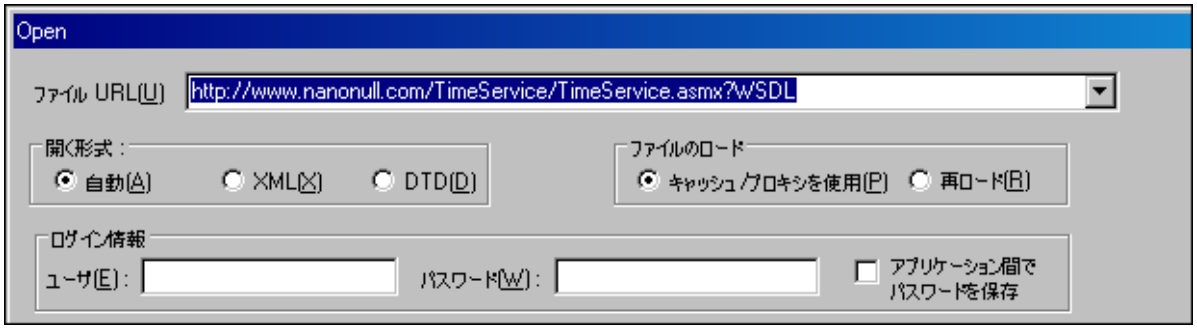
通常ウェブサービスへはHTML ページを通じてアクセスが行われます。このようなページの例の1つとして `Examples/SOAP Debugger` にある `DebuggerClient.htm` があります。このページにより表示されるウェブサービスにアクセスするには:

1. プロジェクトウィンドウが表示されていなければ、ウィンドウを有効にします(「**ウィンドウ | プロジェクトウィンドウ**」)。
2. `SOAP Debugger` フォルダーにある展開アイコンをクリックし、`DebuggerClient.htm` をダブルクリックします。この操作により、メインウィンドウにSOAP Debugger Example Client が表示されます。

XMLSpy を使って WSDL ファイルを開く

ウェブベースの WSDL ファイルを XMLSpy で開くには:

1. メニューオプション「ファイル | 開く」を選択し、ファイルを開くダイアログにて「URL にスイッチ」ボタンをクリックします。アドレス `http://www.nanonull.com/TimeService/TimeService.asmx?WSDL` をダイアログのファイル URL フィールドに入力またはコピーします。



2. 「開く」をクリックして WSDL ファイルをロードします。WSDL ファイルがテキストビューで表示されます。
3. メニューオプションから「ファイル | 名前を付けて保存」を選択し、ファイルをローカルに、例えば `timeservice.wsdl` という名前で保存します。
4. WSDL ビュータブをクリックして、ファイルをグラフィカルな WSDL エディターで表示します。

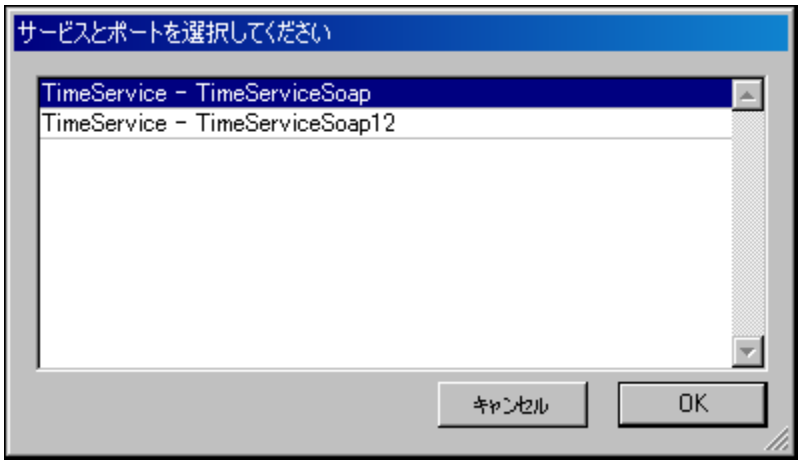
アクティブな WSDL ファイルに割り当てられたスキーマを閲覧する

`timeservice.wsdl` ファイルを WSDL ビューにてアクティブなドキュメントとして選択している場合、メニューオプションから「WSDL | 型 | スキーマビューでスキーマを編集」を選択します。この操作により `timeservice.wsdl` ファイルで使用されている全てのデータ型が定義されたスキーマが開かれます。スキーマの修正や、変更の保存を行うことができます。これらの変更は WSDL ファイルが再度分析される際に反映されます。

15.1.7 WSDL ファイルから SOAP リクエストを送信

`timeservice.wsdl` ファイルから SOAP リクエストを送信するには以下を行います:

1. **`timeservice.wsdl`** をメインウィンドウでアクティブなファイルとして選択します。
2. メニューオプションから「SOAP | 新規 SOAP リクエストを作成」を選択します。
3. **`timeservice.wsdl`** を選択して、「OK」で確定します。
4. ドキュメント内で定義されたサービスにて、SOAP 1.1 または 1.2 バインディングを参照するポートが2つ以上ある場合、(*以下のスクリーンショット*) ポップアップが表示され、目的のサービスとポートを選択するよう促されます。選択を行い、「OK」で確定します。



5. その後表示されるダイアログ(*下のスクリーンショット*)では、SOAP オペレーション(例えば`getServerTime`)を選択して「OK」をクリックします。

この操作により`getServerTime` オペレーションを含むSOAP リクエストドキュメントが作成されます。必要に応じてドキュメントを保存することもできます。

6. 目的のドキュメントがアクティブな状態でメニューオプションから「SOAP | サーバーにリクエストを送信」を選択します。Nanonull.`com` タイムサービスから得られた現在のサーバーの時間である`getServerTimeResult` 要素を含むSOAP レスポンスドキュメントがメインウィンドウに表示されます。

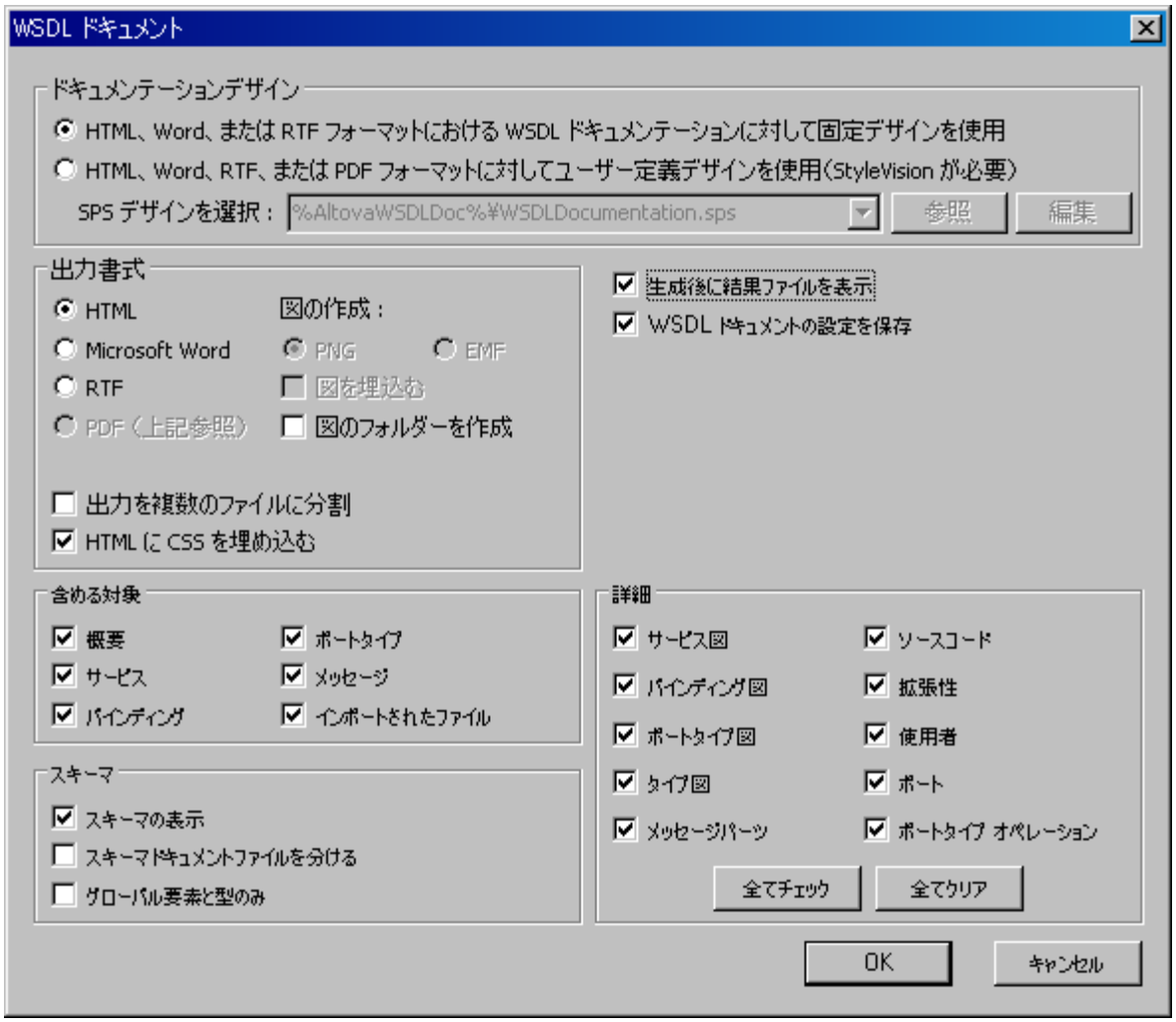
15.1.8 WSDL ドキュメンテーションの作成

「WSDL | ドキュメンテーションの生成」オプションを使えば、現在のWSDL ドキュメントに関する詳細なドキュメンテーションを生成することができます。生成されるドキュメンテーションはHTML、Microsoft Word、またはRTF の形式から選択し、またドキュメンテーションに含まれるコンポーネントも選択することができます。関連するWSDL 要素同士は生成されたドキュメンテーション内でハイパーリンク化され、ナビゲーションを簡単に行うことができます。

注意: Microsoft Word の形式でドキュメンテーションを生成するには、Microsoft Word (バージョン2000 以降)がインストールされている必要があります。

WSDL ファイルのドキュメンテーションを生成するには:

1. timeservice.wsdl をアクティブなドキュメントにします。
2. **WSDL** ビューに切り替えます。
3. メニューオプションから「W SDL | ドキュメンテーションの生成」を選択します。
   この操作により(以下のスクリーンショットにある) W SDL ドキュメントダイアログが起動します。
4. HTML、Microsoft W ord、RTF から生成したい出力ファイルの形式を選択します。
5. ドキュメンテーションに含めたい W SDL コンポーネントを選択し、その他のオプションを指定します(以下の W SDL ドキュメンテーションオプションを参照ください)



6. 「OK」をクリックして、名前を付けて保存ダイアログボックスにて W SDL ドキュメンテーションファイルの名前を入力します。

W SDL ドキュメンテーションオプション

以下のドキュメンテーションオプションを指定することができます:

- ドキュメンテーションの形式は出力書式ペインにより指定されます。ドキュメンテーションは単一のファイル、または複数のファイルに分割して生成することができます。複数のファイルが生成された場合、各ファイルが各コンポーネントに対応します。ドキュメンテーションに含まれるコンポーネントの種類は、含める対象ペインにより指定されます。
- 図を書き込むオプションは Microsoft W ord ならびに RTF 出力オプションに対して有効になります。このオプションにチェックを入れると、ダイアグラムファイルは PNG または EMF ファイルとして出力ファイル内に埋め込まれます。チェックが入れられていない場

合、ダイアグラムはPNG またはEMF ファイルとして作成され、オブジェクトリンクにより出力ファイルから参照されます。出力形式がHTML の場合、全てのダイアグラムはPNG ファイルとして作成され、ドキュメントから参照されます。

- 含める対象ペインでは、ドキュメンテーションに含めるアイテムを選択します。概要オプションにより、ファイルの最初に全てのコンポーネントがその種類ごとに表示されます。インポートされたファイルオプションにチェックが入っている場合、インポートされたファイル内にあるコンポーネントがスキーマドキュメンテーションに含められます。
- スキーマペインでは、ファイル内にあるスキーマのレポートを生成するか選択することができます。スキーマのレポートを生成するよう選択した場合、更に(i) スキーマのドキュメンテーションを独立したファイルとして生成するか、またはメインドキュメンテーションファイル内部に含めるか、そして(ii) スキーマ全体のレポートを生成するのか、グローバル要素、単純型、複合型だけを生成するのか選択することができます。
- 詳細ペインでは、各コンポーネントに含める対象が表示されます。ドキュメンテーションに加えたい項目を選択してください。
- 「生成時にファイルを表示」オプションにより、(HTML 出力の場合は) ブラウザービュー、(Microsoft W ord の場合は) Microsoft W ord、(RTF ファイルの場合は) `.rtf` ファイルに割り当てられたデフォルトアプリケーションで出力ファイルが開かれます。

15.1.9 WSDL 2.0 への変換

XMLSpy を使えば、既存のW SDL 1.1 ドキュメントを、簡単にW SDL 2.0 フォーマットへ変換することができます。以下のように、`TimeService.wsdl` を例にしてみましょう:

1. `TimeService.wsdl` をアクティブなファイルにします。このファイルはXMLSpy に同封されている `Examples` プロジェクトの WSDL エディター フォルダー以下にあります。
2. 「W SDL | W SDL 2.0 に変換」コマンドを選択します。
3. 名前を付けて保存ダイアログにて、W SDL 2.0 ファイルに使用する名前(例えば `TimeService20.wsdl`)を入力します。
4. 新たなファイルが作成、自動的に検証され、W SDL ビューにて表示されます。

15.2 SOAP

Altova ウェブサイト : SOAP デバッガー
このセクションでは以下の事柄について説明します:

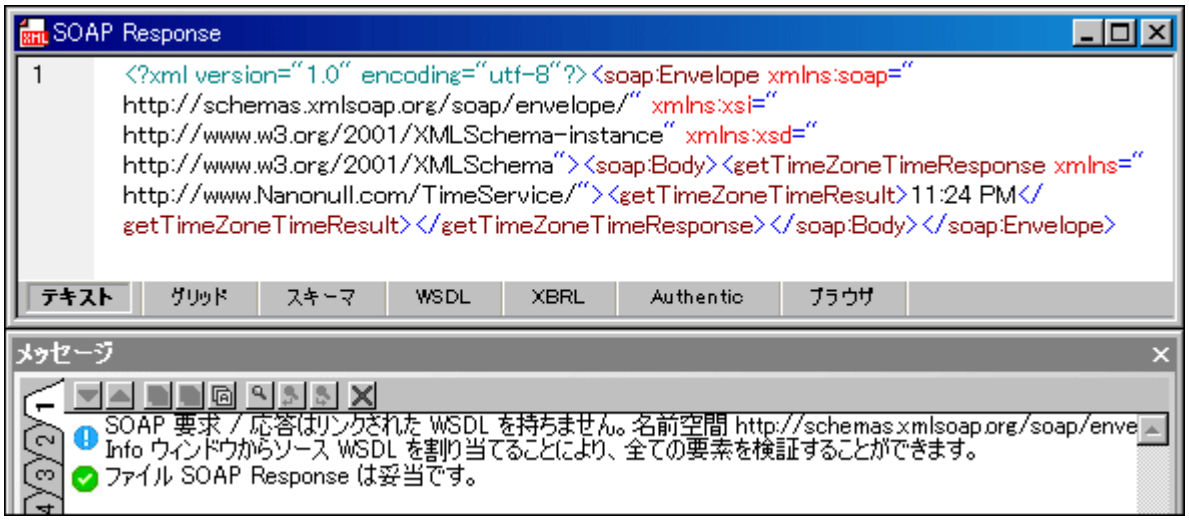
- W SDL に対する SOAP メッセージの検証。SOAP メッセージの検証を、SOAP 仕様だけではなく、W SDL 定義により参照されている任意の XML スキーマに対して行うことができます。
- SOAP デバッガーによる SOAP リクエストの送受信
- SOAP リクエストの送受信にブレークポイントをセットする
- ウェブサービスに送信する前に、不正な SOAP リクエストを編集する

15.2.1 SOAP 検証

SOAP 仕様だけではなく、対応する W SDL 定義により参照されている全ての XML スキーマに対しても、SOAP メッセージの検証を行うことができます。

SOAP ルールに対してのみの検証

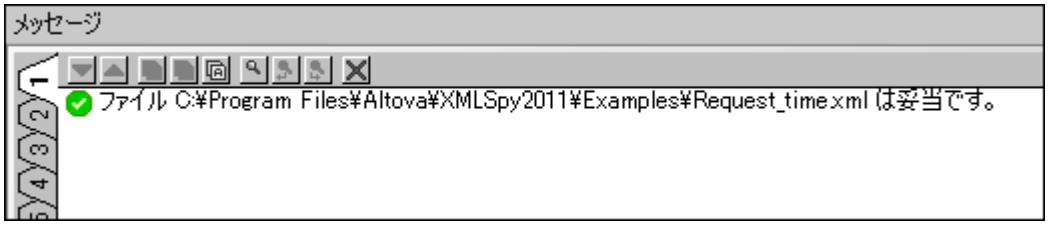
SOAP メッセージの検証を行うには、SOAP メッセージファイルを開き、「F8」キーを押下するか、メニューコマンドから「XML | 検証」を選択します(以下のスクリーンショットを参照)。SOAP メッセージファイルに W SDL ファイルがリンクされていない場合、SOAP メッセージのルールに従い、SOAP メッセージの検証が行われます。これらのルールに準拠していれば、ファイルは妥当ということになります(以下のスクリーンショットのメッセージウィンドウを御覧ください)。



SOAP ルールならびにリンクされている W SDL に対する検証

リンクされた W SDL も含めて SOAP メッセージの検証を行うには、SOAP ファイルに W SDL ファイルをリンクする必要があります。この操作は情報ウィンドウの SOAP タブにて行うことができます(以下のスクリーンショットを参照)。*検証を行う W SDL* アイテムの右にあるボタンをクリックして、「検証を行う W SDL を選択」を選択します。表示されるダイアログにて目的の W SDL ファイルを選択し、「OK」をクリックします。入力された W SDL ファイルは 情報ウィンドウに表示され、SOAP メッセージファイルがリンクされます。

「F8」を押下(または*メニューコマンド*から「XML | 検証」を選択)することで、SOAP *メッセージ*が、SOAP *メッセージのルール*だけでなく、*リンク*された W SDL *ファイルのルール*に対しても検証されます。



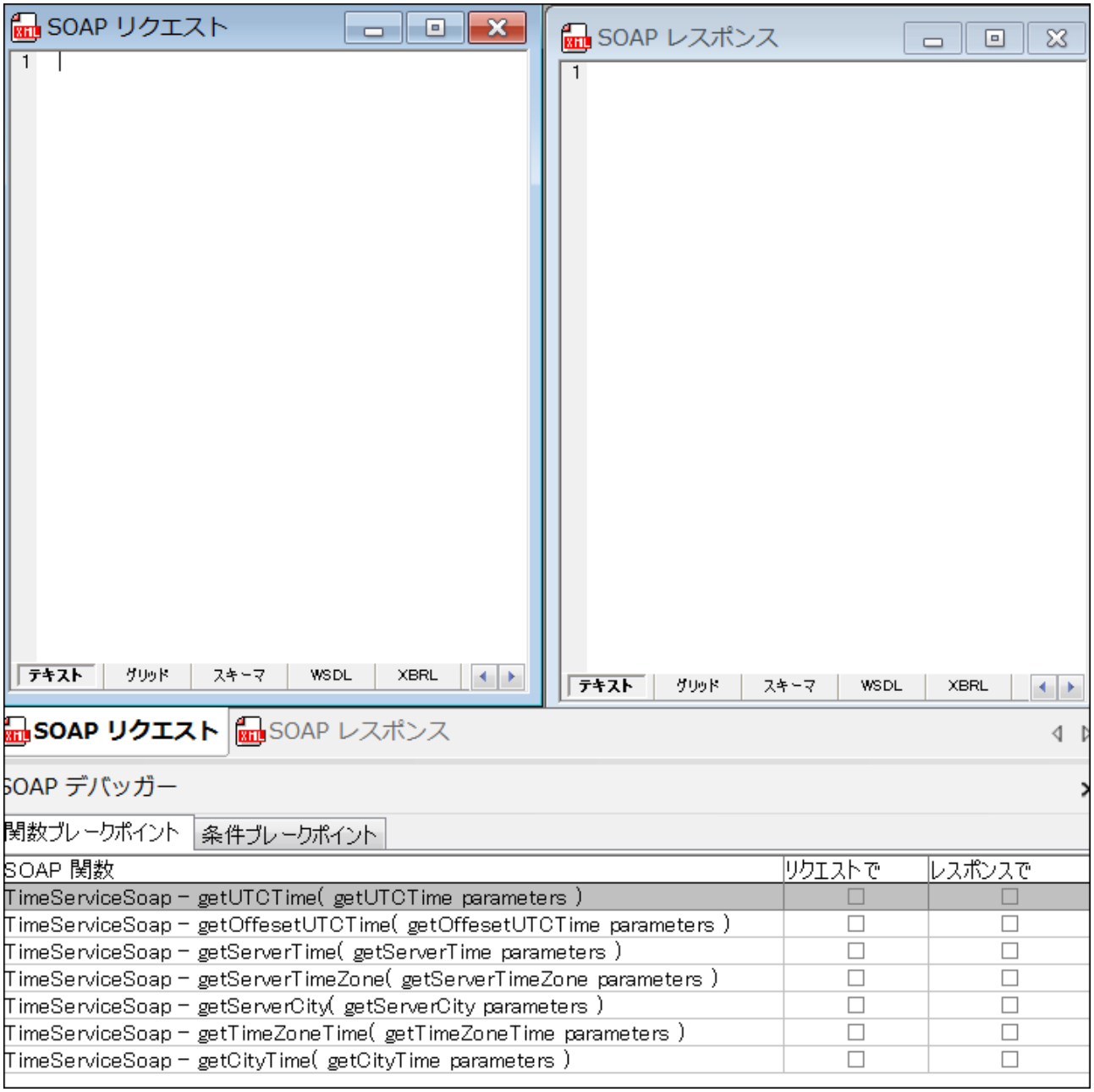
両方の*ルールセット*に*従った*ものであれば、*ファイル*は妥当なものとなります(*上のスクリーンショット*を参照)。

メモ: W SDL ファイルから XMLSpy のSOAP *リクエスト*作成機能(「**SOAP | SOAP リクエストの作成**」)を使用して、SOAP *リクエスト*が作成された場合、SOAP タブが情報ウィンドウに表示されます。(SOAP *リクエスト*が XMLSpy にて作成されなかったため)SOAP タブが情報ウィンドウに表示されていない場合、SOAP *リクエストファイル*を保存することで、SOAP タブが表示されます。

15.2.2 SOAP デバッガー

SOAP デバッガーにより、SOAP *リクエスト*と SOAP *レスポンス*を閲覧ならびに分析することができます(*以下のスクリーンショットを参照*)。デバッガーは*クライアント*と W eb サービス間の*プロキシ サーバー*として動作します。SOAP デバッガーでは以下の機能がサポートされます:

- SOAP *リクエスト*と*レスポンス*のステップ実行
- SOAP *リクエスト*と*レスポンス*の修正
- *クライアント*または*サーバー*への*リクエスト*転送
- *リクエスト*や*レスポンス メッセージ*における*ブレークポイント*のセットや、XPath 条件式を使った条件付き*ブレークポイント*



SOAP デバッガーは以下のように動作します:

- SOAP デバッガーセッションを開始する前にSOAP デバッガーオプション[698]をセットします。これらオプションにはコンピューターの IP アドレスやSOAP デバッガーのレイアウト/タイムアウト オプションなどが含まれます。
- SOAP デバッガーを開く(セッションを開始する)[699]ために、「SOAP | SOAP デバッグ セッション」を選択します。この時点で、(i) SOAP 情報が収められているW SDL ファイルの場所と、(ii) ソースならびにターゲット ポートの情報を入力する必要があります。
- SOAP デバッガーブレークポイント ウィンドウにて、必要なブレークポイントをセットします[704]。
- SOAP リクエストを作成するファイルを開き、SOAP デバッガーを実行[704]します。
- 結果を分析し[705]エラーがある場合にはそれらを修正します。

- SOAP デバッガーを閉じるには、「SOAP | SOAP デバッガー セッション」コマンドを選択してください。

このセクションのサブセクションで、SOAP デバッガーの使用方法が説明されています。

C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples フォルダーに収められているファイル DebuggerClient.htm は、サンプルファイルとして使用されています。このサンプルファイルでは、ブラウザーウィンドウは、SOAP メッセージを送信または受信するクライアントアプリとして作動します。Nanonull Time Service サービスは、http://www.nanonull.com/TimeService/TimeService.asmx?WSDL にある、サービスサーバーです。

15.2.2.1 SOAP コミュニケーション処理

プロキシサーバー(SOAP デバッガー)が開始されると、SOAP のやり取りは以下のように行われます:

プロキシサーバーにより外から入ってくるクライアントリクエストのソケット/ポートを常に監視

- クライアントアプリケーションがプロキシサーバーへリクエストを送信します。
- ブレークポイントがヒットした場合、クライアントリクエストを修正することができます。
- プロキシサーバーリクエストデータがウェブサービスサーバーへ転送されます。

ウェブサービスサーバーはプロキシサーバーに対して応答(レスポンス)を行ない、レスポンスデータをプロキシサーバーに返信

- ブレークポイントがヒットした場合、サーバレスポンスを修正することができます。
- プロキシサーバーレスポンスデータがクライアントアプリケーションへ転送されます。
- クライアントアプリケーションがプロキシサーバーからレスポンスデータを受信

ポート設定
SOAP デバッガーはクライアントのクリエストをモニターするために8080 ポートを使用します。このポート番号は、新たな SOAP デバッグセッションが開始されたときに変更することができます。このポートは、パーソナルファイアウォールや広告フィルターにより閉じられていることがあります。ポートが閉じられている場合、これらのプログラムを無効にするか、別のポートアドレスを割り振る必要があります。

15.2.2.2 SOAP デバッガーオプション

SOAP デバッガーオプションダイアログ(スクリーンショットを参照)によりコンピュータのアドレスと下にリストされる他のデバッガーオプションを指定することができます。「SOAP | SOAP デバッガー オプション」メニューコマンドを選択し、SPS デバッガー オプション ダイアログを開きます

- *コンピューターのアドレス* デバッガーが実行されるプロキシのサーバーのアドレス。プロキシサーバー上のデバッガーは、ネットワーク上のマシンのリクエストを受け、Web サービスに送信します。デバッガーは、XMLSpy 内で作動するため、XMLSpy がイン

ストールされているマシンがプロキシサーバーとして作動します。マシンのIP アドレスは、自動的に検出され、このフィールドに入力されます。IP アドレスが自動的に検出されない場合、このフィールドにIP アドレスを（ `http` アドレスとして）入力します。コンピューターのIP アドレスを検出するには、コマンドプロンプトウィンドウを開き、コマンド `ipconfig /all` を入力して、「Enter」を押します。

- *タイムアウト*: タイムアウトの値は、SOAP デバッガーがブレークポイントにて待機する時間として使用されます。デフォルトのタイムアウト値は5 秒です。
- *入力ヘルパーを閉じる*; *プロジェクト/情報ウィンドウを閉じる*: このオプションによりインターフェイスのSOAP デバッガーウィンドウにより多くの表示領域を割り当てることができます。

15.2.2.3 デバッグ セッションの開始

SOAP デバッガーセッションはXMLSpy のアクティブなファイルでいつでも開始することができます。セッションは、SOAP リクエスト エントリーポイント ファイル[702]を開く前、または開いた後に開始することができます。SOAP デバッガーセッションを開始すると以下を入力するように求められます:

1. SOAP 情報を取得するために使用されるW SDL ファイルの場所
2. 接続の設定

これらステップの詳細については以下を参照ください。

W SDL ファイル ロケーション

デバッガーセッションが開始すると、SOAP情報を取得するためのW SDL ファイルのURL を入力するように求められます。 サンプルファイル[702] `DebuggerClient.html` は、以下のW SDL ファイルurl を使用します:

```
http://www.nanonull.com/TimeService/TimeService.asmx?WSDL
```

メニューコマンドの「SOAP | SOAP デバッグ セッション」を選択することで、SOAP デバッガーを開始します。W SDL ファイルの場所を入力するダイアログ ボックスが表示されます（以下のスクリーンショットを参照）。

W SDL ファイルのURL を入力し、「OK」をクリックします。ソースならびにターゲットダイアログが表示されます（以下のスクリーンショットを参照）。

接続の設定

ソースとターゲットダイアログの接続の設定ダイアログ（下のスクリーンショット）で以下のリストの設定を行うことができます。

- **ソース ポート**: やり取りに使用されるコンピューター上にあるポート番号を指定します。デフォルトでは8080 となっています。この設定はSOAP デバッガーが開始されるごとに変更することができます。
- **ターゲット ポートとアドレス**: これらの設定は直前のステップにて選択されたW SDL ファイルから取得され、ダイアログへ自動的に入力されます。ターゲット ポートが443 の場合、「https による安全な接続」チェック ボックスがデフォルトでチェックされます。

80 以外のポート番号が入力された場合、手動でチェック ボックスにチェックを入れる必要があります。「HTTPS 設定」ボタンをクリックすることで、セキュアな接続のオプションを選択することができます（オプションの使用方法に関する詳細については、「SOAP リクエスト パラメーターの変更」[1383] のセクションを参照ください）。

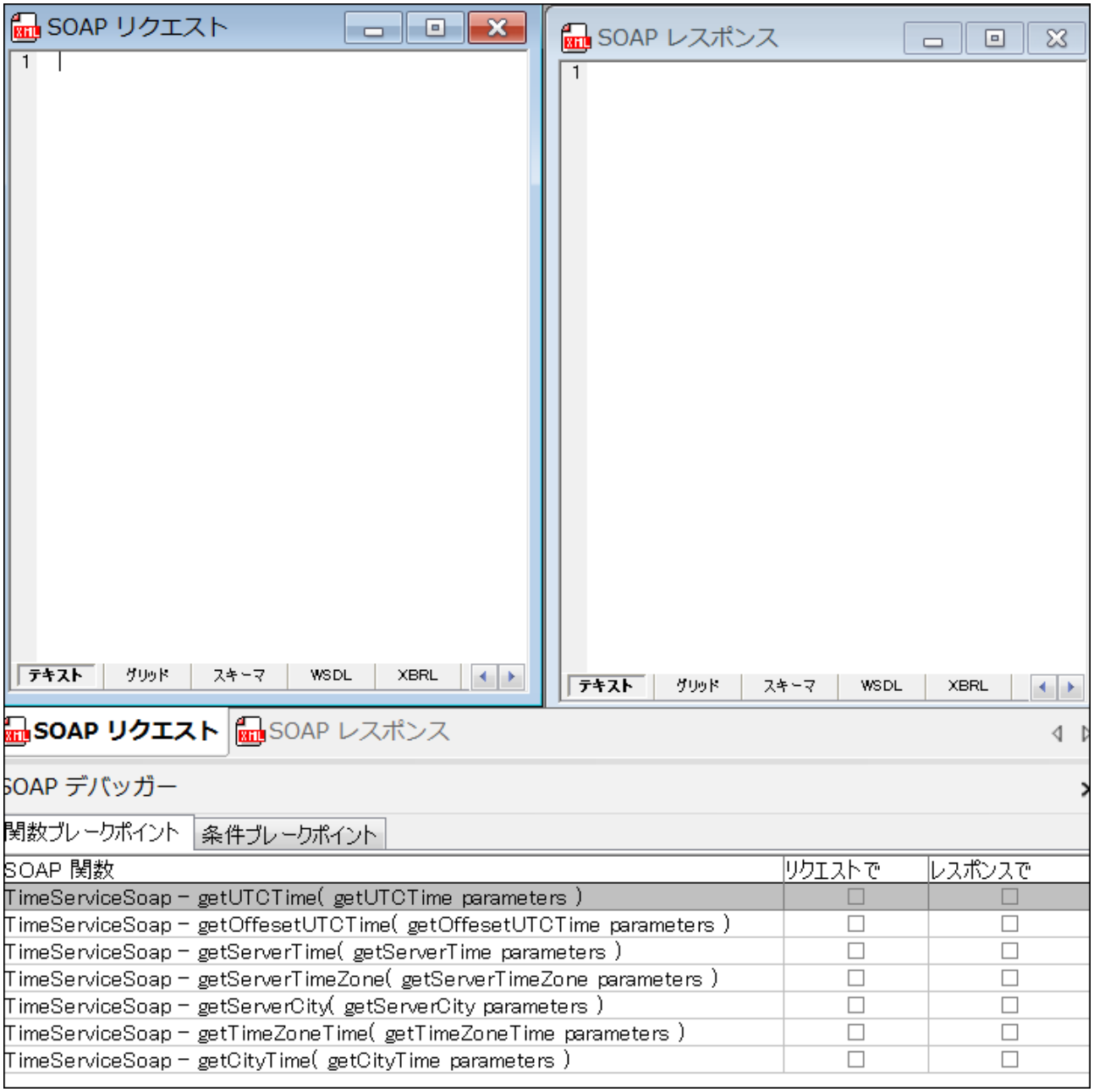
これらの設定を確認し、必要に応じて編集を行った後には、「OK」をクリックすることで SOAP デバッガー セッションが開始されます。SOAP デバッガーが開始された段階では、プロキシ サーバーはアクティブとなっていません（SOAP デバッガー ツール バーにあるプロキシ サーバー アイコン がグレーアウトされます）。SOAP ツール バーにある実行アイコンをクリックするか、メニュー コマンドの「SOAP | 実行」を選択してください。実際のデバッグについては、「デバッグ」[704] のセクションを参照ください。

SOAP デバッガー レイアウト

SOAP デバッガー ウィンドウには 3 つのウィンドウが含まれます（以下のスクリーンショットを参照）:

- SOAP リクエスト ウィンドウ、
- SOAP レスポンス ウィンドウ、
- SOAP デバッガー、ブレークポイント設定ウィンドウ

デフォルトでは XMLSpy インターフェイスの上部にリクエストとレスポンスのウィンドウが開かれ、ブレークポイント設定ウィンドウが下部に表示されます。デフォルトのレイアウトを以下のスクリーンショットに示します。



XMLSpy サイドバーウィンドウ(プロジェクト、Info、入力ヘルパーウィンドウ)を隠すことで、SOAP デバッガーウィンドウへ更なる表示領域を割り当てることができます。「SOAP | SOAP デバッガーオプション」メニューコマンドからアクセスすることのできる SOAP デバッガーオプションダイアログ 698 では、これらウィンドウの表示を指定することができます。

信頼された証明書について

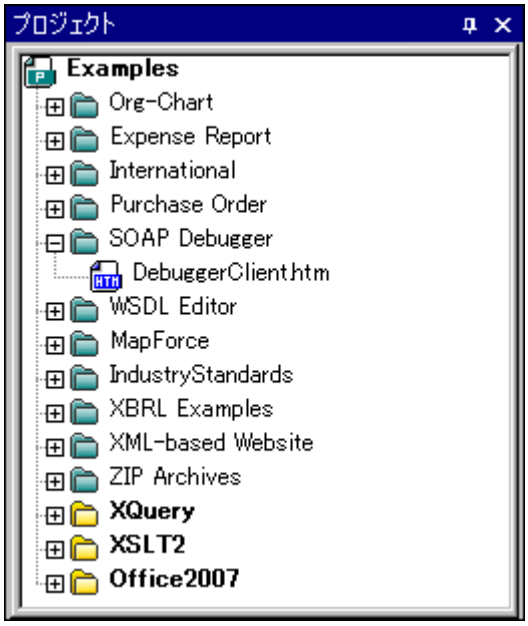
Altova 製品はセキュアな Web サーバーの証明書へアクセス、管理するために、Internet Explorer (IE) を使用します。Web サーバーの証明書を IE へインストールすることにより、警告が表示されたり処理を中断することなく、その Web サーバーへアクセスすることができるようになります。以下の手順を踏んでください:

- Internet Explorer 9 以降で、安全な（セキュアな）Web サイトを開きます。
- 「ファイル | プロパティ」を選択し、「証明書」ボタンをクリックします。
- 「証明書のインストール」をクリックして証明書インポート ウィザードを表示します（このウィザードへは「ツール | インターネット オプション | コンテンツ | 証明書 | インポート」からもアクセスすることができます）。
- 証明書が信頼されたルート証明書ストアに配置され、手動で参照することができるようになります。
- ウィザードを完了し、「OK」をクリックして証明書とプロパティ ダイアログを閉じます。Internet Explorer を再起動する必要があることもあります。

15.2.2.4 SOAP リクエスト エントリー ポイント

サンプル プロジェクトにある HTML ファイル（`DebuggerClient.htm`）には SOAP デバッガーのデモを行うためのスクリプトが含まれています。このファイルにより SOAP リクエストを Web サービスへ送信し、Web サービスからのレスポンスを表示することができます。以下の操作により XMLSpy でこのファイルを開くことができます:

1. メニュー コマンドから「プロジェクト | プロジェクトを開く」を選択します。
2. C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples フォルダーを参照し、`Examples.spp` ファイルを選択します。プロジェクト ウィンドウにサンプル プロジェクトがロードされます（以下のスクリーンショットを参照）。

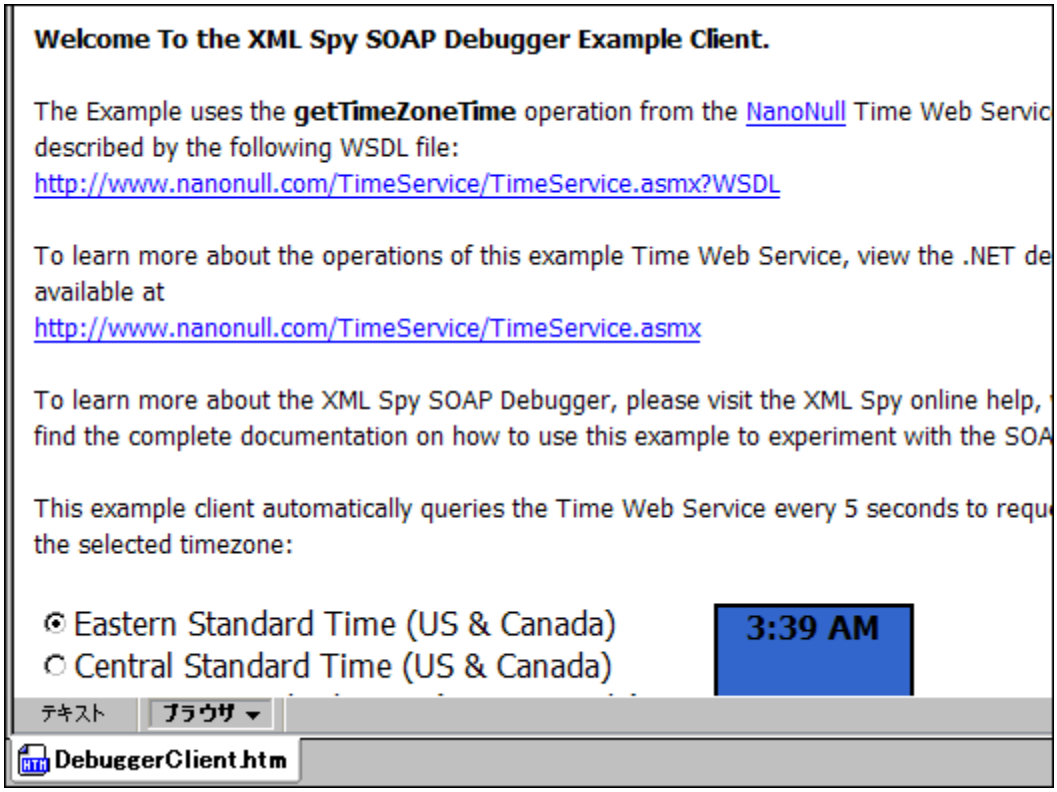


3. SOAP Debugger の＋サインをクリックすることで、そのコンテンツを確認することができます。`DebuggerClient.htm` をダブルクリックすることで、そのファイルが XMLSpy で開かれます。

メモ メニューの「ファイル | ファイルを開く」からもこのファイル（`DebuggerClient.htm`）を開くことができます。`C:\Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples` フォルダーに配置されます。

サンプル ファイル

`DebuggerClient.htm` は XMLSpy のブラウザービューにて以下のように表示されます。ラジオ ボタンが選択されると、SOAP リクエストが Nanonull Time Web Service に送信されます。Web サービスからのレスポンスがラジオ ボタンの右にある色付きのボックスに表示されます。



Eastern Standard Time (東部標準時) リクエストへのレスポンスが、上のスクリーンショットにある青のボックスに表示されます (6:23)。GMT ラジオボタンをクリックしてください。GMT の値が表示される代わりに、エラーメッセージが表示され、ボックスが赤色で表示されます (以下のスクリーンショットを参照)。

SOAP デバッガーを使用することで、SOAP メッセージを分析し、エラーの識別を行うことができます。以下にある 3 つのセクションでは、(i) ブレークポイントのセット 704、(ii) `DebuggerClient.htm` とともに SOAP デバッガーの実行 704、そして (iii) SOAP デバッガー出力の分析とエラーの識別 705 を行う方法について記されます。

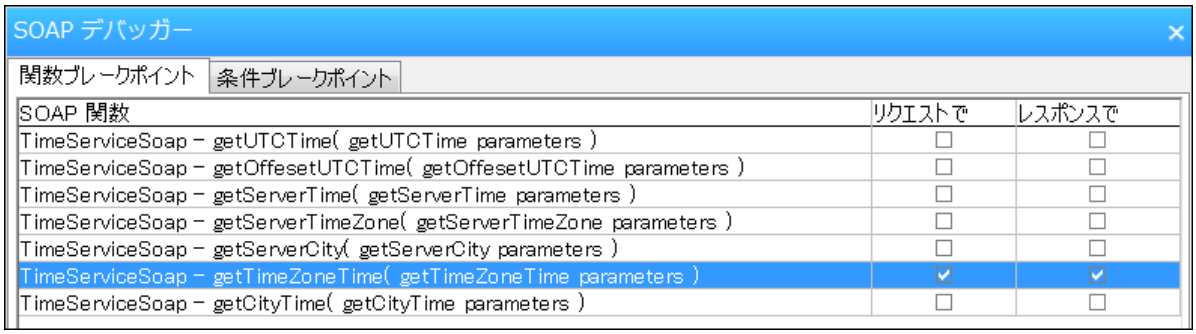
15.2.2.5 ブレークポイントのセット

デバッグを行う前に、SOAP デバッガーにてブレークポイントをセットする必要があります。デバッグを開始すると、ブレークポイントへ処理が移った際にSOAP デバッガーによりリクエストとレスポンスが表示されます。

SOAP デバッガーでは、SOAP デバッガーが開始された時に[699]選択されたW SDL ファイルから取得された情報に従って、(関数ブレークポイントと条件ブレークポイントに) ブレークポイントが表示されます。これらのブレークポイントは、W SDL ファイルから生成することのできる SOAP リクエストに関連付けられます。各 SOAP リクエストに対して対応する列のチェック ボックスにチェックを入れることで、リクエストまたはレスポンスにブレークポイントをセットすることができます(以下のスクリーンショットを参照)。

この例では、以下のファイルを使用します:

- SOAP リクエスト エントリーポイント[702]としての DebuggerClient.htm
- SOAP デバッガーを開始[699]した際に選択されたW SDL ファイルの http://www.nanonull.com/TimeService/TimeService.asmx?WSDL



DebuggerClient.htm からリクエストされたW eb サービスでは、選択されたタイムゾーンの時間を取得するために getTimeZoneTime というメソッドが使用されます。SOAP デバッガーでは、選択されたW SDL ファイルから生成することのできる SOAP リクエストがブレークポイントとして表示されます。ここでは getTimeZoneTime メソッドの「リクエストで」と「レスポンスで」の両方にブレークポイントをセットします(上のスクリーンショットを参照)。これにより、エラーを含むW eb サービスへのリクエストやW eb サービスからのレスポンスを分析することができます。

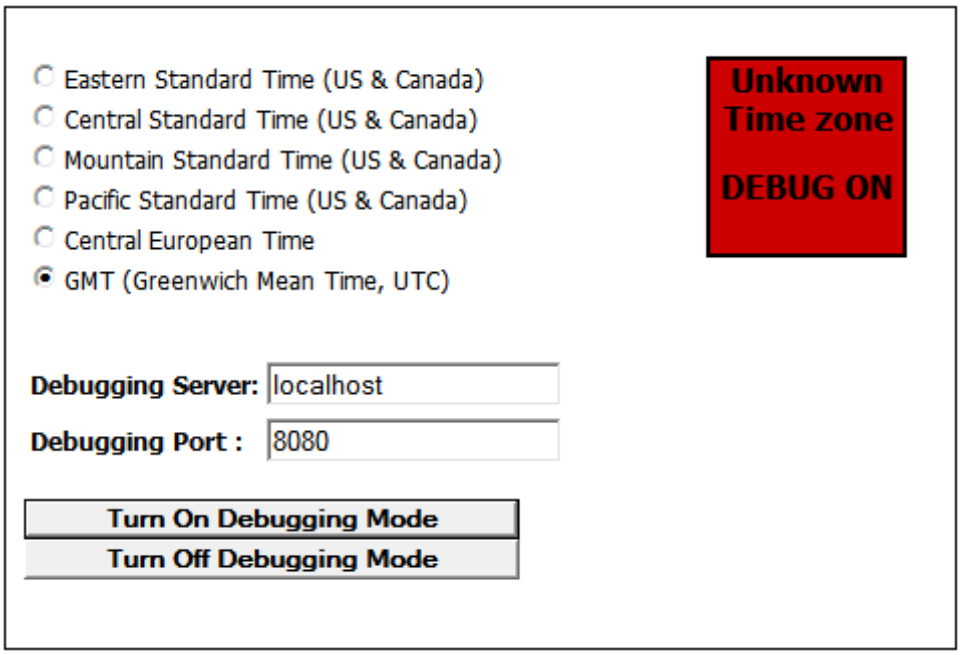
ブレークポイントに関する詳細な情報については、「ブレークポイントに関する更なる情報」[707] のセクションを参照ください。

15.2.2.6 デバッグ

この例では、以下のファイルを使用します:

- SOAP リクエスト エントリーポイント[702]としての DebuggerClient.htm
- SOAP デバッガーを開始[699]した際に選択されたW SDL ファイルの http://www.nanonull.com/TimeService/TimeService.asmx?WSDL

ブレークポイントをセット[704]した後に「実行」アイコン をクリック(またはメニューの「SOAP | 実行」コマンドを選択) します。DebuggerClient.htm のタブをクリックし、SOAP エントリーポイント ファイルを表示してください。GMT オプションが選択されていることを確認し、「**Turn on Debugging Mode**」をクリックしてください(以下のスクリーンショットを参照)。



これにより「Debug On」メッセージが表示され、SOAP リクエストがSOAP デバッガーへ送信されます。デバッガーの結果がSOAP リクエストならびにSOAP レスポンス ウィンドウに表示されます。詳細については次のセクションの「結果とエラーの分析」705 を参照してください。

SOAP デバッガー コントロール

SOAP デバッガー ツール バーにはSOAP デバッガーを操作するためのアイコンが含まれています（以下のスクリーンショットを参照）。

これらのアイコンは、左から以下の様な意味を持ちます:

- **実行**: デバッグを開始します。
- **ステップ実行**: リクエスト-レスポンスの処理をステップ実行し、ブレークポイントで停止します。
- **次のリクエストでブレーク**: 次のSOAP リクエストで停止します。
- **次のレスポンスでブレーク**: W eb サービスから得られた次のレスポンスで停止します。
- **プロキシ サーバーを停止**: デバッグを停止します。この操作はSOAP デバッグ セッションを終了するのとは異なるということに注意してください。SOAP デバッグ セッションを終了/開始するには、メニューから「SOAP | SOAP デバッグ セッション」コマンドを選択してください。

15.2.2.7 結果とエラーの分析

SOAP デバッガーの結果はSOAP リクエストとSOAP レスポンスのウィンドウに表示されます。ブレークポイントは、デフォルトでSOAP デバッガー ウィンドウの下部に配置されているSOAP デバッガー ブレークポイント ペインにてセットされます。

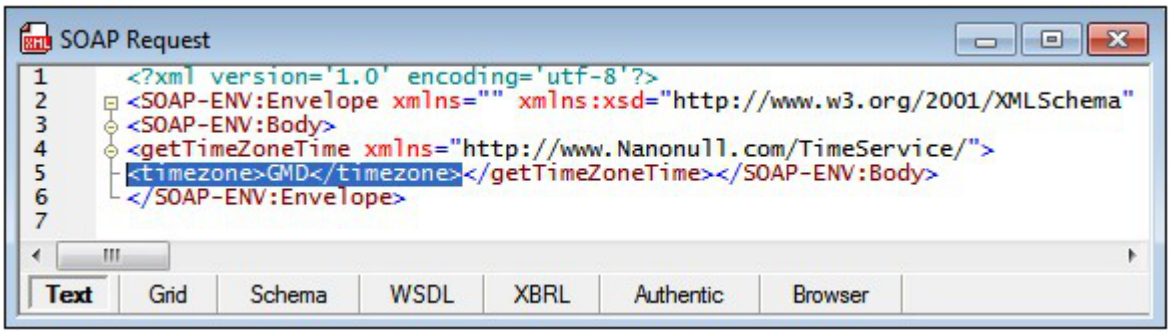
セットされたブレークポイントに従い、対応するウィンドウ（SOAP リクエストまたはSOAP レスポンス）に結果が表示されます。

この例では、以下のファイルを使用します:

- SOAP リクエスト エントリーポイント[702]としてのDebuggerClient.htm
- SOAP デバッガーを開始[699]した際に選択されたW SDL ファイルの http://www.nanonull.com/TimeService/TimeService.asmx?WSDL

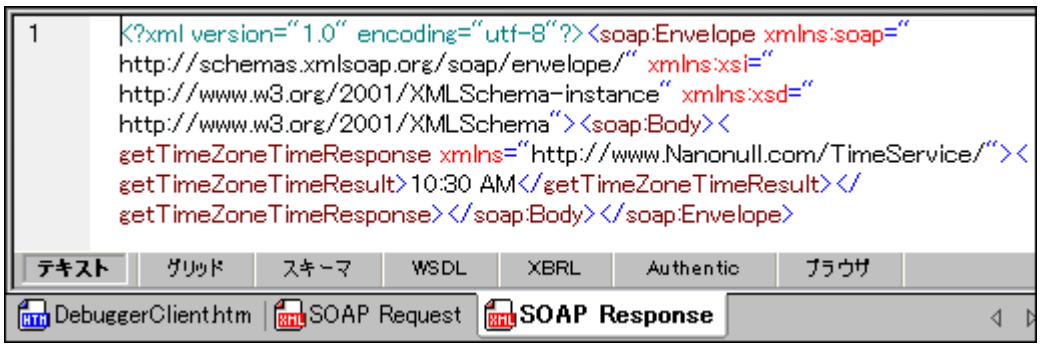
エラーの検知とフィックスのテスト

直前のセクション「デバッグ」[704]に記述された方法によりデバッグは開始されます。GMT を選択した際のSOAP リクエストがデバッガーのSOAP リクエスト ウィンドウのテキスト ビューに表示され、編集を行うことができます。



timezone 要素に注目すると、値がGMD となっています。これは間違った値であるため、GMT に変更します。timezone 要素をダブルクリックして、要素のコンテンツをGMT に変更してください。

問題が解決されたか確認するため、SOAP デバッガーツールバーにある実行アイコンをクリック（またはメニューの「SOAP | 実行」コマンドを選択）して、正しいリクエストをW eb サービスへ送信します。数秒後、SOAP リクエストに対するW eb サービスからのレスポンスがSOAP レスポンス ウィンドウに表示されます。「ビュー | ワード ラップ」を選択することで、SOAP レスポンス全体がウィンドウに表示されます（以下のスクリーンショットを参照）。



DebuggerClient.htm タブをクリックして、SOAP デバッガーツールバーにある「実行」アイコンをクリックしてください。エラーメッセージが消え、正しいGMT の時間が表示されます（以下のスクリーンショットを参照）。

メニューの「SOAP | SOAP デバッガー セッション」コマンドを選択することで、SOAP デバッガー セッションが終了します。

エラーの修正

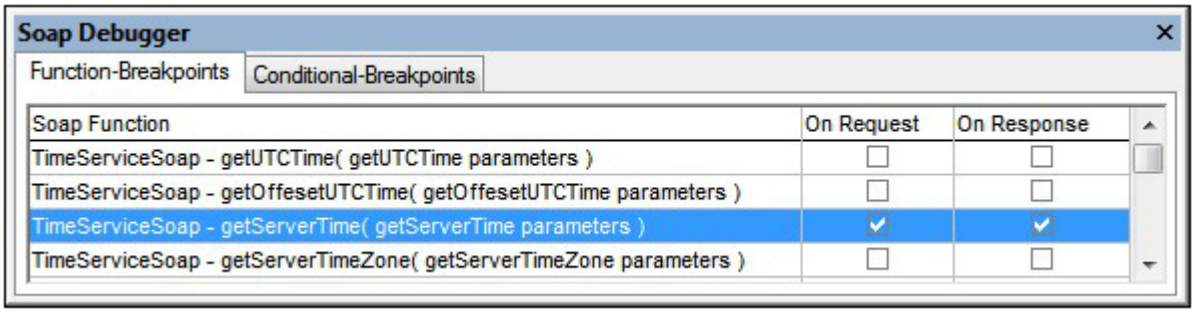
これで、(GMT ではなく) GMD という無効な値がSOAP リクエストにて生成されているということが分かりました。SOAP リクエスト エントリー ポイント ファイルを開き、(「Ctrl」+ F または「編集 | 検索」により) GMD という単語を検索すると、以下のスクリーンショットに示されているように、該当するコードを確認することができます。

```
function changeZones(){
    if (timezone[0].checked)
        msCurrentTimeZone='EST';
    else if (timezone[1].checked)
        msCurrentTimeZone='CST';
    else if(timezone[2].checked)
        msCurrentTimeZone='MST';
    else if(timezone[3].checked)
        msCurrentTimeZone='PST';
    else if(timezone[4].checked)
        msCurrentTimeZone='CET';
    else if(timezone[5].checked)
        msCurrentTimeZone='GMD';
}
```

このエラーが修正された状態で GMT ラジオ ボタンが選択されると、未知のTimezone エラーの代わりに正しい GMT の時間が表示されます。

15.2.2.8 ブレークポイントに関する更なる情報

SOAP デバッガー ウィンドウではブレークポイントをセットならびに削除することができます。ウィンドウには 2 つのタブから構成されます (以下のスクリーンショットを参照)。



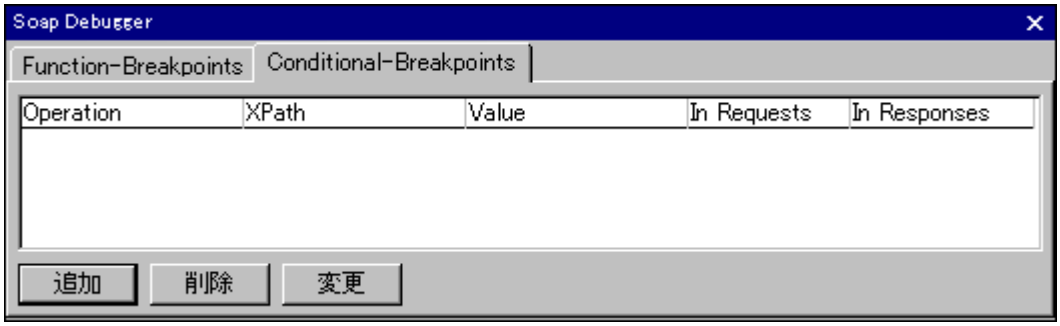
関数ブレークポイント タブ

関数ブレークポイント タブではSOAP リクエストやレスポンスのメソッドにブレークポイントをセットすることができます。デバッガーによりブレークポイントとしてセットされたメソッドの呼び出し行がハイライトされます。クライアントが通信を行うデータパケットが分析され、WSDL から得られた関数にマッチされます。あるメソッドへブレークポイントがセットされている場合、そこでSOAP デバッガーが停止します。ツールバーのボタンはこの時点で選択可能な状態になっています。

データがSOAP リクエストまたはSOAP レスポンスドキュメント ウィンドウに表示されます。この時点でSOAP ウィンドウに表示されたSOAP ドキュメントを修正することができます。ツールバーにある「**サーバーを停止**」以外のアイコンをクリックすることで、データが送信されます。

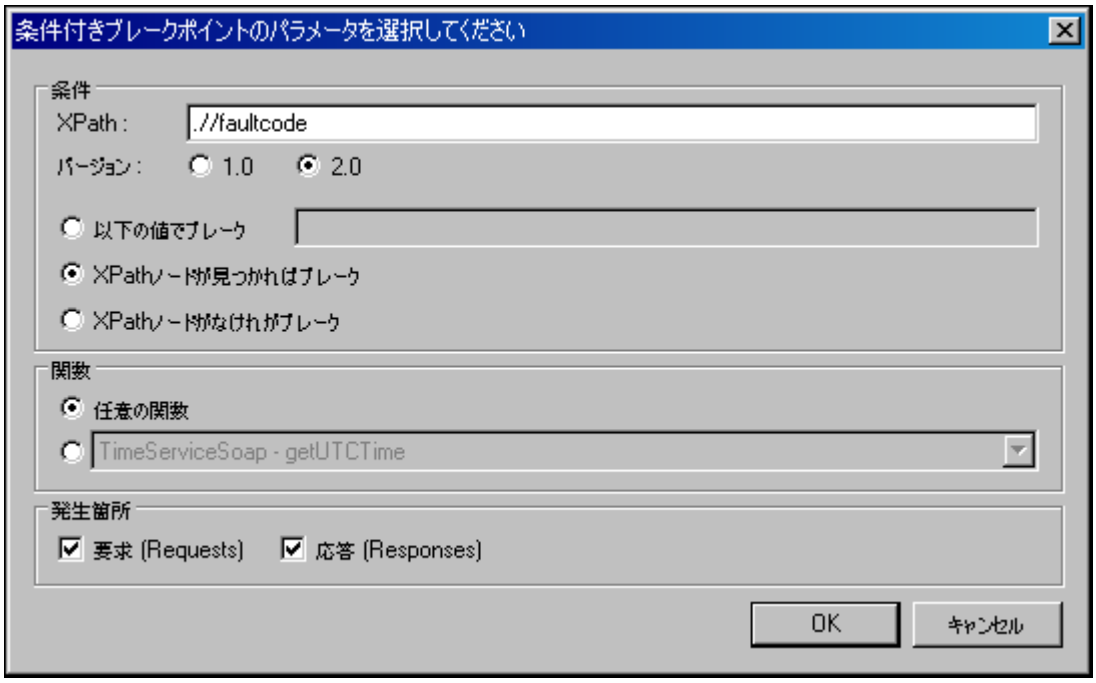
条件ブレークポイント タブ

条件ブレークポイント タブではXPath 条件式によりブレークポイントを定義することができます（以下のスクリーンショットを参照）。SOAP リクエストによりエラーが生じる場合、SOAP レスポンスには `faultocde` 要素が含まれます。そこで、`faultcode` 要素が現れるとデバッガーが停止するようなブレークポイントを作成することにします。



以下の操作により条件ブレークポイントをセットすることができます:

1. 条件ブレークポイント タブをクリックして、「**追加**」ボタンをクリックします。以下に示されるダイアログが表示されます。

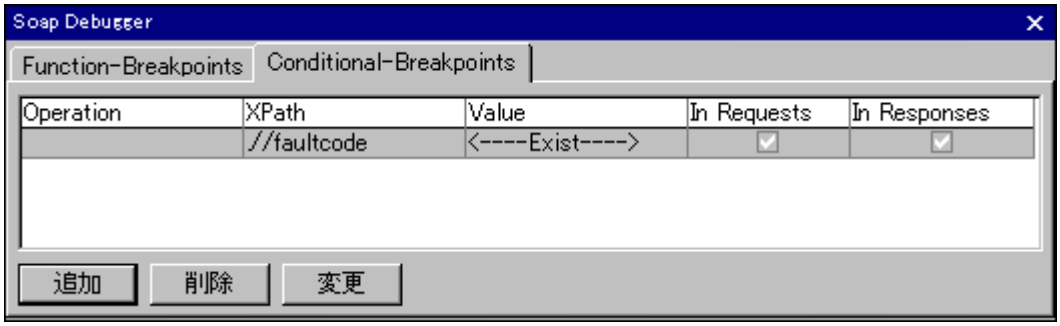


2. XPath フィールドにXPath 条件式 (例えば .//faultcode) を入力します。
3. XPath のバージョンを (1.0 または2.0) 選択し、「XPath ノードが見つかればブレーク」ラジオボタンを選択します。
4. 「OK」をクリックして設定を完了します。SOAP リクエストまたはレスポンスに .//faultcode 要素が出現したところでSOAP デバッガーが停止されるようになります。

このダイアログで使用可能なオプションについて以下に記します:

- **XPath 条件式フィールド**: XPath 条件式を入力してください。XPath を入力することで、ラジオボタンのオプションが利用可能になります。
- **バージョン**: XPath 条件式で使用されるXPath のバージョンを選択します。
- **ブレーク ラジオ ボタン**: 選択されたオプションが発生した時にデバッガーが停止します。利用できるオプションには、(i) ターゲットのXPath ノードがフィールドで入力された値にマッチした時にブレーク、(ii) 指定されたXPath ノードがSOAP リクエストまたはレスポンス内に存在する時にブレーク、または (iii) 指定されたXPath ノードがSOAP リクエストまたはレスポンスに存在しない場合ブレークがあります。
- **リクエストとレスポンス**: ダイアログのオプションがSOAP リクエストまたはレスポンス (またはその両方) に対して適用されるか指定します。
- **関数**: 定義した条件ルールがすべてのメソッド/関数に対して適用されるのか (「任意の関数」ラジオボタン)、または入力したメソッド/関数に対してだけ適用するのかを選択できます。

上に示されるダイアログにて条件が定義されると、その条件ブレークポイントが条件ブレークポイント タブに表示されます。



タブに表示される列について以下で説明します:

- **オペレーション**: 検索されるメソッド/関数が含まれます。「**任意の関数**」ラジオボタンが選択された場合、空のフィールドが表示されます。特定のメソッド/関数が選択された場合、そのメソッド/関数が表示されます。
- **XPath**: 定義したXPath 条件式が含まれます。
- **値**: ノードをチェックするためのXPath 値が含まれます。「**以下の値でブレーク**」が選択された場合、指定された文字列が表示されます。「**XPath ノードが見つかればブレーク**」の場合は`<!--Exist-->`が、「**XPath ノードがなければブレーク**」が選択されている場合には`<!--Missing-->`が表示されます。
- **リクエスト内とレスポンス内**: 条件がチェックされる場所がチェックボックスにより示されます。列にあるチェックボックスを直接クリックすることでも、設定を変更することができます。

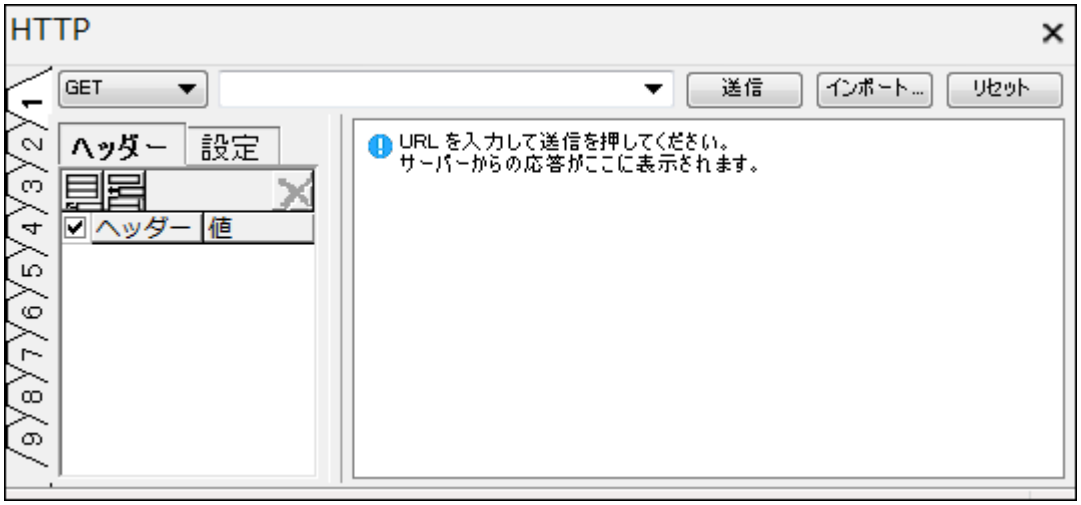
条件ブレークポイントを編集するには、タブに表示された行をダブルクリックするか、「**変更**」ボタンをクリックします。条件ブレークポイントを削除するには、その行が選択されている状態で、「**削除**」ボタンをクリックしてください。

16 HTTP

HTTP（ハイパーテキスト・トランスファー プロトコル、略称 HTTP）はファイル（テキスト、イメージ、オーディオ、ビデオ、および他のマルチメディアファイル）のインターネット上での転送を定義する、プロトコール（または、ルールのセット）です。各 W eb サーバーはHTTP リクエストを継続的に待ち、受信後に処理をする（デーモンとして知られる）プログラムを実行します。例えば、W eb ホームページを訪問すると、ブラウザーはW eb サイトのW eb サーバーのホームページをリクエストするHTTP コマンドを送信します。サーバーのHTTP デーモンはリクエストを受信し、リクエストされたページを送信します。HTTP の重要なプロパティの1つは使用されません。すなわち、前の、または、後のコマンドを参照することなくHTTP コマンドが個別に実行されることを意味し、状態が無いことを表します。
XMLSpy 内では、HTTP 出力ウィンドウ内でHTTP コマンドをテストすることができます（*下のスクリーンショット*）。W eb サーバーにHTTP リクエストを作成、送信、受信し、更にレスポンスをチェックすることができます。

HTTP 出力ウィンドウのパーツ

HTTP 出力ウィンドウは9つのタブが存在します（*下のスクリーンショット参照*）。各タブ内に個別のリクエストを保管することができ、タブ間を切り替えることが出来ます。ウィンドウ内でリクエストを作成した後、「送信」ボタンをクリックしてリクエストを送信することができます。レスポンスは直接ウィンドウ内に表示されます。



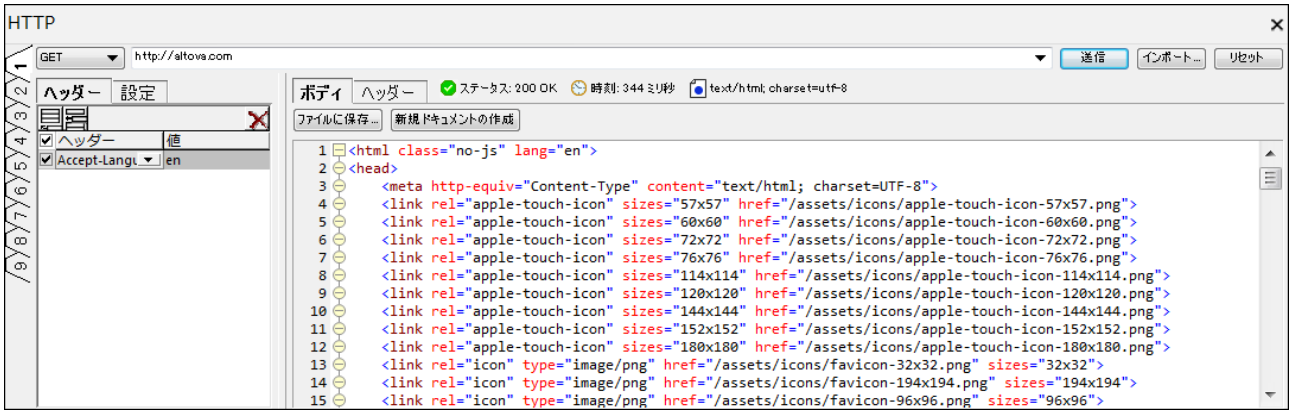
ウィンドウは以下のパートにより構成されています:

- 上の部分: (i) 使用するHTTP メソッドを選択するコンボボックス。(ii) W eb サーバーのURL のための入力フィールド。(iii) HTTP リクエストの実行に関連したボタン（「送信」、「インポート」、および「リセット」）。
- リクエストの作成[712] のための左側のペイン
- レスポンスの表示[718] のための右側のペイン

HTTP 出力ウィンドウの作動方法に関しては、このセクションのサブセクションで説明されています。

16.1 リクエストの送信

HTTP 出力ウィンドウ内でHTTP リクエストを送信することができます（*下のウィンドウ*）。**ウィンドウの左側のペイン** 内でリクエストは定義されます。ウィンドウの3つのタブそれぞれに異なるリクエストを設定することができ、それぞれは以下の部分から構成されています: (i)（ダイアログの上の部分で定義されている）リクエストとターゲット URL の HTTP メソッド[712] 。(ii)（ヘッダータブ内の）リクエストのHTTP ヘッダー。(iii)（設定タブ内の）接続設定。(iv) 接続設定。および(iv)（下のスクリーンショットに表示されていない ボディタブ）POST とPUT メソッド[712] の場合の HTTP メッセージボディ。（右上にある）「リセット」をクリックしてタブを空の状態に戻すことができます。



HTTP リクエストを送信するには、次をおこないます:

1. 左上のコンボボックスからHTTP メソッド（GET、POST、PUT、DELETE、HEAD、または OPTIONS）を選択します（*スクリーンショットを参照*）。
2. ターゲット W eb ページのURL を入力します（例、`http://www.altova.com`。`altova.com` のみを入力することもできます。URL の`http://` の部分は自動的に補完されます）。
3. ヘッダータブ内で、HTTP ヘッダー値を指定することができます（*上のスクリーンショットを参照*）。ヘッダーを選択、または、入力することができ、値を入力することができます（HTTP 1.1 ヘッダーのリストに関しては、ここを参照してください）。ツールバーのタブ内の「**挿入**」、「**追加**」、「**削除**」アイコンを使用して、ヘッダーを追加または削除してください。ヘッダーを削除する代わりに、ヘッダー名の左側の「有効化」チェックボックスをのチェックを解除して、ヘッダーを無効化することができます。これによりヘッダーを後で使用する場合、ヘッダーを再度入力する手間を省くことが出来ます。ヘッダーを受け入れる[719] も参照してください（送信時に自動的に追加されるヘッダーのために値を設定すると、自動的に追加する値の代わりにこの値を使用することができます）。
4. `POST` または `PUT` リクエストを送信する場合、ヘッダーと設定タブに加え、*ボディ* タブを使用できるようになります。ボディの作成方法に関しては、*POST とPUT リクエストのボディ*[713] のセクションで詳しく説明されています。
5. 設定タブ内でタイムアウトとセキュリティ設定を指定することができます。このタブに関する詳細は、下の*HTTP リクエストの設定*[714] のセクションを参照してください。
6. リクエストを送信するために、（ウィンドウの右上にある）「**送信**」をクリックします。
7. タブを空の状態に戻すには、「**リセット**」をクリックします。以下が発生します: (i) コンボボックスのドロップダウンリスト内の最初のメソッドを使用する（`GET`）。(ii) 現在のURL エントリが削除されます。(iii) 全てのヘッダー、設定、および、ボディの定義が削除されます。

メモ ウィンドウのインポート[716] ボタンを使用してW ADL ファイルからHTTP 出力ウィンドウにリクエストをインポートすることができます。

メモ UTF-8 エンコードでリクエストは送信されます。他のエンコードはUTF-8 に変換され、UTF-8 データが送信されます。

HTTP メソッド

次のHTTP メソッドがサポートされます

GET
GET メソッドは、指定されたURL にあるリソースをリクエストします。URL にクエリを追加することもできます。例: http://www.altova.com?name1=value1&name2=value2。ヘッダーとボディを含むメッセージ内でリソースは返されます。

HEAD
HEAD メソッドとGET リクエストは類似していますが、メッセージボディは返されず、指定されたURL にあるリソースについて欲めた情報を含むメッセージヘッダーのみが返されます。

POST
指定されたURL にある既存のリソースを更新するために、または、指定されたURL で新規のリソースを作成するためにPOST メソッドが使用されます。リソースに提出するデータはHTTP リクエストのボディ内に置かれます。詳細に関しては、次を参照してください: *POST とPUT リクエストのボディ*[713]。

PUT
PUT メソッド は、指定されたURL で新しいリソースを作成するために使用されます。リソースに提出するデータは、HTTP リクエストのボディ内に置かれます。詳細に関しては、次を参照してください: *POST とPUT リクエストのボディ*[713] 。

DELETE
DELETE メソッドは、指定されたURL にあるリソースを削除します。

OPTIONS
OPTIONS メソッド はサーバーがサポートするHTTP メソッドのリストを返します。

POST とPUT リクエストのボディ

POST とPUT リクエストのために、POST またはPUT リクエストのボディを指定することのできる追加 *ボディ*タブを使用できるようになります (*下のウィンドウを参照*)。*ボディ*タブには2 つのモードがあります: **エディターモード** と**ファイルモード**。ボディタブの左上のツールバーボタンをしようして、これら2 つのモード間を切り替えることが出来ます(*スクリーンショットを参照*)。エディターモード内では、HTTP リクエストをペイン内で直接編集することができ、ファイルモード内ではHTTP リクエストのボディを含むファイルを選択することができます(*のスクリーンショット内で選択されている状態が示されています*)。

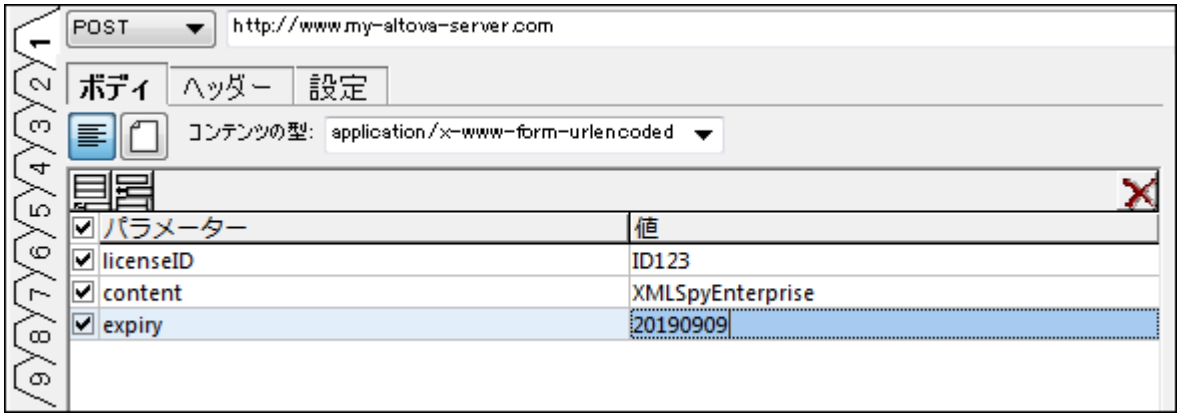
コンテンツ型フィールドによりリクエストのコンテンツの型 ヘッダーを指定することができます。このフィールドのコンボボックスオプションは各モード(エディターとファイル)により異なります。使用できるコンボボックスオプションから選択、または、MIME 型を入力します。このフィールドで指定されている値は、ヘッダー、または、ボディタブ内で指定されているコンテンツの型 ヘッダーを上書きすることに注意してください。

エディターモード
エディターのスタイルは選択されたコンテンツの型により異なります:

- **text/plain**、**text/xml**、**application/xml**、と**application/json** コンテンツ型: XML とJSON ドキュメントのための構文色分けや、行番号などのインテリジェントな編集機能を搭載したテキストエディター。上のスクリーンショットは

text/xml コンテンツ型のためのエディターを表示しています。リクエストのためのボディはエディター内に入力します。コンテンツ型フィールドにリクエストのコンテンツの型が指定されており、リクエスト以外の場所によるエントリにより上書きされることはありません。

- **application/x-www-form-urlencoded コンテンツ型**: 各ラインがリクエストのボディの名前値ペアを表示するグリッドビューを持つエディターです（*下のスクリーンショット*）。



ファイルモードを切り替えるには、ボディタブの左上にある「ファイル」アイコンを選択します（*上のスクリーンショットを参照*）。自動的にボディのコンテンツ型を有効化するには、コンテンツ型フィールド内で「自動」を選択します。コンテンツ型の自動的な検知はファイルの拡張子により決定されます。コンテンツ型を入力すると、入力されたコンテンツ型を使用してリクエストが送信されます。この場合、コンテンツ型が正確であることを確認してください。

リクエストのヘッダーと設定を（ヘッダータブと設定タブなどの）他のリクエストと同様簡素化することができます。

HTTP リクエストのための設定

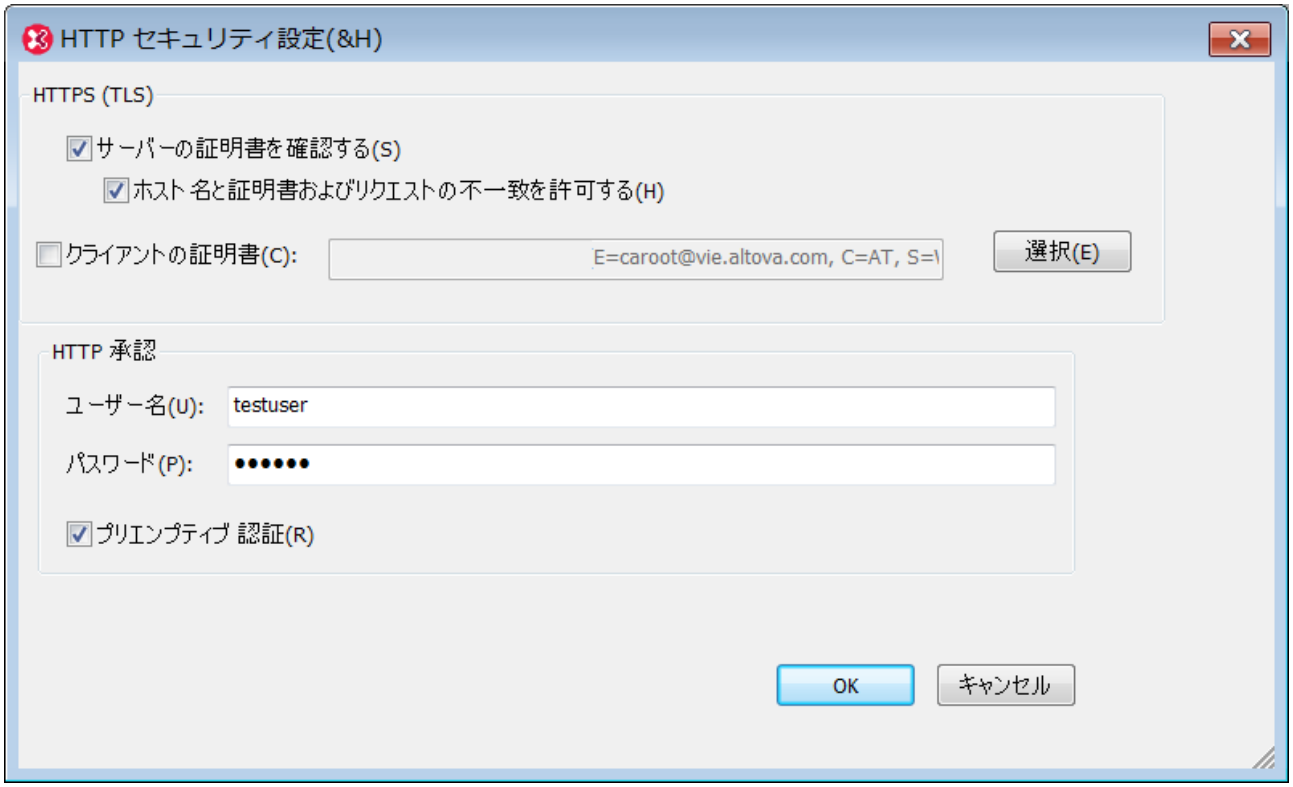
HTTP 出力ウィンドウの設定タブ内で次を設定することができます（i）接続のタイムアウト、（ii）リクエストのセキュリティ設定（*下のウィンドウ*）。9つのウィンドウのタブのそれぞれに接続の設定を個別に設定できることにご注意してください。下のスクリーンショット派 HTTPS URL のための設定を表示しています。

タイムアウト

XMLSpy がWeb サーバーと接続を試みる時間を秒数で指定することができます。接続が確立されずタイムアウトの時間に達すると、I/O Error 28: タイムアウト のメッセージが表示されます。タイムアウトの期間を指定しない場合、「無限」チェックボックスをチェックしてください。

セキュリティ設定

「編集」をクリックしてリクエストのセキュリティ設定を編集します。HTTP セキュリティ設定ダイアログが表示されます（*下のウィンドウ*）。HTTPS セキュリティ設定を指定し、そのタブにより作成されたリクエストのためのHTTP 認証資格情報を設定することができます。リクエストのターゲットのWeb サーバーがSSL を使用しない場合、HTTP 認証資格情報のみが使用されます。ターゲット Web サーバーが SSL を使用する場合、HTTPS セキュリティ設定とHTTP 認証資格情報が使用されます。



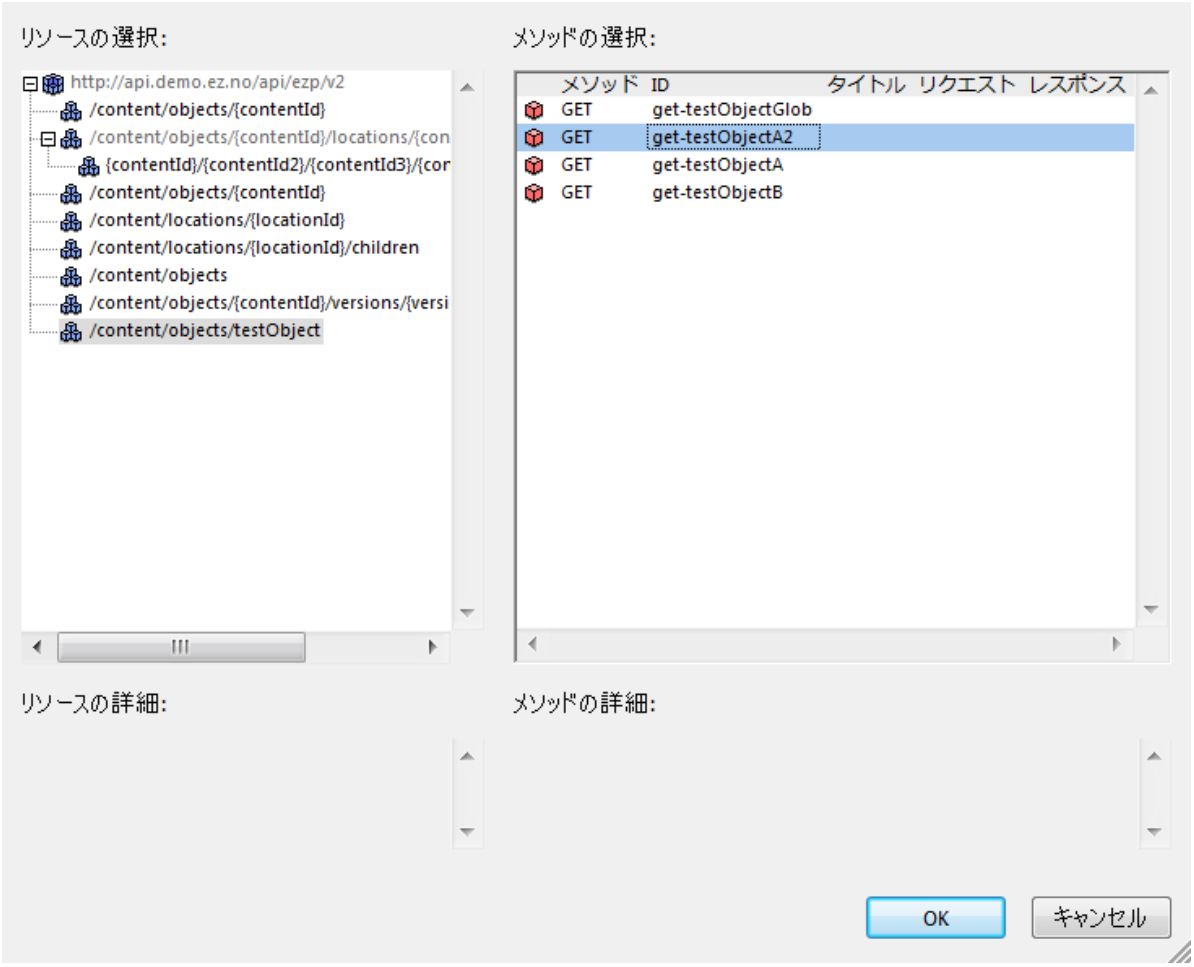
- *HTTPS セキュリティ設定:* デフォルトでは、サーバーの資格情報をチェックオプションがチェックされ、リクエスト内のホスト名が資格情報内のホスト名と異なる場合を指定することができます。クライアントの資格情報の使用を選択することもできます。(企業のネットワークなどの)、インターネット URL をターゲットする場合、(ローカル証明書ストア内にある) クライアント証明書をインターネットサーバー上の証明書を検証するために使用することがあります。
- *HTTP 認証:* サーバーへのリクエストの一部はユーザー認証を必要とする場合があります。このような場合、ユーザー名とパスワードをここに入力することができます。サーバーにより認証が必要とされる場合、認証は自動的に提供されます。それ以外の場合、サーバーが接続された後にプロンプトされる可能性があります。サーバーへの当初のリクエストが認証情報を含む場合、このプロセスはプリエンプティブ認証として称されます。プリエンプティブ認証がサーバーにより必要とされると、プリエンプティブ認証オプションを選択します。

16.2 送信するリクエストのインポート

HTTP 出力ウィンドウ[711]内で、Web Application Development Language (WADL) ファイルからリクエストをインポートし送信することができます。これはXMLSpy のW ADL インポートウィザードを使用しておこなうことができます。ウィザードはW ADL ファイルを開きファイルのリソースからリクエストを選択することにより、リクエストの編集することの出来るパラメーターを編集できるようにし、HTTP 出力ウィンドウ[711]にインポートします。以下をおこないます:

HTTP 出力ウィンドウ[711] 内では、W SDL 1.1、W SDL 2.0、またはWeb Application Development Language (WADL) ファイルからリクエストをインポートし、送信することができます。これはXMLSpy のW SDL/W ADL インポートウィザードを使用して行います。ウィザードはW SDL またはW ADL ファイルを開き、ファイルのW SDL エンドポイント、または、W ADL リソースからリクエストを選択し、リクエストの編集可能なパラメーターを編集できるようにし、HTTP 出力ウィンドウ[711]にリクエストをインポートします。以下を行ってください:

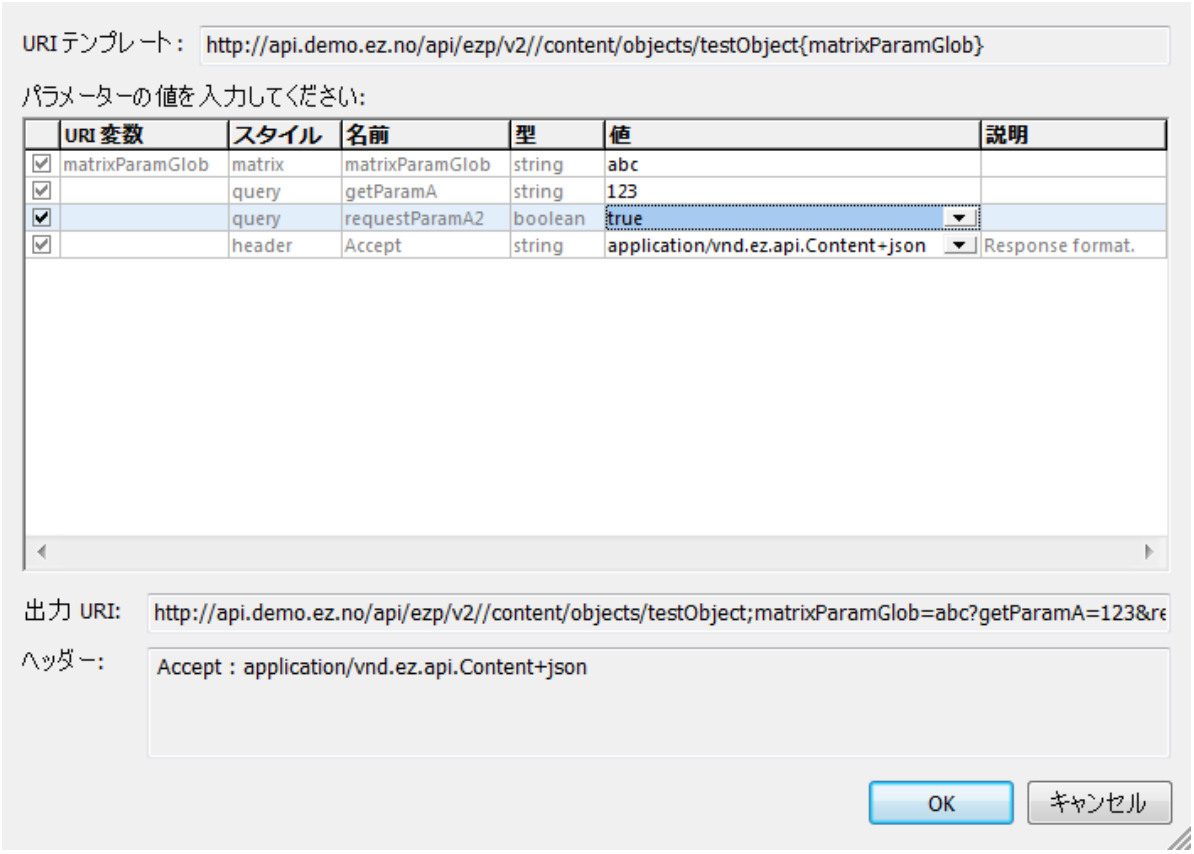
1. HTTP 出力ウィンドウ[711] 内で、W SDL/W ADL インポートウィザードを開始するために「インポート」ボタンをクリックします。
2. 表示されるファイル選択ダイアログ内で、インポートするリクエストを含むW ADL またはW SDL ファイルを参照し「OK」をクリックします。これによりW SDL/W ADL ウィザードのインポートが開始されます(*下のスクリーンショット*)。



3. 左側のペイン内で、(インポートするリクエストを含む)関連するW SDL エンドポイント、または、W ADL リソースを選択します。(*上のスクリーンショット参照*)。
4. 右側のペインでインポートする(W SDL オペレーション、または、W ADL メソッドである)リクエストを選択します(*上のスクリーンショット参照*)。W SDL を使用してインポートする場合以下の点に注意してください: (i) HTTP インポートはSOAP と

HTTP 拡張のために提供されており、(ii) サポートされるバインディング(SOAP とHTTP) がこの(右側の) ペイン内に表示されます。

5. 「OK」をクリックします。リクエストに1つまたは複数のパラメーターが含まれている場合、ウィザードの次のスクリーンは選択したリクエストのパラメーターを表示します(*下のスクリーンショット*)。それ以外の場合、リクエストはHTTP 出力ウィンドウ[711]にインポートされ、ウィザードは閉じられます。*下のポイント 7 を参照してください*。パラメーターはリクエストの一部です。検索リクエスト内では、例えば、1つのパラメーターが検索用語の場合があります。ウィザードはパラメーターの値をデータ型に対して検証し、3つの状態のいずれかを示します。ピンクの背景は、無効な値を示し、ベージュの背景はリクエスト内に入力される無効な値を示し、白い背景は有効な値を示しています。



6. HTTP リクエストパラメータースクリーン内で、必要に応じてパラメーターの値を(値列に)入力し編集します(*上のスクリーンショット*)。パラメータ値の一部の編集は許可されていない場合があることにご注意してください。パラメーターの使用を希望しない場合、有効化チェックボックスのチェックを解除して無効化します。この値のセルは編集できないように無効化されます。パラメーターの一部は必須であり、有効化チェックボックスはロックされ、チェックを解除することはできません。パラメーターの値を編集すると、*出力 URI* フィールド内でリクエストが作成されます。パラメーターグリッドには、1つ、または、複数の行がグリッドの下にヘッダーのために含まれています。これらのヘッダーはW ADL ファイルから来ており、W ADL ファイル内の定義に従い定義が許可されている場合、これらの値をグリッド内で編集することができます。ヘッダーのまとめはウィンドウの下のヘッダーフィールドにリストされています。
7. 「OK」をクリックします。HTTP 出力ウィンドウ[711] にリクエストがインポートされ、次の方法で表示されます: (i) メソッド リクエストのHTTP メソッドが表示されます。(ii) URL は、リクエストパラメーターをベースにして構成されます。(ii) リクエストのHTTP ヘッダーはヘッダータブに入力されます。(iv) POST または PUT リクエストがリクエストの場合、ボディタブ内にボディが入力されます。
8. 設定を変更する必要があるかを確認するために設定タブをクリックします。
9. 「送信」をクリックしてリクエストを送信します。

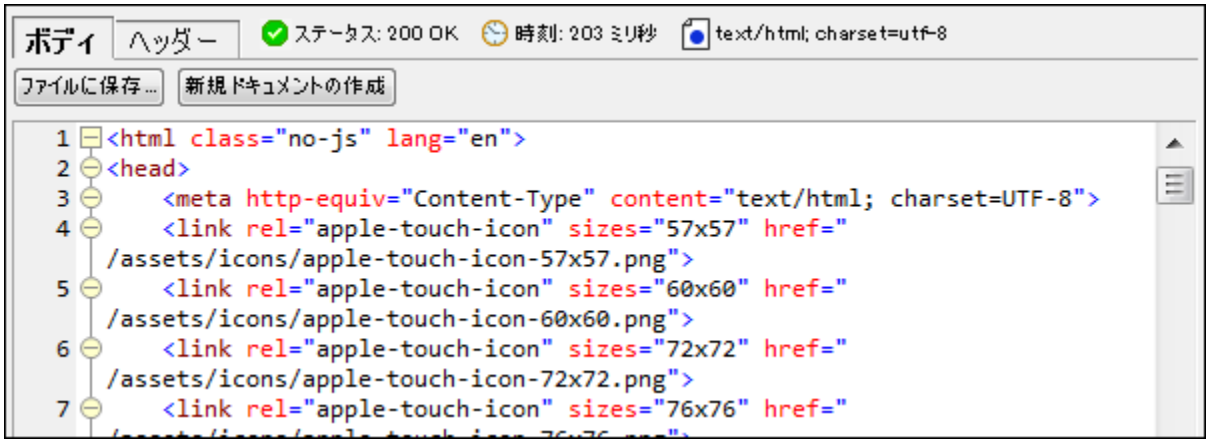
16.3 レスポンスの受信

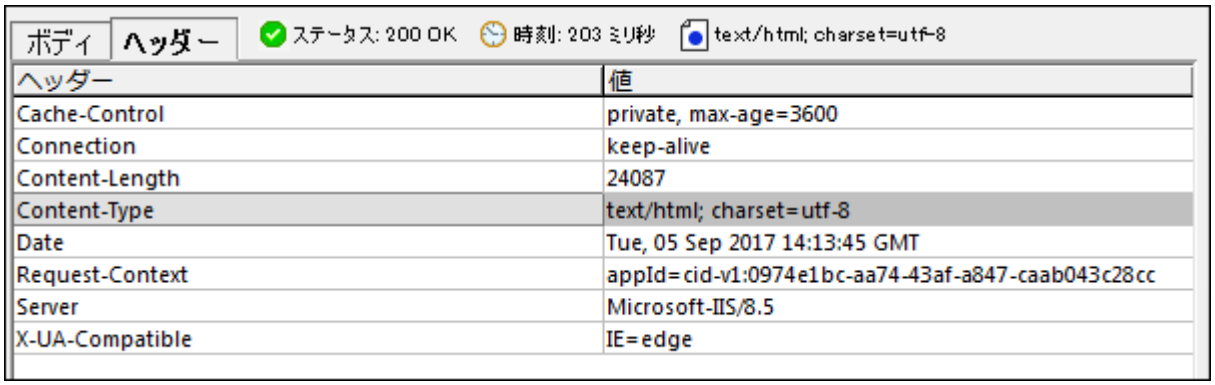
HTTP リクエストへのレスポンスはHTTP 出力ウィンドウの右側のペインで受信されます（*下のスクリーンショットを参照*）。

ボディとヘッダータブ名の右側には、レスポンスに関する以下の詳細がリストされています:

- （下で説明されている[719]）HTTP 状態コード
- 接続の確立時から最後のレスポンスチャンクが受信された時間
- レスポンスのコンテンツの型

レスポンスペインには2つのタブが存在します: *ボディ*（*左下のスクリーンショット*）と*ヘッダー*（*右下のスクリーンショット*）.





- *ボディタブ*: ドキュメントがHTML, XML, or JSON の場合、*レスポンス*のボディが構文色分けと行番号を用いて表示されます。タブには2つのボタンがあります: (i) ファイルにボディを保存する「**ファイルに保存する**」。(ii) XMLSpy 内で新規ドキュメントを表示する「**新規ドキュメントを作成する**」とGUI のメインウィンドウ内で新規に作成されたドキュメントが表示されます。新規に作成されたドキュメントは通常の方法で編集し、保存することができます。*レスポンス*のボディから*レスポンス*を作成することの出来ない場合、(例えばボディがイメージの場合など)「**新規ドキュメントを作成する**」ボタンが表示されます(*スクリーンショットを参照*)。
- *ヘッダータブ*: *レスポンス*のヘッダーが含まれています。コンテンツの型 ヘッダーはペインの上の部分に表示されます。

HTTP 状態コード

状態コードは次のように分類されます:

- 2XX は、成功した*リクエスト*のために使用されます。
- 3XX は、*リダイレクト*のために使用されます。
- 4XX は、*リクエスト*に問題がある場合使用されます。
- 5XX は、サーバーに問題がある場合使用されます。

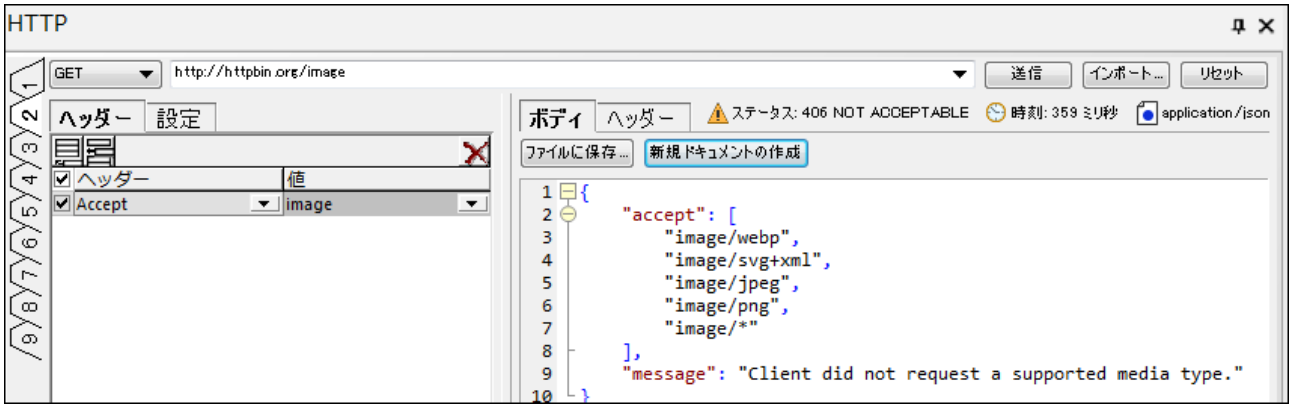
一般に使用されるコードの一部:

- *200 OK*: *リクエスト*の送信に成功した場合。
- *206 部分的内容*: *リソース*の一部のみが*リクエスト*されているため、サーバーは*リソース*の一部を返します。
- *301 恒久的に移動した* 与えられたURL に*リクエスト*がリダイレクトされます。
- *401 認証が必要である* リソースは認証を必要とし、認証が失敗したか、または、認証が与えられていません。
- *403 禁止されている* 有効な*リクエスト*ですが、サーバーがアクションを拒否しています。これは、ユーザーが必要な資格を有していない可能性を意味します。
- *404 未検出*: *リソース*が見つからなかった場合。
- *500 サーバー内部エラー*: 詳細なメッセージが存在しない場合送信される、ジェネリックなエラーメッセージ。

次も参照: *HTTP 状態コードの完全なリスト*。

ヘッダーの受け入れ

*リクエスト*のヘッダーの受け入れは、*レスポンス*内で受け入れられるコンテンツの型を指定します。例えば、`Accept=image` (*下の最初のスクリーンショット*) と`Accept=image/png` (*下の2番目のスクリーンショット*)の違いを確認してください。最初の場合、*レスポンス*に送信されるイメージの書式が指定されていないため、エラーの状態コードと詳細を含むJSON メッセージが送信されます。



(C) 2015-2021 Altova GmbH

17 XBRL

Altova Web サイト: XBRL タクソノミエディター、XBRL バリデーター

XMLSpy の XBRL ビュー[243] は XBRL タクソノミのグラフィカルな概要を表示し、タクソノミのインテリジェントな編集機能を搭載した XBRL タクソノミエディターです。このセクションでは、XBRL ビューでタクソノミの作成や編集を行う方法を説明します。

このセクションは以下のように構成されます:

- XBRL タクソノミ マネジャー[722] は XMLSpy との使用のためにタクソノミをインストール、更新、管理する XBRL タクソノミ マネジャーツールの使用法について説明しています。
- 基本のプロシージャ[738] は、最も必要なコンポーネントを含むタクソノミの作成方法を説明します。
- 追加プロシージャ[763] は優先ラベルとの作業および複製ファクトとの作業などの追加機能について説明しています。
- XBRL フォーミュラエディター[767] は、XBRL ビューを使用して XBRL フォーミュラと作業する方法を説明します。
- XBRL テーブル定義 エディター[791] は、テーブル構造[794]、XBRL テーブルの定義のためのエディターの使用方法、およびテーブル レイアウト プレビュー[817] の使用方法を説明します。また、テーブルセットと共に使用されるテーブル パラメーターを含む、テーブル パラメーター[814] の使用方法についても説明しています。
- XBRL 内の検索[829] は、XMLSpy の XBRL-固有のグラフルな検索機能について説明しています。

より詳細な情報は、編集ビュー | XBRL ビュー[243] と XBRL メニュー[1391] のコマンドに関する記述を参照ください。例えば、(XBRL ビューにあるような) タクソノミのドキュメンテーションの生成に関しては、ユーザーリファレンス | XBRL メニュー | ドキュメンテーションの生成[1398] を参照ください。

XBRL ビューで表示される XBRL ファイルに対する XML 署名[370] は独立したファイルとして作成することができます。署名の詳しい使用方法については、XML 署名[370] のセクションを参照ください。

XMLSpy 内の US-GAAP と他のタクソノミのサポート

XMLSpy は、以下のタクソノミをサポートします:

- US-GAAP 1.0/2009/2011/2012/2013/2014/2015/2016/2017/2018/2019/2020
- IFRS

US-GAAP (2013 と 2014) の最新バージョンは、XMLSpy とともにインストールされています。古い US-GAAP タクソノミを含む、追加タクソノミを、Altova Web サイトから無料でダウンロードすることのできる、タクソノミインストーラーを介して、使用することができます。

XBRL 証明書

XMLSpy は XBRL International により XBRL 認定を受けました。XBRL 証明書に関する説明は、XBRL ソフトウェア証明書を参照してください。

Altova Web サイト: XBRL タクソノミエディター、XBRL バリデーター

17.1 XBRL タクソノミ マネジャー

XBRL タクソノミ マネジャーはXMLSpy を含むAltova XBRL-が有効化されているアプリケーション全体で使用するためにXBRL タクソノミをインストールし管理する中央化した方法を提供するツールです。W indows ではXBRL タクソノミ マネジャーには視覚的なユーザーインターフェイスがあり、コマンドラインでも使用可能です。Linux とMac* 上ではコマンドラインでのみツールを使用することができます。

> * Linux とmacOS オペレーティングシステムをこれらのオペレーティングシステム上でXBRL タクソノミ マネジャーをMapForce Server、StyleVision Server、またはRaptorXML+XBRL Server などのAltova クロスプラットフォームサーバーと共に実行している場合に適用可能です。

XBRL タクソノミ マネジャーは以下の機能を搭載しています:

- 使用中のコンピューターにインストールされているXBRL タクソノミをビューして、新規のバージョンをダウンロード可能か確認してください。
- Altova 製品のリリースサイクルに関わらずXBRL タクソノミの新しいバージョンをダウンロードします。全てのタクソノミはXBRL タクソノミ マネジャーによりアクセス可能なオンラインベースのパッケージストレージ上でAltova により管理されており、使用可能になるとすぐにダウンロードすることが可能です。
- タクソノミの複数のバージョン(または、必要な場合全てのバージョン)をインストールまたはアンインストールする
- 単一のXBRL タクソノミは[パッケージ]を表しており、他のタクソノミ上に依存関係が存在する場合があります。特定のタクソノミをインストールまたはアンインストールすることを選択すると、依存関係を持つタクソノミが検知され、自動的にインストールまたはアンインストールされます。グラフィカルなユーザーインターフェイス(または適用できる場合コマンドラインは)依存が追加または削除されることを通知します。
- XBRL タクソノミ マネジャーにより管理されるXBRL タクソノミはインスタンス、またはスキーマドキュメントインのURI レファレンスがインターネットからファイルを抽出することとは異なり、ローカルファイルから解決されることを有効化するXML カタログ メカニズムから利益を得ることができます。通常パフォーマンス上の理由からリモート URI からのスキーマのレゾルーションが実質的ではない場合、大きなXBRL タクソノミの場合これはとても重要です。

XBRL タクソノミ マネジャーはXBRL タクソノミの管理者がAltova XBRL-有効化されたアプリケーション[722]の一つを使用していることを必要とします。欧州銀行監督局レポーティングフレームワークタクソノミ、US-GAAP 金融レポーティングタクソノミ、他の多種の国別、または、ドメイン固有のXBRL タクソノミが含まれます。完全なリストを確認するにはXBRL タクソノミ マネジャーを実行または`list` コマンドをコマンドラインで実行します 。を参照してください。

カスタムXBRL タクソノミ

XBRL タクソノミ マネジャーに含まれるカスタムXBRL タクソノミと作業する場合、これらを以下のように有効化します:

- Altova XBRL-有効化済みのデスクトップアプリケーションから**ツール | オプション** メニューコマンドを実行し、**XBRL | タクソノミ パッケージ** 設定ページに移動します。設定ページからカスタムXBRL タクソノミの.zip パッケージを参照してください
- サポートするコマンドまたはAPI メソッドを実行する場合Altova XBRL-有効化済みサーバーアプリケーションから―`taxonomy-package` または`--taxonomy-package-config-file` オプションを提供してください。例えば、これらのオプションは**`valxbrl`** または**`valxbrltaxonomy`** などのRaptorXML+XBRL Server 内のXBRL 検証コマンドまたはMapForce Server 内の**`run`** コマンド(または関連するAPI メソッド)によりサポートされています。

Altova XBRL-有効化済みアプリケーション

以下のAltova アプリケーションはXBRL-有効化済みで、XBRL タクソノミ マネジャーにより提供される機能によりメリットを受けています:

- Excel のためのAltova XBRL アドイン(EBA, Solvency II)
- MapForce Enterprise Edition
- MapForce Server
- MapForce Server Advanced Edition

- RaptorXML+XBRL Server
- StyleVision Server
- StyleVision Enterprise Edition
- XMLSpy Enterprise Edition

XBRL タクソノミ マネジャーを使用したXBRL タクソノミ内での変更は同じコンピューターにインストールされている場合、上にリストされたすべてのアプリケーションで反映されます。

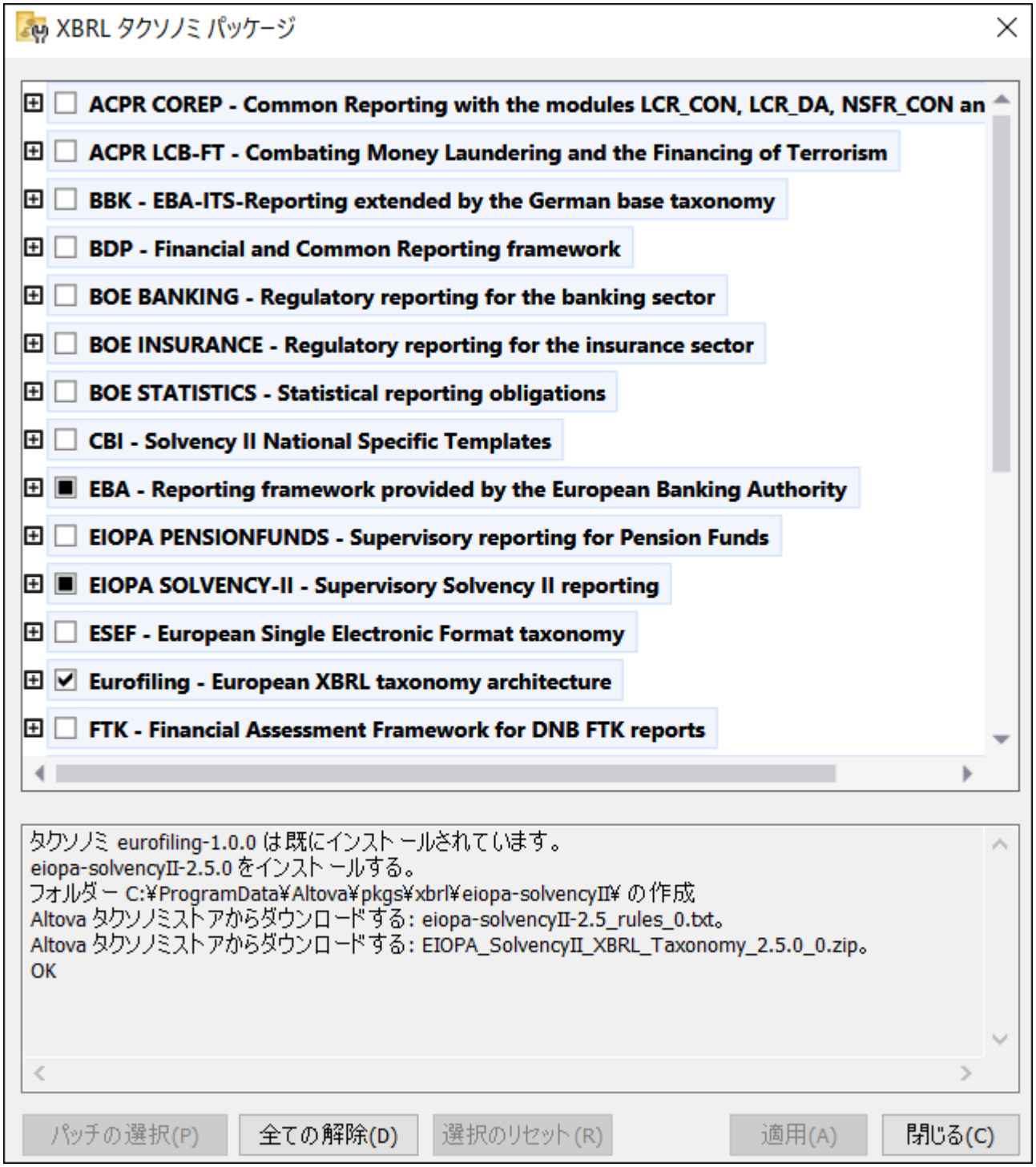
インストール

Altova XBRL-有効化済みアプリケーション、または **Altova Mission Kit Enterprise Edition** をインストールするとXBRL タクソノミ マネジャーは自動的にインストールされます。同様に、Altova XBRL-有効化済みアプリケーション、または **Altova Mission Kit Enterprise Edition** をコンピューターからアンインストールすると自動的にアンインストールされます。

しくみ

Altova は製品内で使用されている全てのXBRL タクソノミを保管するオンラインタクソノミストレージを管理しています。このタクソノミストレージは定期的に更新されます。例えば、発行機関により関連するタクソノミの新規のバージョンがリリースされた後などに更新されます。

XBRL タクソノミ マネジャーをグラフィカルなユーザーインターフェイスで実行すると、タクソノミをインストール、更新またはアンインストールすることができるダイアログボックス内に使用可能な最新のタクソノミの情報が表示されます。コマンドラインインターフェイスで同じアクションを実行することができます。



Altova W eb サイト (https://www.altova.com/download) からダウンロードされた running .altova_ taxonomies ファイルを実行してタクソノミをインストールすることができます。更にXBRL タクソノミをロードするアクションを取る場合 XMLSpy は不足するXBRL タクソノミをインストールするようにプロンプトします。

タクソノミのインストール方法に関わらず、インストール済みのタクソノミに関する情報はコンピューター内で中央化された場所でトラックされてい

ます。これはローカルキャッシュディレクトリとして知られています。ローカルキャッシュディレクトリには(オンデマンドでインストールされる実際のタクソノミファイルを除き) Altova パッケージに関する情報が含まれています。ローカルキャッシュディレクトリは以下のパスで見つけることができます:

Linux	`/var/opt/Altova/pkgs`
macOS	`/var/Altova/pkgs`
Windows	`C:\ProgramData\Altova\pkgs`

ローカルコンピューターにオンラインストレージの最新の状態を作成するため、ローカルキャッシュディレクトリは時々自動的に更新されます。更に具体的には、キャッシュは以下のように更新されます:

- XBRL タクソノミマネジャーを実行する場合。
- 同じカレンダーの日付で XMLSpy を実行する場合。
- XMLSpy が既に実行中の場合、キャッシュディレクトリは 24 時間ごとに更新されます。
- コマンドラインインターフェイスから更新を実行しローカルキャッシュをオンラインストレージからオンデマンドで更新することができます。

タクソノミをインストールまたはアンインストールすると、ローカルキャッシュディレクトリは使用可能な情報、およびインストールされているタクソノミの情報、およびタクソノミファイル自身と共に自動的に更新されます。

ローカルキャッシュディレクトリはインストールまたはアンインストールするタクソノミをベースに自動的に管理されます。手動で変更または削除されてはなりません。XBRL タクソノミ マネジャーを元の状態にリセットする必要がある場合、コマンドラインインターフェイスの `reset` コマンドを実行してから `initialize` コマンドを実行します。または `reset` コマンドを `-i` オプションと共に実行してください。

17.1.1 XBRL タクソノミ マネジャー の実行

XBRL タクソノミ マネジャーを実行するには、以下の1つを行ってください:

- XMLSpy の[ツール] メニュー内で、[XBRL タクソノミマネージャー]を選択します。
- Altova W eb サイトからダウンロードされた.altova_ taxonomies 拡張子を持つファイルを(ダブルクリックして) 実行します。
- W indows コントロールパネルから Altova タクソノミマネジャー エントリを右クリックして、変更 または アンインストール をコンテキストメニューから選択します。

更に、XMLSpy のインストール後チェックボックス Altova タクソノミマネージャーの呼び出し がインストールウィザードの最後のページで使用可能になります。

コマンドライン インターフェイス

XBRL タクソノミ マネジャーをコマンドインターフェイスから実行する方法:

1. コマンドプロンプトを開きディレクトリを **C:\ProgramData\Altova\SharedBetweenVersions** に変更します。
2. コマンドラインでヘルプを表示するには、以下を実行します:

```
TaxonomyManager.exe --help
```

17.1.2 タクソノミのインストール

タクソノミをインストールする方法:

1. 実行 する。
2. インストールするタクソノミまたはタクソノミ バージョンの横のチェックボックスを選択し [適用]をクリックします。

または.**altova_taxonomies** 拡張子を持つファイルを Altova W eb (https://www.altova.com/taxonomy-manager)サイトからダウンロードし、**.altova_taxonomies** ファイルを実行するためにダブルクリックします。**.altova_taxonomies** ファイルを実行する際に XBRL タクソノミ マネジャーが開かれます。

以下の点に注意してください:

- XBRL タクソノミ マネジャーからタクソノミをインストールまたはアンインストールすると同じコンピューター上のすべてのユーザーアカウントに影響を及ぼします。
- XBRL タクソノミ マネジャーからタクソノミをインストールまたはアンインストールすると同じコンピューター上にインストールされている Altova XBRL-有効化済みのすべてのアプリケーションに影響を与えます。
- 現在のタクソノミは他のタクソノミ上に依存関係があり、依存するタクソノミもインストールされます(または適用可能な場合アンインストールされます)。

コマンドライン インターフェイス

タクソノミをインストールするには、以下を実行します:

```
TaxonomyManager.exe install FILTER...
```

`FILTER` が以下を意味する箇所:

1. フォーマット `<name>-<version>` 内のタクソノミ識別子の例: **eba-2.10**、**us-gaap-2020.0**。使用可能なすべてのタクソノミ識別子をビューするには `list` コマンドを実行します。
2. Altova W eb サイトからダウンロードされた**altova_taxonomies** ファイル。

オンデマンドでタクソノミをインストールする

XMLSpy が特定のXBRL タクソノミを検知すると使用中のコンピューターにインストールされる必要があります。オンデマンドでタクソノミをインストールするようにプロンプトされる場合があります。
不足するタクソノミをインストールする場合 XBRL タクソノミ マネジャーはインストールとこの情報をトラックするためにローカルキャッシュディレクトリの更新を実行します。ツール | XBRL タクソノミ マネジャーから XBRL タクソノミ マネジャーを実行して以前にインストールされたすべてのタクソノミを常に確認することができます。

無視 を選択すると、XMLSpy はよしいパフォーマンスまたはエラーを引き起こす可能性のあるローカルファイルの代わりに URI にを公開されたオリジンから解決します。

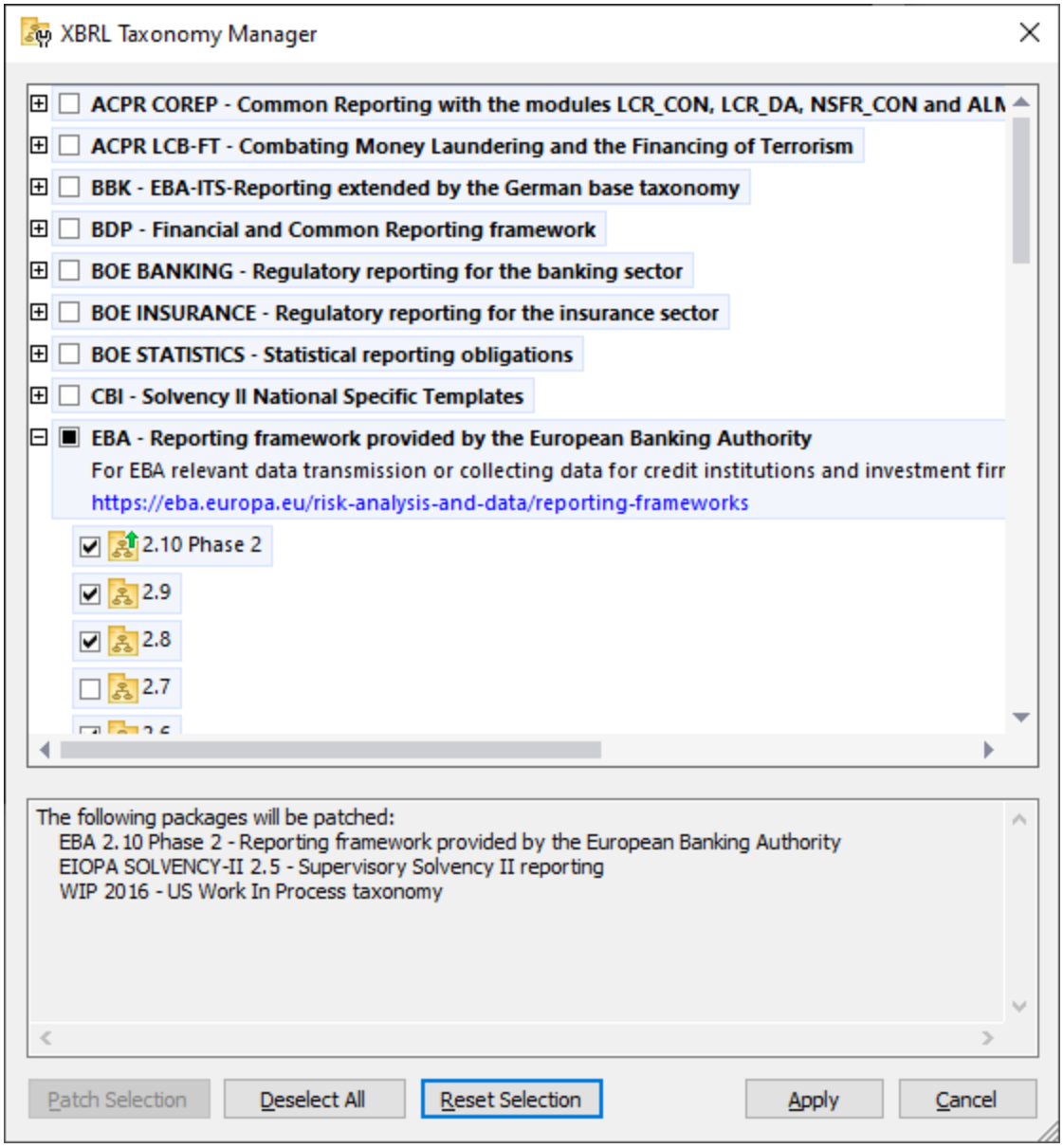
パッチの適用

不定期にXBRL タクソノミは発行元からパッチを受け取ります。XBRL タクソノミマネジャーがパッチが使用可能であることを検知すると、以下が表示されます:

- W indows の視覚的なユーザーインターフェイスを介してXBRL タクソノミマネジャーを使用すると、対応するXBRL タクソノミは アイコンと共に表示されます。
- コマンドラインまたはLinux/macOS システムを使用すると、`list -u` コマンドと共に実行可能ファイルを実行する際にリストされているパッチを持つXBRL タクソノミがリストされます。

Windows 上でパッチを適用する方法:

1. [パッチの選択] ボタンをクリックします。 から への変更を修飾する各 XBRL タクソノミのアイコンとダイアログボックスは適用されるパッチについて通知します。例:



メモ **パッチの選択** ボタンは現在インストールされている XBRL タクソノミのためにパッチが使用な箇所でのみ有効化されています。

2. [適用]をクリックします。

コマンドラインインターフェイスでパッチを適用する方法:

1. `list -u` コマンドを実行します。パッチ更新が使用可能なタクソノミをリストします。

2. `upgrade` コマンドを実行してパッチをインストールします。

更に詳しい情報に関してはコマンドラインインターフェイスを参照してください。

17.1.3 インストール済みのタクソノミをビューする

視覚的なユーザーインターフェイスからインストール済みのすべてのタクソノミをビューするにはXBRL タクソノミ マネジャーを 実行 します。タクソノミ（または、タクソノミのバージョン）の横の選択済みのチェックボックスはそのタクソノミがインストール済みであることを示しています。

コマンドライン インターフェイス

コマンドラインインターフェイスからすべての使用可能なタクソノミをビューするには、以下を実行します:

```
TaxonomyManager.exe list
```

インストール済みのタクソノミのみをビューするには、以下を実行します:

```
TaxonomyManager.exe list -i
```

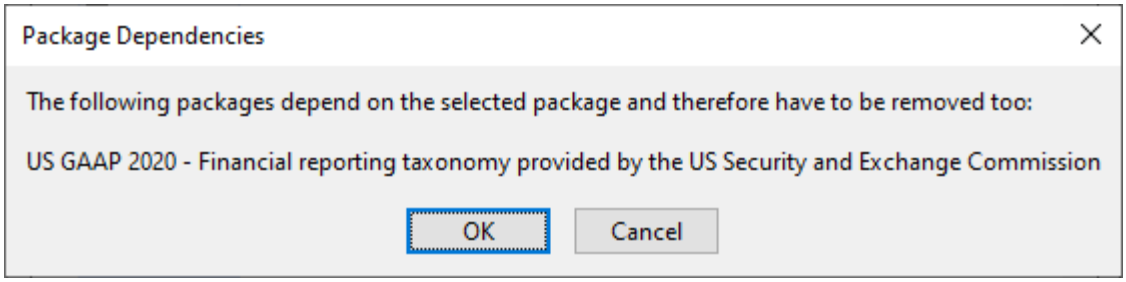
新規のバージョンが使用可能なタクソノミのみをビューするには、以下を実行します:

```
TaxonomyManager.exe list -u
```

17.1.4 タクソノミのアンインストール

タクソノミをアンインストールする方法:

1. 実行 する。
2. アンインストールするタクソノミまたはタクソノミバージョンの横のチェックボックスのチェックを解除します。選択されたタクソノミが他のタクソノミパッケージに依存する場合、依存関係も削除されることを通知するダイアログボックスが開かれます。例:



3. [適用] をクリックしてタクソノミとその依存関係を削除します。

コマンドライン インターフェイス

タクソノミをアンインストールするには、以下を実行します:

```
TaxonomyManager.exe uninstall FILTER...
```

FILTER が以下を意味する箇所:

1. フォーマット <name>-<version> 内のタクソノミ識別子の例: **eba-2.10**、**us-gaap-2020.0**。使用可能なすべてのタクソノミ識別子をビューするには list コマンドを実行します。
2. Altova W eb サイトからダウンロードされた**altova_taxonomies** ファイル。

17.1.5 コマンドライン インターフェイス

XBRL タクソノミ マネジャーをコマンドラインで呼び出すには場合実行可能ファイルのパスを知っている必要があります。デフォルトで XBRL タクソノミ マネジャー実行可能ファイルを以下のパスで見つけることができます:

*Linux**	/opt/Altova/<%APPNAME-UL%>2021/bin/taxonomymanager
*macOS**	/usr/local/Altova/<%APPNAME-UL%>2021/bin/taxonomymanager
Windows	C:\ProgramData\Altova\SharedBetweenVersions\TaxonomyManager.exe

＊Linux とmacOS パスをこれらのオペレーティングシステム上で XBRL タクソノミ マネジャーを MapForce Server、StyleVision Server、または RaptorXML+XBRL Server などの Altova クロスプラットフォームサーバーと共に実行している場合に適用可能です。

規則としては、このドキュメンテーションは特定のコマンドを説明する際に実行可能ファイルのフルパスを省略し、実行可能ファイルのなめを使用する代わりに **<exec>** を使用します。例:

```
<exec> help
```

<exec> は実行可能ファイルのパスまたは名前です。

XBRL タクソノミ マネジャーをインストールディレクトリから呼び出す方法

実行可能ファイルをフルパスを入力することなく呼び出すには、現在のディレクトリを以下のいずこに変更します:

Linux	cd /opt/Altova/<%APPNAME-UL%>2021/bin
macOS	cd /usr/local/Altova/<%APPNAME-UL%>2021/bin
Windows	cd "C:\ProgramData\Altova\SharedBetweenVersions"

相対的なパスを使用してコマンドを実行し実行可能ファイルを呼び出すことができます。

Linux	sudo ./taxonomymanager help
macOS	sudo ./taxonomymanager help
Windows	TaxonomyManager.exe help

メモ Linux とmacOS システム上では、プレフィックス `./` は実行可能ファイルが現在のディレクトリ内にあることを示しています。`sudo` はコマンドがルート特権と共に実行されなければならないことを示しています。

XBRL タクソノミ マネジャーをディレクトリから呼び出す方法

実行可能ファイルをディレクトリから呼び出すには絶対パスを使用して参照してください。または、プログラムを実行可能ファイル名を入力するだけで呼び出す場合、使用中のオペレーティングシステムのPATH 環境変数を編集することにより、実行可能ファイルのディレクトリへのフルパスが含まれるようになります。PATH 環境変数を変更する方法に関しては、使用中のオペレーティングシステムのドキュメンテーションを参照してください。

メモ

- PATH 環境変数の変更後、ターミナルウィンドウを閉じ、変更の効果が反映されるように新規のウィンドウを開く必要がある場合があります。
- Linux とmacOS 上では、`sudo` の使用はユーザーのPATH を考慮していません。

コマンドライン構文

コマンドラインを使用するための一般的な構文は以下の通りです:

```
<exec> -h | --help | --version | <command> [options] [arguments]
```

上のリスト内で垂直バー `|` は相互に排他的なアイテムを分割します。角かっこは `[]` は任意のアイテムを示しています。基本的には、`--h`, `--help`、または `--version` オプションまたはコマンドが後に続く実行可能ファイルパスを入力することができます。各コマンドはオプションと引数を持つ場合があります。コマンドのリストは以下のセクションで説明されています:

17.1.5.1 help

このコマンドはXBRL タクソノミ マネジャー実行可能ファイルに関連するコマンドに関するコンテキストのヘルプを提供します。

構文

```
<exec> help [command]
```

[command] が有効なコマンド名を指定するオプションの引数である個所。

リマーク

`-h` または `--help` が後に続くコマンドを入力してヘルプを呼び出すことができます。例:

```
<exec> list -h
```

実行可能ファイルの後に `-h` または `--help` を入力して一般的なヘルプを呼び出すことができます。例:

```
<exec> --help
```

サンプル

次のコマンドは list コマンドに関するヘルプを表示します:

Linux、*macOS*	`./taxonomymanager help list`
Windows	`TaxonomyManager.exe help list`

17.1.5.2 info

このコマンドは引数として与えられたタクソノミのそれぞれの詳細を表示します。詳細にはタイトル、バージョン、詳細、後悔元、依存レファレンスが含まれています。

構文

```
<exec> info [options] FILTER...
```

複数のタクソノミを指定するには FILTER を必要な回数繰り返します。

引数

FILTER	名前内にこの文字列を含むタクソノミをリストします（例えば、**eba** または **us-gaap-2020.0**）。この引数を複数回指定することができます。 識別子のフルリストをビューするには list コマンドを実行します。

オプション

--help, --h	コマンドラインでこのコマンドに関するヘルプを表示します。

例

以下のコマンドは **eba-2.10** と **us-gaap-2020.0** タクソノミに関する詳細を表示します:

Linux、*macOS*	`./taxonomymanager info eba-2.10 us-gaap-2020.0`
Windows	`TaxonomyManager.exe info eba-2.10 us-gaap-2020.0`

17.1.5.3 initialize

このコマンドは XBRL タクソノミ マネジャー環境を初期化します。タクソノミに関するすべての情報が保管されている個所でキャッシュディレクトリを作成します。XBRL タクソノミ マネジャーを含む Altova アプリケーションをインストールすると初回に初期化が実行されるため、通常このコマンドを実行する必要はありません。reset コマンドを実行後に通常このコマンドを 実行する必要があります。

構文

```
<exec> initialize [options]
```

このコマンドのエイリアスは `init` です。

オプション

`--help, --h`	コマンドラインでこのコマンドに関するヘルプを表示します。
`--silent, --s`	エラーメッセージのみを表示します。デフォルトの値は **false** です。
`--verbose, --v`	実行中の追加情報を表示します。デフォルトの値は **false** です。

17.1.5.4 install

このコマンドは1つまたは複数のタクソノミをインストールします。以下の点に注意してください:

- XBRL タクソノミ マネジャーからタクソノミをインストールまたはアンインストールすると同じコンピューター上のすべてのユーザーアカウントに影響を及ぼします。
- XBRL タクソノミ マネジャーからタクソノミをインストールまたはアンインストールすると同じコンピューター上にインストールされている Altova XBRL-有効化済みのすべてのアプリケーションに影響を与えます。
- 現在のタクソノミは他のタクソノミ上に依存関係があり、依存するタクソノミもインストールされます(または適用可能な場合アンインストールされます)。

構文

```
<exec> install [options] FILTER...
```

複数のタクソノミを指定するには `FILTER` を必要な回数繰り返します。

引数

`FILTER`	`FILTER` が以下を意味する箇所: 1. フォーマット `<name>-<version>` 内のタクソノミ識別子の例: **eba-2.10**、**us-gaap-2020.0**。使用可能なすべてのタクソノミ識別子をビューするには `list` コマンドを実行します。 2. Altova W eb サイトからダウンロードされた**altova_taxonomies** ファイル。 一意の場合省略された識別子を使用することができます、例えば、**eba**。省略された識別子を使用する場合、そのタクソノミの使用可能な最新バージョンがインストールされます。

オプション

`--help, --h`	コマンドラインでこのコマンドに関するヘルプを表示します。

--silent, --s	エラーメッセージのみを表示します。デフォルトの値は**false** です。
--verbose, --v	実行中の追加情報を表示します。デフォルトの値は**false** です。

サンプル

以下のコマンドは最新の**eba**（欧州銀行監督局）と**us-gaap**（米国会計基準）タクソノミをインストールします:

Linux、macOS	./taxonomymanager install eba us-gaap
Windows	TaxonomyManager.exe install eba us-gaap

17.1.5.5 list

このコマンドを使用して以下の方法の1つによりコマンドラインでタクソノミをリストします。

- 使用可能なタクソノミを全てリストします
- 特定のタクソノミをリストします
- インストール済みのタクソノミのみをリストします
- アップグレードが必要なタクソノミのみをリストします

構文

```
<exec> list [options] [FILTER...]
```

このコマンドは ls を使用して省略することができます。

引数

FILTER	名前内にこの文字列を含むタクソノミのみをリストします。この引数を複数回指定することができます。

オプション

--help, --h	コマンドラインでこのコマンドに関するヘルプを表示します。
--installed, --i	インストール済みのタクソノミのみをリストしますデフォルトの値は**false** です。
--upgradeable, --u	パッチ更新が使用可能なタクソノミのみをリストします。デフォルトの値は**false** です。

例

使用可能な全てのタクソノミをリストするには、以下を実行します:

Linux、macOS	./taxonomymanager list
Windows	TaxonomyManager.exe list

インストール済みのタクソノミのみをリストするには、以下を実行します:

Linux、macOS	`./taxonomymanager list -i`
Windows	`TaxonomyManager.exe list -i`

名前内に"eba" または"us-gaap" を含むすべてのタクソノミをリストするには、以下を実行します:

Linux、macOS	`./taxonomymanager list eba us-gaap`
Windows	`TaxonomyManager.exe list eba us-gaap`

17.1.5.6 reset

このコマンドはすべてのインストールされたタクソノミとキャッシュディレクトリを削除します。

警告: このコマンドはすべてのインストールされたタクソノミとその情報を削除します。

このコマンドの実行後、キャッシュディレクトリを再作成するために `initialize` コマンドを実行してください。または `reset` コマンドを `-i` オプションと共に実行してください。

`reset -i` は製品オリジナルのインストールを復元するため、リセット後に更に `update` コマンドを実行することが奨励されます。または、`-i` と `-u` オプションを使用してコマンドをリセットします。

構文

```
<exec> reset [options]
```

オプション

`--help, --h`	コマンドラインでこのコマンドに関するヘルプを表示します。
`--init, --i`	XBRL タクソノミマネジャー環境をリセット後初期化してください。有効な値は**true** または**false** です。デフォルトの値は**false** です。
`--silent, --s`	エラーメッセージのみを表示します。デフォルトの値は**false** です。
`--update, --u`	XBRL タクソノミマネジャー環境をリセット後に初期化し更新してください。有効な値は**true** または**false** です。デフォルトの値は**false** です。
`--verbose, --v`	実行中の追加情報を表示します。デフォルトの値は**false** です。

例

XBRL タクソノミ マネジャーをリセットするには、以下を実行します:

Linux、*macOS*	`./taxonomymanager reset`
Windows	`TaxonomyManager.exe reset`

17.1.5.7 setdeflang

このコマンドはXBRL タクソノミ マネジャーの言語を設定します。

構文

```
<exec> setdeflang language
```

`language` が言語コードを提供する必須の引数の箇所。このコマンドのエイリアスは `sdl` です。

引数

`language`	設定する言語有効な言語は英語 (`en`)、フランス語、(`fr`)、ドイツ語 (`de`)、日本語 (`ja`)、およびスペイン語 (`es`) です。

オプション

`--help, --h`	コマンドラインでこのコマンドに関するヘルプを表示します。

サンプル

言語をスペイン語に設定するには、以下を実行します:

Linux、*macOS*	`./taxonomymanager setdeflang es`
Windows	`TaxonomyManager.exe setdeflang es`

17.1.5.8 uninstall

このコマンドは1つまたは複数のタクソノミをアンインストールします。デフォルトでは現在のタクソノミにより参照されるタクソノミもアンインストールされます。現在のタクソノミだけをアンインストールし、参照されたタクソノミを保つには オプション `--k` を設定します。

構文

```
<exec> uninstall FILTER...
```

複数のタクソノミを指定するには `FILTER` を必要な回数繰り返します。

引数

FILTER	FILTER が以下を意味する箇所: 1. フォーマット <name>-<version> 内のタクソノミ識別子の例: **eba-2.10**、**us-gaap-2020.0**。使用可能なすべてのタクソノミ識別子をビューするには list コマンドを実行します。 2. Altova W eb サイトからダウンロードされた**altova_taxonomies** ファイル。

オプション

--help, --h	コマンドラインでこのコマンドに関するヘルプを表示します。
--keep-references, --k	このオプションが設定されると、参照されたタクソノミはアンインストールされません。デフォルトの値は **false** です。
--silent, --s	エラーメッセージのみを表示します。デフォルトの値は **false** です。
--verbose, --v	実行中の追加情報を表示します。デフォルトの値は **false** です。

サンプル

以下のコマンドは **eba-2.10** と **us-gaap-2020.0** タクソノミとその依存関係をアンインストールします:

Linux、macOS	`./taxonomymanager uninstall eba-2.10 us-gaap-2020.0`
Windows	`TaxonomyManager.exe uninstall eba-2.10 us-gaap-2020.0`

17.1.5.9 update

このコマンドクエリはオンラインストレージから使用可能なタクソノミのリストをクエリし、ローカルのキャッシュディレクトリを更新します。この情報の更新は明示的に行われ、reset と initialize を実行した場合を除いてこのコマンドを実行する必要はありません。

構文

```
<exec> update [options]
```

オプション

--help, --h	コマンドラインでこのコマンドに関するヘルプを表示します。
--silent, --s	エラーメッセージのみを表示します。デフォルトの値は **false** です。
--verbose, --v	実行中の追加情報を表示します。デフォルトの値は **false** です。

サンプル

以下のコマンドはタクソノミの情報を明示的に更新します。

Linux、macOS	`./taxonomymanager update`
Windows	`TaxonomyManager.exe update`

17.1.5.10 upgrade

このコマンドは全ての対象のタクソノミを使用可能な最新 *patch* バージョンにアップグレードします。すなわち、特定のリリースのパッチレベルでのみ更新を行います。このコマンドを実行することは使用可能なタクソノミを更新する場合のみ意味があります。`list -u` コマンドを実行して更新可能なタクソノミを識別することができます。

構文

```
<exec> upgrade [options]
```

オプション

`--help, --h`	コマンドラインでこのコマンドに関するヘルプを表示します。
`--silent, --s`	エラーメッセージのみを表示します。デフォルトの値は **false** です。
`--verbose, --v`	実行中の追加情報を表示します。デフォルトの値は **false** です。

17.2 基本のプロシージャ

基本のプロシージャのセクションでは、最も需要なコンポーネントを含むタクソノミの作成方法を説明します。このセクションは以下のように構成されます:

- 最初に新たに作成されたタクソノミと既存のタクソノミ[738]の違いと、その重要性について簡単に説明します。その後、XBRL タクソノミを構成するファイル[739]と、それらがどのようにXBRL ビューに表示されるかの説明を行ないます。
- タクソノミの新規作成[741]セクションからは、XBRL ビューでタクソノミを構築する方法を順を追って解説します。この中で新規タクソノミの作成ウィザードの説明も行ないます。各セクションの終わりには、XBRL ビューを使って、オリジナルのタクソノミを作成するための例題が与えられます。例題は、その前のセクションの例題により作成されたタクソノミを元に構築され、そのセクションで与えられた情報を使った演習やテストとして使用することができます。

17.2.1 タクソノミ: 新規と既存

XMLSpy のXBRL ビューでは既存のタクソノミの編集や新規タクソノミの作成を行うことができます。

- *既存のタクソノミ* : 既存のタクソノミには2種類あります: (i) 編集することのできない標準タクソノミと、(ii) あなた自身またはサードパーティにより作成された標準ではないタクソノミで、編集可能。
- *新規タクソノミ* : XMLSpy では新規タクソノミの作成を行うことができます。作成方法には2種類あります: (i) 何も無い状態からのタクソノミの作成と、(ii) 標準タクソノミを拡張したタクソノミ。

XBRL ビューでは上に挙げられた両方の種類のタクソノミを閲覧、編集することができます。(タクソノミの拡張のために) 標準タクソノミを作成中のタクソノミにインポートするような場合、インポートされたタクソノミを編集することはできません。インポートされたタクソノミで、編集することが許されていないものはグレーで表示されます。

タクソノミ パッケージ

XBRL タクソノミ パッケージはタクソノミのオフラインのコピーを含む圧縮されたアーカイブです。 タクソノミ パッケージは、オフラインのタクソノミのファイルの場所へのURI を再マップするカタログ XML ファイルを含んでいます。このため、オフラインのタクソノミがアプリケーションで使用できるようになります。 タクソノミ パッケージがどのように構成されビルドされるかに関するルールは、XBRL.org のタクソノミ パッケージ勧告 で説明されています。

タクソノミ パッケージをダウンロードし、XMLSpy に登録すると、XMLSpy は、(スキーマなど) パッケージのオフラインのソースを検証の際に使用することができます。パッケージの登録は、「ツール | オプション | タクソノミ パッケージ」[1486] ペインで行うことができ、手順は、このペインの説明[1486]で説明されています。

新規タクソノミの作成方法

新規タクソノミは通常既存のタクソノミの上に構築されます。新規タクソノミでは新しい要素が追加され、これら新しい要素とインポートされた要素間のリレーションシップが作成されます。以下では新規タクソノミに求められる一般的な要件と、その作成方法の概要を説明します:

1. 新規タクソノミは、他のタクソノミと区別するために、自身の名前空間内で作成されなければなりません。新規タクソノミが既存のタクソノミを拡張したものであれば、既存のタクソノミを新規タクソノミにインポートする必要があります。
2. 新規タクソノミでは、新しいコンセプト (要素) が定義されます。
3. リレーションシップファイル (またはリンクベース) には定義、Presentation、Calculation、ラベル、そして新規タクソノミへの参照リレーションシップが含まれます。
4. 新規タクソノミへのリレーションシップは、テンプレートなど無く、最初から作成する必要があります。

上の記述にあるように、ここで使用されているタクソノミという用語は、コンセプト定義ファイルやリレーションシップファイルといった複数のファイルからなるタクソノミ全体のことを指しています。タクソノミを構成する様々なファイルの詳細についてはタクソノミドキュメントファイル[739]を参照ください。

(C) 2015-2021 Altova GmbH

XBRL ビューの使用

以下に続くセクションでは、XBRL ビューの機能を使ったタクソノミの作成や編集方法について説明します。新規タクソノミの作成[741]以降のセクションでは、各セクションの最後に独自のタクソノミを作成するための例題が与えられています。各セクションにある例題は、それ以前のセクションにて構築されたものをベースにしています。リレーションシップの作成: パート 1[757]に辿りつく頃には、XBRL ビューの操作方法にも慣れ親しみ、自信を持って使用することができるようになっているでしょう。

ここで作成するタクソノミは、最終的に C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント\Altova\XMLSpy2021Examples\XBRLExamples\Nanonull フォルダー以下に格納されている XMLSpy に付属のファイル Nanonull.xsd のようになります。メインタクソノミファイルは常に .xsd ファイル拡張子を持ち、.xbrl ファイル拡張子を持つのは XBRL インスタンスファイルで、タクソノミファイルではないことご注意してください。

17.2.2 タクソノミファイルの概要

注意深くデザインされた XBRL タクソノミは、タクソノミコンセプトをタクソノミリレーションシップとは別のファイルに保管されます。このファイルのことをメインタクソノミ、またはコンセプト定義ファイルと呼びます。更に、リレーションシップにも複数の種類があるため、これらリレーションシップもその種類ごとに個別のファイルで管理されます。以下のテーブルでは、タクソノミドキュメントを一般的に構成するファイルの種類を示します。

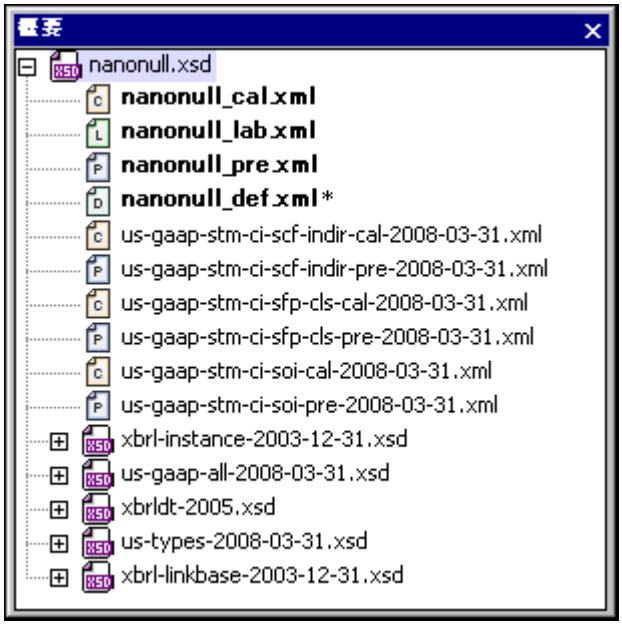
XBRL ファイル	説明	ファイルの種類
コンセプト	各コンセプトは XML スキーマ element 要素内で定義されます。	XML スキーマファイル(.xsd) コンセプト定義ファイル
定義 リレーションシップ	definitionLink 要素には コンセプト リレーションシップの全てのロケータと 定義 アークが含まれます。	XML ファイル(.xml)
計算 リレーションシップ	calculationLink 要素には全てのロケータと 計算アークが含まれます。	XML ファイル(.xml)
プレゼンテーション リレーションシップ	presentationLink 要素には全てのロケータと プレゼンテーション アークが含まれます。	XML ファイル(.xml)
ラベル	labelLink 要素には全てのロケータと ラベル アーク、そしてラベルが含まれます。	XML ファイル(.xml)
参照	referenceLink 要素には全てのロケータと、参照アーク、そして参照リソースが含まれます。	XML ファイル(.xml)

リレーションシップファイルの場所はコンセプト定義ファイル(.xsd ファイル) 内の schema/annotation/appinfo 要素により定義されます:

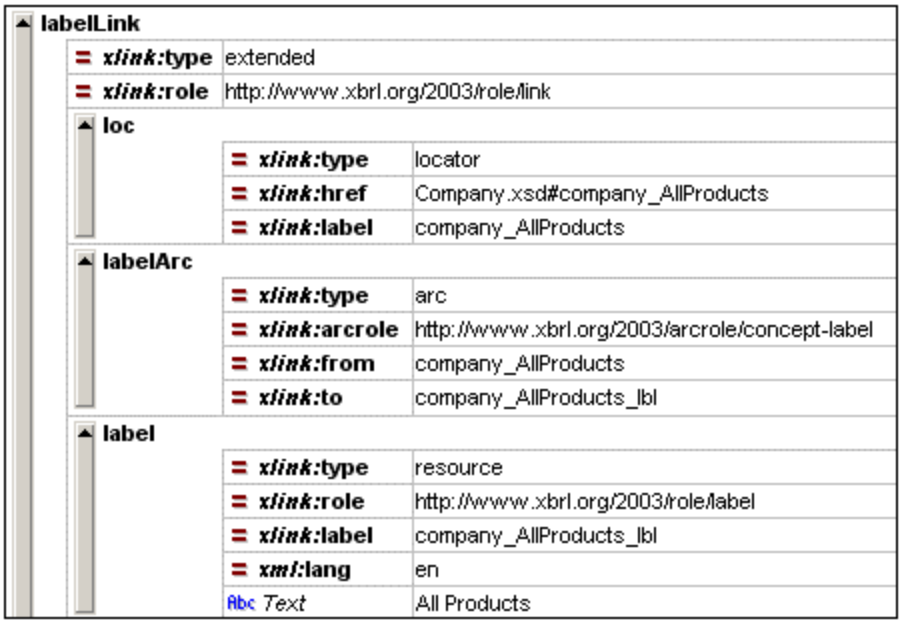
```
<xsd:annotation>
   <xsd:appinfo>
      <link:linkbaseRef xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
         xlink:href="NanonullLabels.xml" xlink:type="simple"
         xlink:role="http://www.xbrl.org/2003/role/labelLinkbaseRef" />
      <link:linkbaseRef xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
         xlink:href="NanonullDefinitions.xml" xlink:type="simple"
         xlink:role="http://www.xbrl.org/2003/role/definitionLinkbaseRef" />
      <link:linkbaseRef xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
         xlink:href="NanonullPresentations.xml" xlink:type="simple"
         xlink:role="http://www.xbrl.org/2003/role/presentationLinkbaseRef" />
      <link:linkbaseRef xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
```

```
         xlink:href="NanonullCalculations.xml" xlink:type="simple"
         xlink:role="http://www.xbrl.org/2003/role/calculationLinkbaseRef" />
      <link:linkbaseRef xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase"
         xlink:href="NanonullReferences.xml" xlink:type="simple"
         xlink:role="http://www.xbrl.org/2003/role/referenceLinkbaseRef" />

   </xsd:appinfo>
</xsd:annotation>
```

コンセプト定義ファイル(.xsd ファイル)がXBRL ビューにて開かれると、概要入力ヘルパー[250]にて様々なタクソノミファイルがツリー構造で表示されます(以下のスクリーンショットを参照)。



上のスクリーンショットにおいて、ファイル名の左にあるアイコンがスキーマを表していることにご注目してください。XML スキーマ(.xsd) ファイルは XSD アイコンにより示されます。リレーションシップファイルは、リレーションシップの種類を表す文字が記された色付のアイコンで表示されます。例えば、D アイコンは 定義 リレーションシップファイルであることを表し、P アイコンは プレゼンテーション リレーションシップファイルであることを表しています。これらファイルをダブルクリックすると、ファイルがXMLSpy のグリッドビューで開かれます(以下のスクリーンショットを参照してください)。



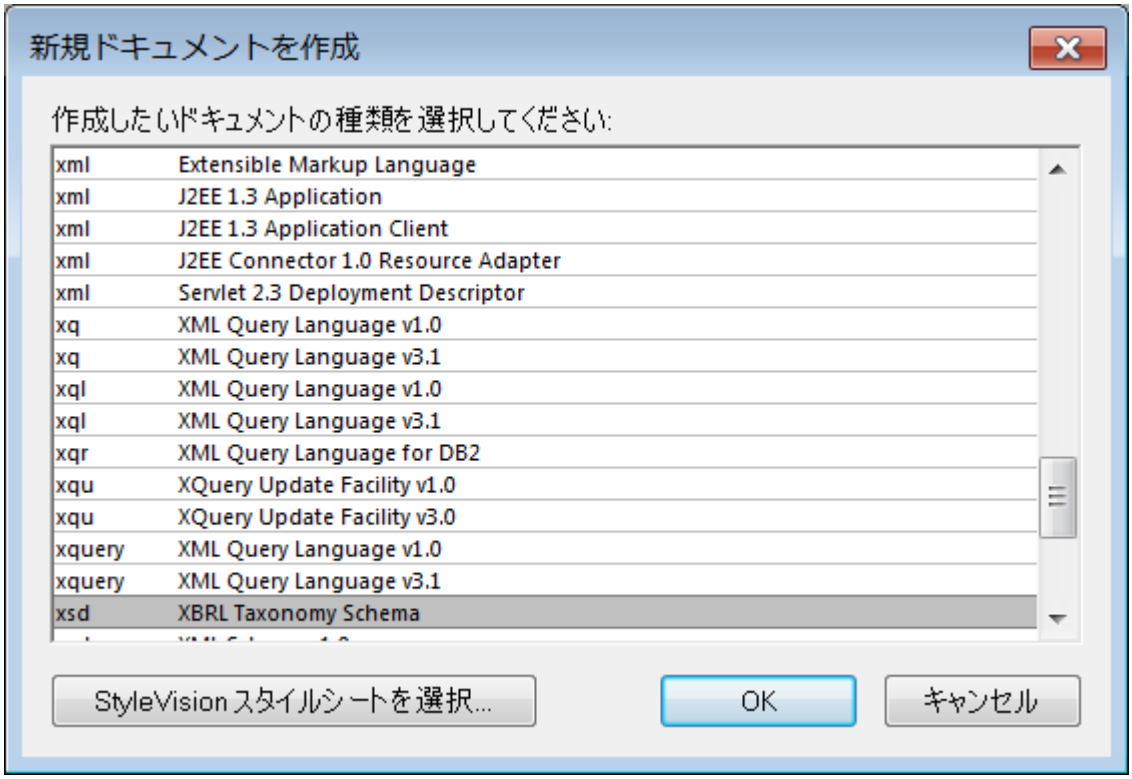
17.2.3 新規タクソノミの作成

通常新規タクソノミは1つまたは複数の標準タクソノミを拡張する形で作成されます。標準タクソノミ、または既存のタクソノミを元に新規タクソノミを構築するのであれば、既存のタクソノミをインポートする必要があります。新規タクソノミは何も無い状態から作成することもできます。XMLSpy のXBRL ビューでは、新規タクソノミウィザードを使ってUS-GAAP やIFRS タクソノミを簡単にインポートすることができます。タクソノミはXBRL ビューのグラフィカルなインターフェースを使って編集することができます。

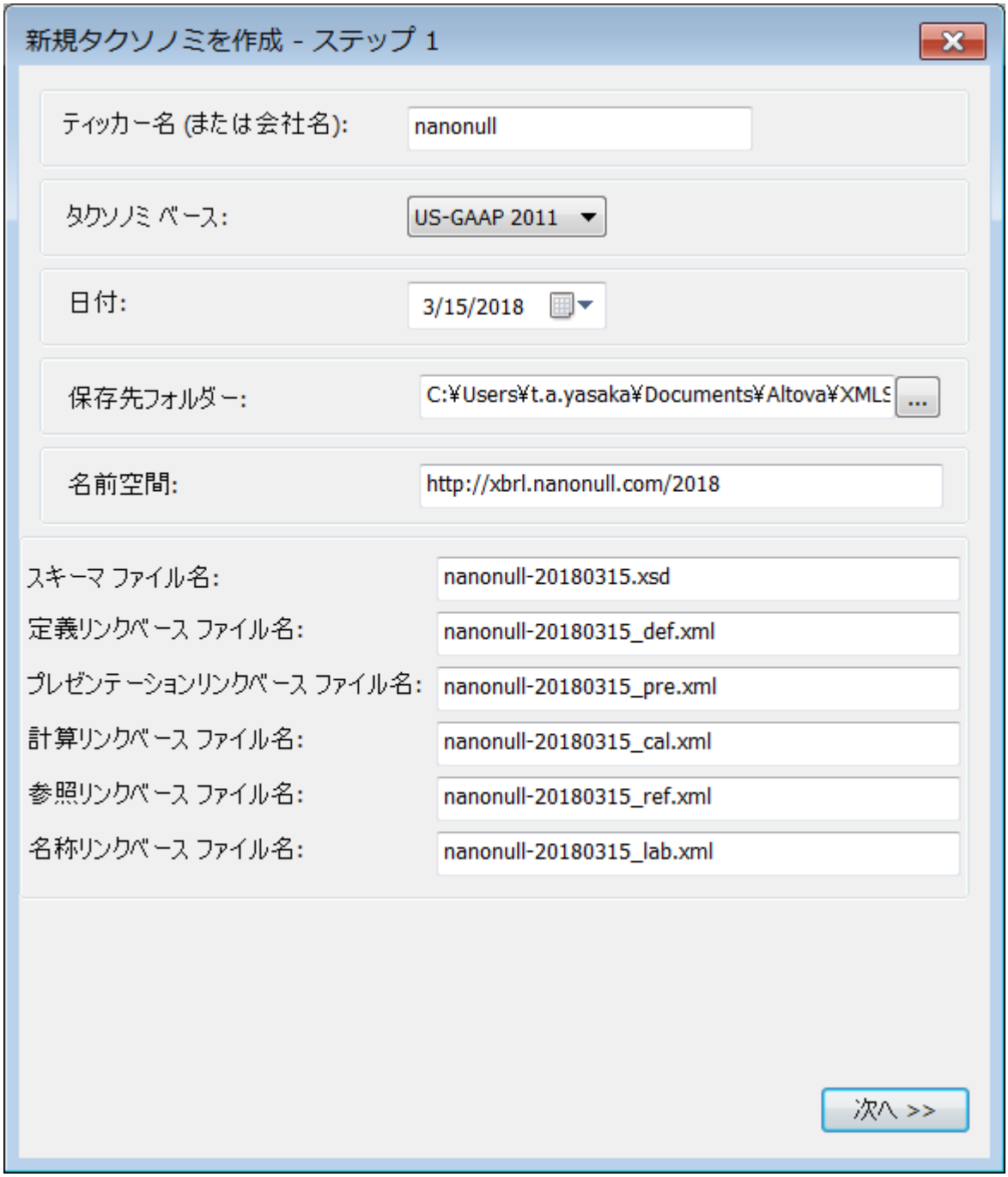
新規タクソノミを作成するにあたり最初にすることは、XML スキーマファイル（.xsd）であるコンセプト定義ファイルを作成することです。コンセプト定義の他にも、このファイルには新規タクソノミの名前空間定義、宣言が含まれており、インポートするタクソノミやタクソノミのリレーションシップファイルの場所を特定し、インポートされたタクソノミの名前空間や、その他使用される名前空間の宣言を行ないます。

コンセプト定義ファイルの作成

新規 XBRL タクソノミを作成するには、メニューコマンドから「**ファイル | 新規作成**」を選択します。新規ドキュメントを作成ウィンドウが表示されます。



.xsd XBRL Taxonomy Schema を選択し、「OK」で確定します。この操作により、タクソノミウィザードの最初の画面が表示されます。



新規タクソノミを作成ウィザードによりUS-GAAP やIFRS 要件に準拠したXBRL タクソノミを作成することができます。実質的には、指定したタクソノミがインポートされ、その後、必要に応じてファイルの編集を行うことができます。

ウィザードの最初のダイアログでは、以下の入力を行ないます:

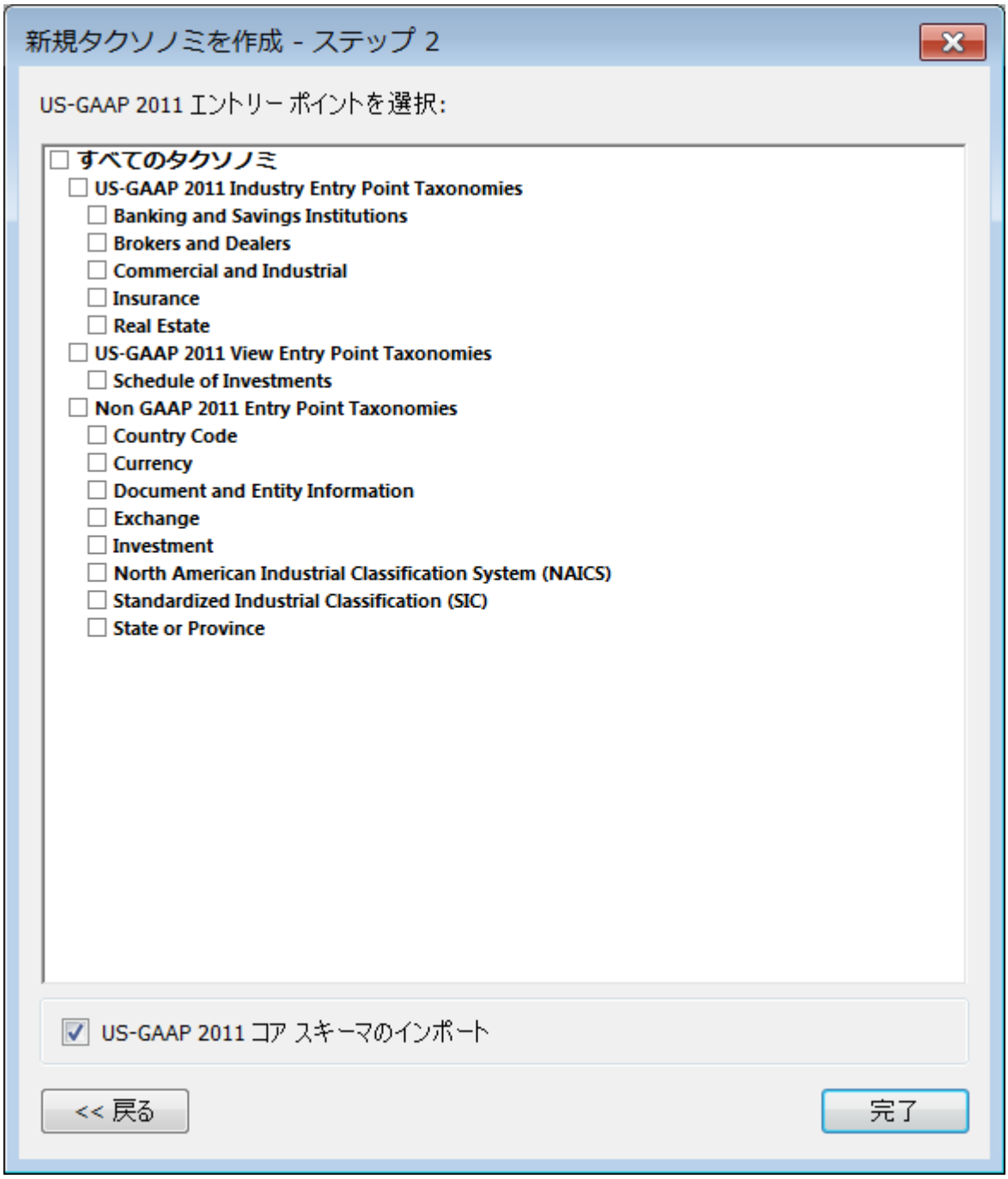
- タクソノミに使用する会社のティッカーシンボル(または名前)を作成します。入力する会社名は通常ユニークなもので、ここで入力されるティッカーシンボルをベースに重要なタクソノミプロパティやコンポーネントの名前が作成されます。例えば、*ティッカー名* フィールドに入力された値はタクソノミ名前空間やタクソノミのルーションシップファイルを自動生成する際に使用されます(上のスクリーンショッ

トを参照）。会社がティッカーシンボルを持っていない場合、このフィールドには*タクソノミ*が対象とする会社の名前を入力することができます。

- *タクソノミベース*で指定できるオプションは: *US-GAAP タクソノミ*、*IFRS*、そして*無し* です。このコンボボックスにて選択した*タクソノミ*により、あなたの*タクソノミ*が形づくられます。次に表示されるウィザードのダイアログの内容は、ここで選択した*タクソノミベース*により変わってきます。*タクソノミベース*については、次に表示されるウィザードの画面で説明されます。
- 日付の値は直接入力するか、コンボボックスのドロップダウンからアクセスすることのできる日付選択画面から選択することができます。ティッカーシンボルと同様、日付も*タクソノミ*名前空間や*タクソノミ*のリレーションシップファイルの生成時に使用されます。
- 保存先フォルダーは*メインタクソノミ*ならびに関連するファイルが保存される場所になります。
- *タクソノミ*名前空間（.xsd 拡張子を持つ*メインタクソノミ*コンセプト ファイルのターゲット名前空間）とコンセプト 定義ファイル（スキーマファイル）、そして*リレーションシップファイル*は、(i) ティッカーシンボル（または会社名）と、(ii) 日付から自動的に作成されます。これらの値を変更しようとすると、関連する*タクソノミ*に準拠しなくなるという旨の警告が表示されます。

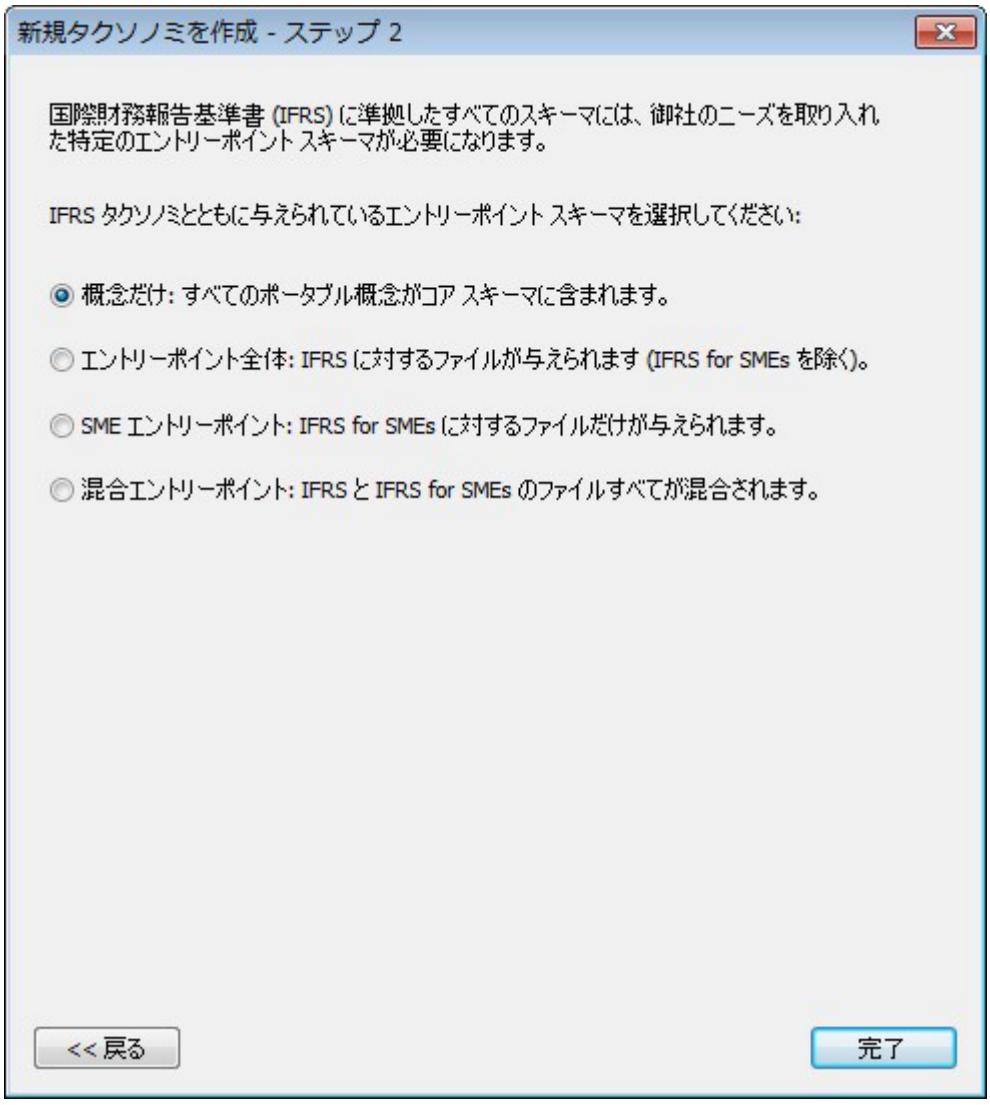
入力を完了後、**次へ**をクリックします。次に表示されるダイアログは、*タクソノミ*のベースに選択した内容により変化します。3つのケースが考えられます:

- US-GAAP *タクソノミ*を選択した場合、（以下のスクリーンショットにあるように）新規*タクソノミ*を作成ウィザードのダイアログにはUS-GAAP に特化した内容が表示されます。*タクソノミ*へインクルードしたい*エントリポイント*を選択し、（ダイアログの下部にあるチェックボックスにて）US-GAAP コアスキーマをインポートするかどうか指定することができます。



完了ボタンをクリックすると、選択したエントリポイントをインポートした新規コンセプト定義ファイル(.xsd) が、ウィザードにて指定した保存先フォルダーに作成されます。XBRL ビューで開かれるタクソノミを編集することもできます。

- タクソノミのベースとしてIFRS が選択された場合、(以下のスクリーンショットにあるように) 新規タクソノミを作成ウィザードにはIFRS に特化した内容が表示されます。目的のエントリー ポイント スキーマ オプションを選択し、「**完了**」をクリックしてください。



完了ボタンをクリックすると、選択したエントリポイントをインポートした新規コンセプト定義ファイル(.xsd)が、ウィザードにて指定した保存先フォルダーに作成されます。XBRL ビューで開かれるタクソノミを編集することもできます。

- 無し(つまりベースとするタクソノミが無い)を選択すると、新規コンセプト定義ファイル(.xsd)が、ウィザードで指定した保存先フォルダーに作成され、XBRL ビューにて開かれます。2つの要素(xbrldt:hypercube と xbrldt:dimension)が自動的に生成され、全てのタクソノミに必要となる基本的なスキーマが既にインポートされていることにご注目してください(概要入力ヘルパー[250]を参照ください)。

ベースタクソノミの作成後、次のステップは: (i) タクソノミの名前空間(スキーマのターゲット名前空間)の作成[747]と、(ii) 新規タクソノミを構築する土台となるタクソノミのインポート[748]となります。新規タクソノミウィザードを使用したのであれば、デフォルトのタクソノミ名前空間が作成され、関連するタクソノミがインポートされます。

例題: ステップ1

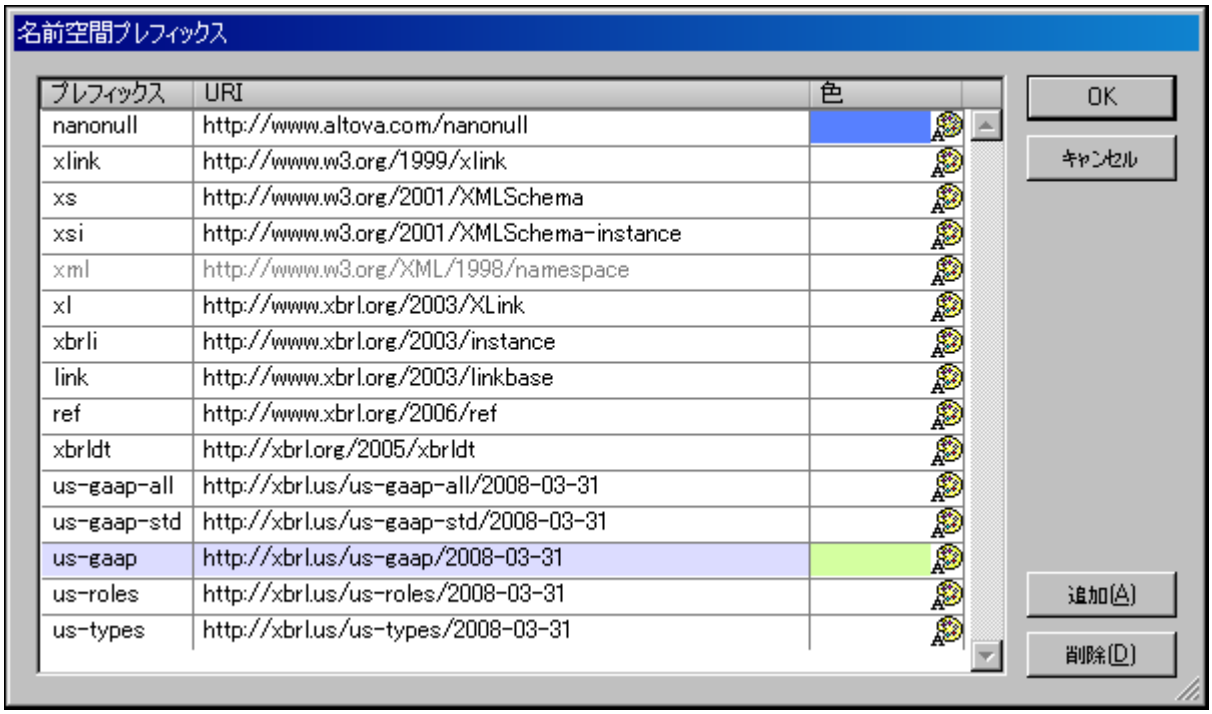
上の記述にある様に、新規タクソノミドキュメントを作成ウィザードを使って、新規タクソノミドキュメントを作成します(US-GAAP 2011 をベースタクソノミとして使用し、US-GAAP 2011 コアスキーマをインポート)。名前を入力し、適切な場所へ保存します。このファイルがメインタクソノミファイル、またはコンセプト定義ファイルとなり、XML スキーマファイルであるため、.xsd 拡張子を持つことになります。この例では作成するファイルを Nanonull.xsd とします。これは C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント

\Altova\XMLSpy2021Examples\XBRLExamples\Nanonull にある例と同じ名前で、リレーションシップファイルの名前をシンプルにするために、名前の日付部分を省略しています。

次のステップでは、新規タクソノミウィザードにより自動的に作成されたタクソノミのターゲット名前空間を修正[747]します。

17.2.4 タクソノミ内の名前空間

タクソノミ名前空間は、メニューコマンド「XBRL | 名前空間プレフィックス」からアクセスすることができる(以下のスクリーンショットの)名前空間ダイアログにより管理することができます。このダイアログでは名前空間ならびにプレフィックスの宣言や、各名前空間に対応した背景色を設定することができます。このダイアログでなされた変更により、タクソノミの名前空間宣言が修正されます。



名前空間プレフィックスダイアログにはタクソノミ内にある全ての名前空間が表示されます。

- 名前空間の追加や削除を行うには、それぞれ「追加」または「削除」ボタンを使用します。名前空間を追加した後は、対応するフィールドをダブルクリックして値を入力することで、デフォルトのプレフィックスとデフォルトのURI を修正します。
- 名前空間のカラーパレットにより、名前空間に色を割り当てることができます。色が名前空間に割り当てられると、その名前空間のコンポーネントは全て、XBRL ビューのメインウインドウならびに入力ヘルパーにて、その色を背景に表示されます。特定の名前空間に割り当てられた色の設定は、タクソノミドキュメントを超えて、XBRL ビューで開かれているドキュメント全ての名前空間に適用されることにご注意してください。

名前空間ダイアログでの編集を終了後、「OK」ボタンをクリックして変更を確定してください。

ターゲット名前空間

タクソノミのターゲット名前空間は、タクソノミの xs:schema 要素にある xs:targetNamespace 属性により定義されます(以下のコードを参照)。xs:schema 要素はコンセプト定義ファイルのドキュメント要素です。

```
<xs:schema targetNamespace="http://www.altova.com/XBRL/Taxonomies">
```

```
        ...
    </xs:schema>
```

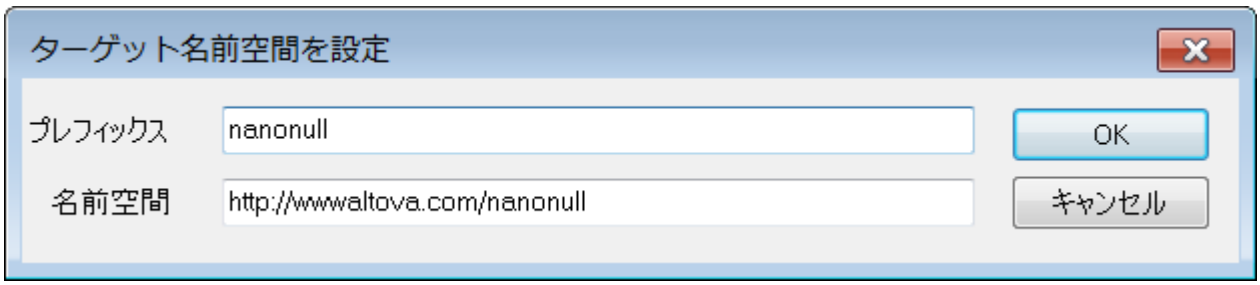
ターゲット名前空間を定義(指定)するのに加えて、ターゲット名前空間は xs:schema 要素にて宣言されなければならず、これによりドキュメント全体をスコープに収めることができます。以下のコードはターゲット名前空間である名前空間の宣言を行っています。

```
    <xs:schema targetNamespace="http://www.altova.com/XBRL/Taxonomies"
xmlns:ns1="http://www.altova.com/XBRL/Taxonomies" >
        ...
    </xs:schema>
```

上のコードでは、名前空間は xs:schema 要素にて宣言され、ns1 というプレフィックスが与えられています。

ターゲット名前空間の設定

新規タクソノミウィザードにより新しいタクソノミが作成されると、デフォルトのターゲット名前空間とプレフィックスが、タクソノミに対して自動的に作成されます。デフォルトのターゲット名前空間は、新規タクソノミウィザードで入力された値をベースに作成されます。デフォルトで作成されるターゲット名前空間のプレフィックスは nX となります(X は整数)。デフォルトターゲット名前空間の宣言とプレフィックスは、(「XBRL | ターゲット名前空間の設定」コマンドによりアクセスできる)ターゲット名前空間ダイアログを使って編集することができます(以下のスクリーンショットを参照)。これらの値を編集することにより、ターゲット名前空間の定義(targetNamespace 属性の値)だけでなく、ターゲット名前空間の宣言も変更されます。



(ターゲット名前空間の定義を除いて)ターゲット名前空間の宣言だけを修正、または任意の名前空間の宣言だけを修正するには、名前空間プレフィックスダイアログ(「XBRL | 名前空間プレフィックス」コマンド)から、名前空間と、そのプレフィックスの値を編集してください。

例題: ステップ2

「XBRL | ターゲット名前空間の設定」コマンドを使用して、ターゲット名前空間を設定ダイアログを開きます。デフォルトのターゲット名前空間は http://xbrl.nanonull.com/2011 で、プレフィックスは n1 となっています。名前空間、そしてプレフィックスのフィールドをダブルクリックして、値の修正を行ないます。(上のスクリーンショットにあるように)名前空間には http://www.altova.com/nanonull を、そしてプレフィックスには nanonull を指定します。ダイアログにある「OK」ボタンをクリックすると、ターゲット名前空間が定義され、割り当てたプレフィックスと共に宣言されます。この例では、ターゲット名前空間が http://www.altova.com/nanonull で、プレフィックスが nanonull となります。

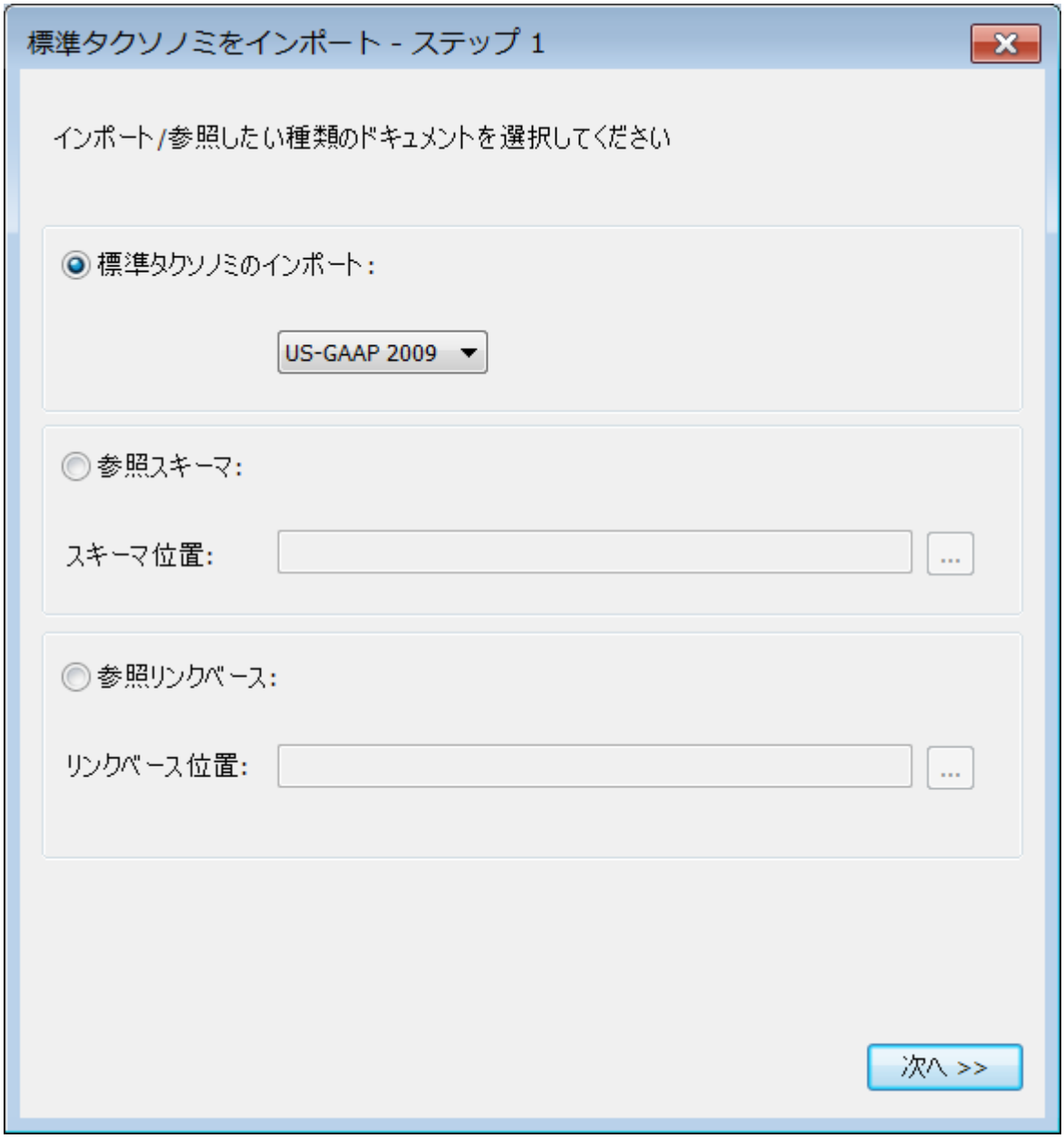
次のステップでは、XBRL ビューのインポートメカニズム[748]について詳しく見ていくことにします。

17.2.5 タクソノミのインポート

新規タクソノミが、既存のタクソノミの上に構築される場合、拡張したタクソノミに既存のタクソノミをインポートする必要があります。タクソノミを新規タクソノミウィザードを使って作成する場合、US-GAAP や IFRS をベースにしたタクソノミを、作成時にインポートすることができます。タクソノミのインポートはその後いつでも行うことができます。

以下の操作によりタクソノミをインポートすることができます:

1. XBRL ビューの概要入力ヘルパーを右クリックして、メニューコマンドから「**インポート/参照**」を選択します。
2. 標準スキーマのインポート ダイアログが表示され、インポートするタクソノミや参照するリンクベースを指定することができます（以下のスクリーンショットを参照）。選択されたオプションにより、ダイアログの名前が変化します。



インポート/参照には、(i) 標準タクソノミ（US-GAAP または IFRS）、(ii) その他のタクソノミ（参照スキーマ）、そして (iii) リンクベースという 3 つのオプションがあります。非標準タクソノミをインストールしている場合、参照スキーマ ラジオ ボタンを選択し、スキーマ位置のテキスト フィールドにある「**参照**」ボタンをクリックして、目的のタクソノミを選択してください。

3. 選択を行った後に「**完了**」ボタンをクリックします。選択されたタクソノミがインポートされ、その要素とリレーションシップが XBRL ビューに表示されます。

以下の点に注意してください:

- 概要入力ヘルパーには、インポートされたタクソノミがインポートするタクソノミと、インポートされたタクソノミが使用するリンクベースが表示されます。
- グローバル要素入力ヘルパーでは、インポートされたタクソノミ内のコンセプトが表示されます。
- デザインウィンドウならびに詳細入力ヘルパーでは、インポートされたコンセプトはグレーで表されます。
- インポートされたタクソノミを削除するには、概要入力ヘルパーにてタクソノミを右クリックし、「削除」を選択します。

タクソノミをインポートした後は、必要に応じてタクソノミを拡張することができます。

メモ: US-GAAP の様な巨大なタクソノミにより編集のスピードが落ちてしまうと感じるのであれば、メインウィンドウのフィルタリング[243]を使って、表示対象を新規タクソノミに制限することができます。この操作によりスピードが大幅に向上します。

インポートメカニズム

上で述べたような標準的なインポートの追加により、新規タクソノミファイルに `xs:import` 要素が追加されます。`xs:import` 要素によりインポートされたタクソノミの名前空間と場所が指定されます。

```
<xs:import namespace="http://fasb.org/us-gaap/2011-01-31"
           schemaLocation="http://xbrl.fasb.org/us-gaap/2011/elts/us-gaap-2011-01-31.xsd"/>
```

上の設定では、`schemaLocation` 属性によりタクソノミがインターネット上からロードされるよう指定されています。しかしこのURI は XMLSpy のカタログメカニズム[409] により、(XMLSpy パッケージに同封される)US-GAAP タクソノミのローカルにコピーへマップされます。

ローカルに保存されているタクソノミの位置を確認するには、ローカルのアドレスを直接使用することができます。その他にも、カタログファイル[409]を使ってウェブアドレスからローカルのアドレスを発見することができます。

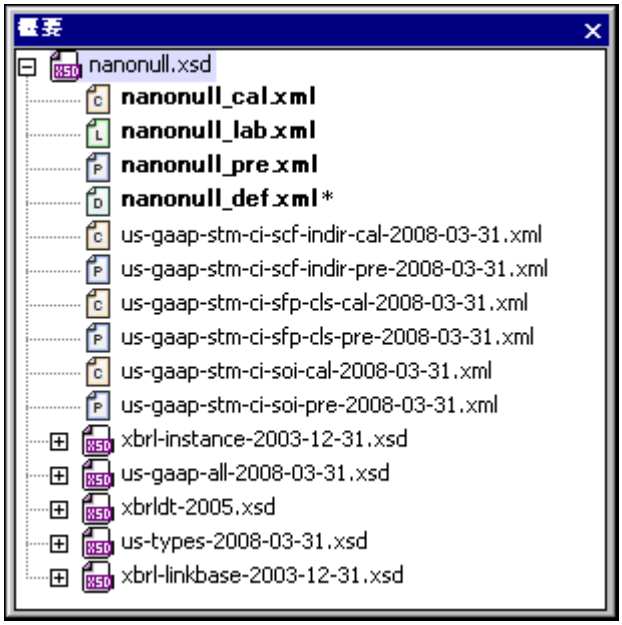
例題: ステップ3

この例題で作成しているタクソノミはUS-GAAP 2011 タクソノミを使用しています。このタクソノミは、タクソノミの作成時[741]に既にインポートされています。概要入力ヘルパーにて、インポートされたタクソノミと参照されているリンクベースを良く見てください。テキストビューへ切り替え、`xs:import` 要素を探してみてください。XBRL ビューのメインウィンドウでは、インポートされたコンセプトがグレーのフォントカラーで表示されていることに注目してください。また、概要入力ヘルパーにて、US-GAAP タクソノミのリンクベースとインポートされたスキーマが表示されていることにも注目してください。

次のステップ[750]では、リンクベースファイルと、参照メカニズムに関する解説を行ないます。

17.2.6 タクソノミファイルの設定

概要入力ヘルパーには、タクソノミを構成するファイルが、以下のスクリーンショットで示されるように、ツリー構造で表示されます。ツリーのルートには、メインタクソノミファイル(コンセプト定義ファイル)があり、これが現在アクティブなファイルになります。その下のレベルにあるファイルには2種類あります: (i) タクソノミ内の様々なリレーションシップを指定するリンクベースファイル(これらは色付のアイコン[250]で示されます)と、(ii) インポートされたスキーマ(`.xsd` ファイル)です。



タクソノミのインポート[748]セクションでは、概要入力ヘルパーからタクソノミをインポートする方法が記述されました。インポートされたタクソノミは、インポートされたスキーマと共に、概要入力ヘルパーに表示されます。

このセクションでは、概要入力ヘルパーを使ってリンクベースファイルを管理する方法について説明します。概要入力ヘルパーのコンテキストメニューからは、リンクベースを管理するための、4つのオペレーションにアクセスすることができます:

- 新規リンクベースの追加[751]とタクソノミと共にそれらを保存[751]。
- リンクベースの種類の設定[752]。リンクベースの種類(計算、定義、表示、名称、そして参照)をXMLSpyが特定できなかった場合、リンクベースの種類を明示的に指定することができます。
- 特定の種類のリレーションシップリンクベースに対して、任意のリンクベースをデフォルトリンクベース[752]として設定します。特定の種類のリンクベース、例えばラベルリレーションシップに対して、2つ以上のリンクベースがある場合、タクソノミ内で新たに作成されるラベルは、デフォルトラベルリンクベースで作成されます。
- リンクベースの削除[752]

メモ: リレーションシップには5つの種類があります: (i) 定義、(ii) 計算、(iii) プレゼンテーション、(iv) ラベル、(v) 参照。各リレーションシップの種類に対して別々のリンクベースファイルを作成することができます。

新規リンクベースの追加

新規リンクベースを追加するには:

1. 概要入力ヘルパーを右クリックして、「**新規リンクベースの追加 | <リレーションシップの種類>**」を選択します。選択されたリレーションシップの種類のリンクベースファイルが、デフォルトの名前で概要入力ヘルパーに作成されます。作成された新規リンクベースは、そのリレーションシップの種類のデフォルトリンクベースになることに注意してください(ファイル名が太字で表示されます)。
2. デフォルト名を右クリックして、「**名前の変更**」を選択し、名前を修正します。
3. 新たに作成されたリンクベースファイル名は、メインタクソノミファイルが保存される際に、物理的に(ファイルシステムで)保存されます。詳細については以下を参照してください。

リンクベースファイルの保存

リンクベースファイルが保存されていない場合、リンクベースファイルの名前の右側にアスタリスク(*)が表示されます。メインタクソノミファイルの保存は、以下のように行われます:

1. リンクベースのパスを確認するダイアログが表示されます。このダイアログでは、新たに追加されたものも含めて、タクソノミ内にある全リンクベースの名前と場所が表示されます。保存されていないリンクベースファイルのパスには、メインタクソノミファイルが保存されているフォルダーへのパスが与えられます。リンクベースを別の場所に保存したい場合は、各リンクベースファイルのパスを編集することができます。ファイルの名前も変更することができます。
2. 完了後、「OK」をクリックします。指定された場所にリンクベースファイルが保存されます。

リンクベースの種類のセット

このコマンドを使うと、ファイルのリンクベースの種類をセットすることができます。リンクベースの種類を変更したいファイルを右クリックして、コンテキストメニューから「**リンクベースの種類を設定** | <**リンクベースの種類**>」を選択します。「**すべて**」オプションにより、リンクベースファイルには2つ以上のリレーションシップが含まれていると指定することができます。

デフォルトリンクベースのセット

各リレーションシップの種類に対して、デフォルトリンクベースファイルをセットすることができます。その種類のリレーションシップがタクソノミエディターにてセットされると、リレーションシップがその種類のデフォルトリンクベースファイルに保存されます。リンクベースファイルをデフォルトリンクベースとしてセットするには、リレーションシップを右クリックして、「**デフォルトリンクベースの設定** | <**リンクベースの種類**>」を選択します。デフォルトリンクベースの名前は太字で表示されます。

リンクベースの削除

リンクベースを右クリックし、**削除**を選択することで、タクソノミからそのリンクベースを削除することができます。

例題: ステップ4

作成中のタクソノミファイルには、既にリンクベースファイルが含まれています。これらのリンクベースファイルは新規タクソノミの作成ウィザード[741]により作成されたものです。サンプルファイルでは、デフォルトで与えられている名前が以下のように変更されています:

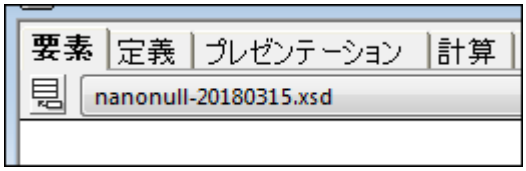
- 計算リンクベース: `nanonull_cal.xml`
- 定義リンクベース: `nanonull_def.xml`
- ラベルリンクベース: `nanonull_lab.xml`
- プレゼンテーションリンクベース: `nanonull_pre.xml`

このセクションで解説されているコマンドを試してみるには、「**新しいリンクベースの追加**」コマンドを使って、お好きな種類のリンクベースファイルを作成します。上記にあるように、名前の変更をします。新たに作成されたリンクベースが、その種類のデフォルトリンクベースとなっていることに注意してください(名前が太字で表示されます)。そのリンクベースを選択し、「**リンクベースの種類を設定**」コマンドを使って、他のリレーションシップの種類に変更します。ファイルが新たなリンクベースの種類のデフォルトリンクベースにはならないことに注目してください。「**削除**」コマンドにより、リンクベースを削除します。オリジナルのリンクベースファイルは既にデフォルトリンクベースではないので、そのファイルをデフォルトリンクベースに設定しなおします。

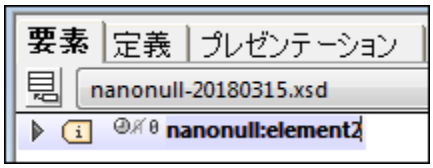
次のステップでは、メインタクソノミファイルへの新たな要素の追加[752]について説明します。

17.2.7 タクソノミへ要素の追加

タクソノミへ要素(コンセプト)を追加するには、メインウィンドウにある「**新しい要素の追加**」アイコンをクリックします(スクリーンショットを参照)。

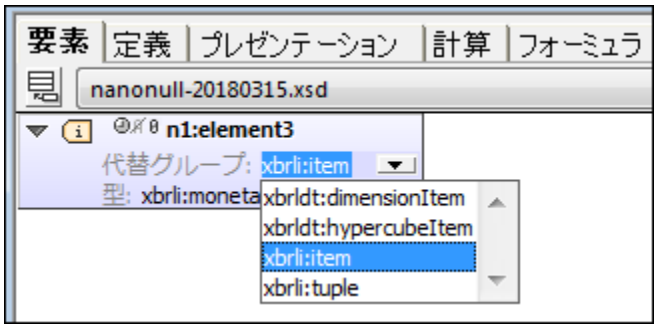


xbrli:item の代替グループを持つ新規要素がデフォルトの名前で、画面上の要素リストに追加されます(以下のスクリーンショットを参照)。



要素ボックスについては、メインウィンドウ| 要素タブ[243]のセクションを参照ください。以下の方法により、メインウィンドウにて要素のプロパティを編集することができます。

- 要素のデフォルト名をダブルクリックすることで、要素の名前を変更することができます。名前には適切な名前空間プレフィックス[747]も入力する必要があるということに注意してください。
- プラス(+)アイコンをクリックして要素ボックスを展開し、(以下のスクリーンショットにあるように)フィールドのドロップダウンリストから目的の代替グループを選択することで、要素の代替グループを変更することができます。



- Balance、Period、Abstract、またはNillable プロパティを変更するには、要素名の左に配置されている対応するアイコンをクリックし、そこで表示されるボックスに表示されるオプションを選択します。
- 要素にラベルリンクロールを追加するには、要素ボックスを右クリックし、「ラベルリンクロールの追加」コマンドを選択します。ラベルリンクロールの行が追加され、この行にラベルリンクロールを入力し、コンボボックスからオプションを選択することができます。タクソノミにラベルリンクロールが割り当てられていない場合、ここでラベルが作成され、概要入力ヘルパー[250]にて表示されます。
- ラベルリンクロールを右クリックして、「ラベルの追加」コマンドを選択することで、ラベルリンクロールにラベルを追加することができます。ラベルの詳細を入力するには、フィールドをダブルクリックして新しい値を入力するか、対応するコンボボックスから新しい値を選択します。ラベルに加えられた変更は、メインタクソノミファイルの保存時にラベルリンクベースに保存されます。
- 参照はラベルと同様の方法により参照リンクベースに追加されます。参照リンクベースはまず要素に対して追加され、特定の参照リンクロールに対して追加されます。

要素プロパティは詳細入力ヘルパーで編集することができます。編集方法についてはXBRL ビューの入力ヘルパー[250]のセクションを参照ください。

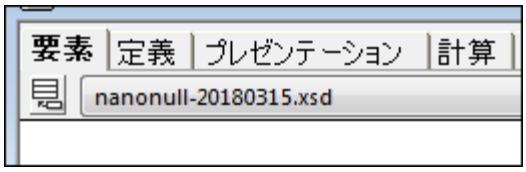
例題:ステップ5

このセクションでは、新しい要素を作成してUS-GAAP タクソノミの拡張を行ないます1。

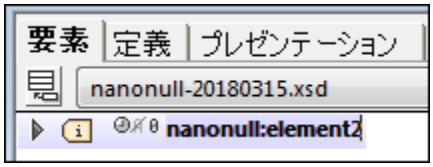
最初に作成する要素は nanonull:OnboardAndOther で、Nanonull 社の客船での売上による歳入を表しています。これほど詳細な歳入項目はUS-GAAP タクソノミ内で定義されておらず、US-GAAP の拡張を作成する必要があります。Nanonull 社に特化したタクソノミとして、タクソノミはNanonull 名前空間（http://www.altova.com/nanonull）内に作成されなければなりません。名前空間はnaanonull という名前空間プレフィックスにより既に宣言されています。このプレフィックスを持つ要素を作成すると、要素は Nanonull 名前空間内に配置されます。

要素を作成するには:

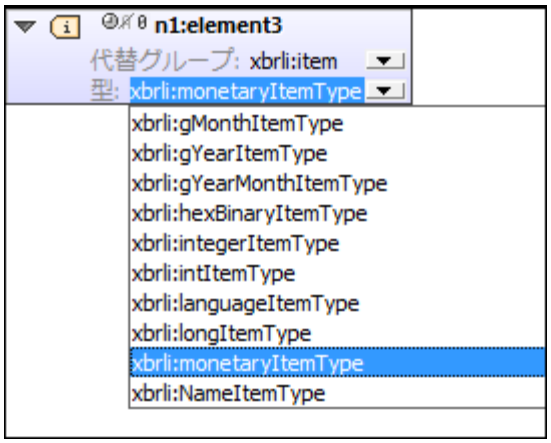
1. メインウィンドウにある、「新しい要素の追加」アイコンをクリックする



xbrli:item という代替グループと、デフォルトの名前を持つ新しい要素が、要素のリスト内に作成されます。



2. 要素名をダブルクリックして、nanonull:OnboardAndOther を入力します。この操作により Nanonull 名前空間内に OnboardAndOther 要素が作成されます。
3. 要素ボックスを展開します。要素には金額が入るので、（以下のスクリーンショットにあるように）Type 属性を xbrli:monetaryItemType に変更します。



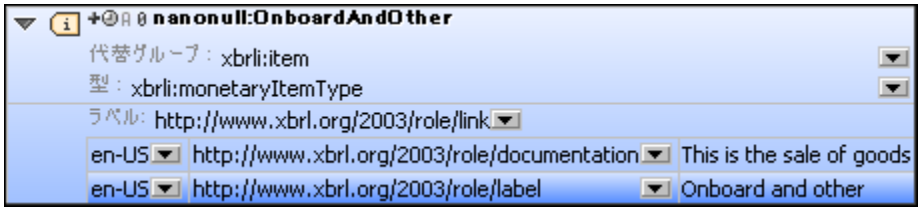
4. 時計アイコンの左の空間をクリックし、ポップアップから credit を選択します（以下のスクリーンショットを参照）。

これで xbrli:balance 属性の値が credit になりました。

5. 時計アイコン、A アイコン、そして 0 アイコンをクリックして、それぞれ *duration*、*非 Abstract*、*Nillable* にセットします。xsd ファイル内部では、xbrli:duration、xs:abstract、xs:nillable の値が、それぞれ credit、duration false、そして true となります。

6. 要素ボックスを右クリックして、表示されたメニューから「ラベルリンクロールの追加」を選択します。この操作により要素ボックスの下部にラベルリンクロールの行が追加されます。

7. XBRL リンク URI を選択します。
8. ラベルリンクロールの行を右クリックして、メニューから「ラベルの追加」を選択します。この操作により、ラベルリンクロール以下に新たにラベルの行が作成されます。
9. 新たに作成されたラベル行の(一番左にある)言語フィールドをダブルクリックして、`en-US` を入力します(以下のスクリーンショット参照)。リンクロールを指定する次のフィールドでは、ドロップダウンリストからドキュメント を選択します。ラベルフィールドにはドキュメンテーションに表示されるテキストを入力します。その後、同様の操作を `label` リンクロールに対しても行ないます。要素の表示が(左側にある矢印をクリックすることで) 展開されると、名称の右側にあるプラス/マイナス シンボルをクリックすることで、`label` ロールの表示を切り替えることができます(名称の表示を切り替え)。



これで要素 `nanonull:OnboardAndOther` が作成されました。

`OnboardAndOther` には `credit`(貸方)の `xbrli:balance` 値があることに注意してください。これはこのアイテムが収益のアイテムであるためです。当然売られる品物にも仕入れのためにコストがかかるため、対応する `nanonull:CostOfOnboardAndOther` という要素を借方側に作成します。nanonull:OnboardAndOther を作成したのと同様の方法でこの要素を作成します。この要素では、`xbrli:balance` の値を `credit` ではなく `debit` にセットしてください。

コストの中にはエージェントに対するコミッションも含めなければなりません。従って、`xbrli:balance` は debit の `nanonull:CruiseCommissionsTransportationAndOther` という要素を作成します。`nanonull:CostOfOnboardAndOther` をと全く同じような方法でこの要素を作成してください。

最後に、地域ごとのグループ化を行うために、`Asia`、`Europe`、そして `USA` という3つの要素を作成します。これら要素はグループ化の為だけに使用され、要素そのものは値を持たず、このような要素は Abstract 要素と呼ばれます。その要素がどのような型を持つかは重要ではありません。Abstract 要素には、それら要素が使用される状況に最も適した型を与えるのが良いでしょう。例えば、サンプルファイルでは、これら3つの要素に `stringItemType` 型が与えられています。今までに作成した要素と同様に、`nanonull:Asia`、`nanonull:Europe`、そして `nanonull:USA` を作成します。これら要素に対しては、Abstract 属性を *Abstract* にセットします(`.xsd` ファイル内に記述される実際の属性値は `true` になります)。xbrli:balance 属性の値はセットしません。

メモ: Abstract 要素にて `xbrli:balance` 属性許されているのは `monetaryItemType` の要素だけで、そうでない場合、タクソノミが不正となってしまいます。Abstract 要素からは、オプションの `xbrli:balance` 属性をセットしないようにした方が賢明です。

次のステップでは、新規タクソノミのリンクロールを指定 756します。リレーションシップの作成時にこれらリンクロールが必要になります。

17.2.8 リレーションシップとリンクロール

リレーションシップのセットが作成される時、これらリレーションシップは要素内にて作成されます。例えば、Definition リレーションシップが作成されるとき、これら 定義 リレーションシップを定義するもの(ロケータと 定義 アーク)はすべて、以下にあるような definitionLink 要素内にて作成されます。

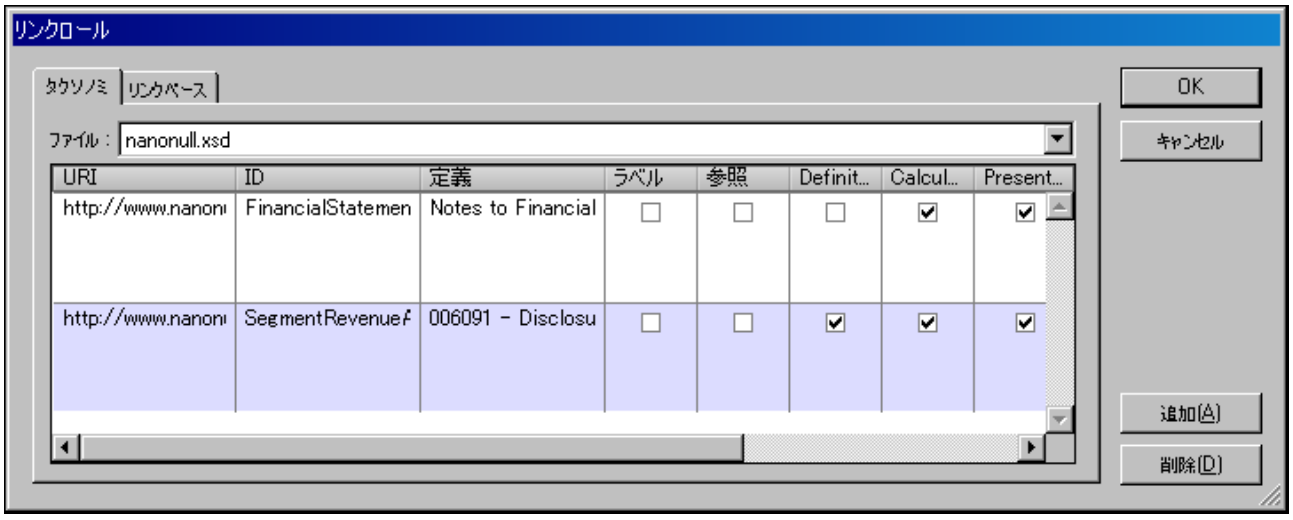
```
<link:definitionLink xlink:type="extended"
xlink:role="http://www.nanonull.com/taxonomy/role/SegmentRevenueAndOperatingIncome">
```

definitionLink にある xlink:role 属性の値は、Definition リレーションシップで使われるリンクロールの roleURI 属性の値で無ければなりません(以下のコードを参照)。以下のコードに見られるように、リンクロールはタクソノミの appinfo 要素内に含まれています。

```
<xs:appinfo>
   <link:roleType id="SegmentRevenueAndOperatingIncome"
   roleURI="http://www.nanonull.com/taxonomy/role/SegmentRevenueAndOperatingIncome">
      <link:definition>006091 - Disclosure - Segment Revenue and Operating
Income</link:definition>
      <link:usedOn>link:calculationLink</link:usedOn>
      <link:usedOn>link:definitionLink</link:usedOn>
      <link:usedOn>link:presentationLink</link:usedOn>
   </link:roleType>
</xs:appinfo>
```

リンクロールは他のリレーションシップの種類を含んだ(例えば calculationLink や presentationLink 要素といった definitionLink 要素以外の)要素にて使用されます。上のコードにおいて、このリンクロールが使用されても良いリレーションシップの種類を指定する usedOn 要素があることに注目してください。

リンクロールを作成するには、XBRL ビューにて、メニューコマンドから「XBRL | リンクロール」を選択してください。以下のスクリーンショットに示されるリンクロールダイアログが表示されます。



タクソノミ タブにおいて、ファイル コンボ ボックスのドロップダウン リストからタクソノミを選択し、「追加」をクリックしてリンクロールを追加します。リンクロールのURI とID (上のコードを参照)を指定します。このリンクロールが使用されても良いリレーションシップの種類を選択します。目的のリレーションシップの種類があるチェックボックスにチェックを入れてください。

例題: ステップ6

リンクロールダイアログ(「XBRL | リンクロール」)を使って、上のスクリーンショットにあるように2つのリンクロールを作成します:

1. id="SegmentRevenueAndOperatingIncome" URI="http://www.nanonull.com/taxonomy/role/SegmentRevenueAndOperatingIncome"（定義、計算そしてプレゼンテーション リレーションシップで利用可）。
2. id="FinancialStatements" URI="http://www.nanonull.com/taxonomy/role/FinancialStatements"（計算 ならびにプレゼンテーション リレーションシップで利用可）。

次のステップでは、新規タクソノミのリレーションシップの作成[757]について説明します。

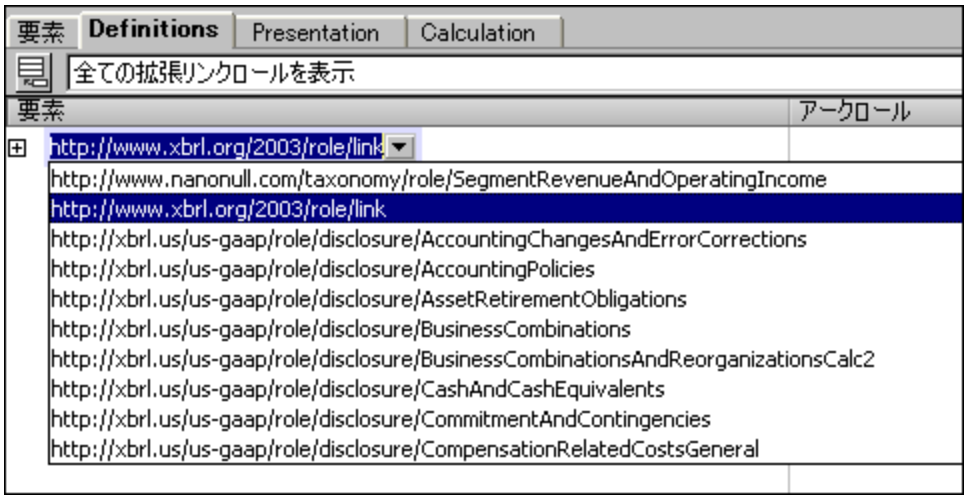
17.2.9 リレーションシップの作成：パート1

リレーションシップは、それぞれ対応する各タブにて作成されます：Definisions、Presentation、Calculation。これら3種類のリレーションシップの作成は、Definition リレーションシップがアークロールを持ち、Presentation リレーションシップと検証リレーションシップは持たないという点を除いてほとんど同じです。このセクションでは 定義 リレーションシップを例にしてリレーションシップの作成方法を説明します。次のセクション[760]では、Presentation ならびに 検証リレーションシップとの違いや、リレーションシップに関するその他の機能について解説します。

以下の説明を読み進めるにあたり、XBRL ビューにて既に完成されたドキュメントを開くことをお勧めします。Nanonull タクソノミはフォルダー C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント \Altova\XMLSpy2021Examples\XBRLExamples\Nanonull にて見つけることができます。

リンクロールの追加

メインウィンドウにて目的のタブ（Definitions、Calculation、Presentation）をクリックします。メインウィンドウを選択し、「**拡張リンクロールの追加 / リンクロールの管理**」ボタンから「**拡張リンクロールの追加**」コマンドを選択します。この操作により、（以下のスクリーンショットにあるように）デフォルト リンクロールの URI が、含まれる行が追加されます。行の右側にあるドロップダウンの矢印をクリックすると、選択可能なリンクロールのリストが表示されます。

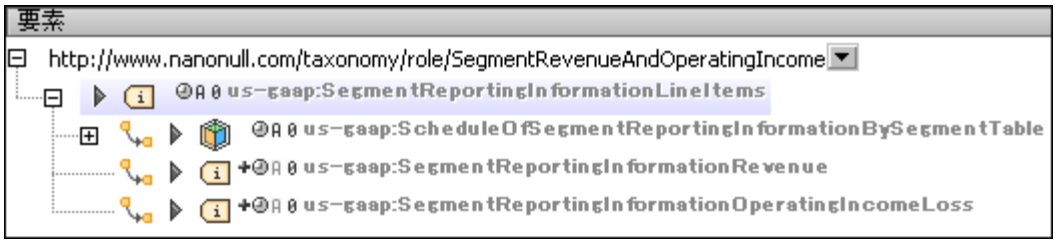


目的のリンクロールが表示されていない場合、それはタクソノミまたは現在のリレーションシップの種類にて、そのリンクロールが定義されていないためです。リンクロールとその作成方法に関する詳細はリレーションシップとリンクロール[756]のセクションを参照ください。

追加できるリンクロールの数に制限はありません。

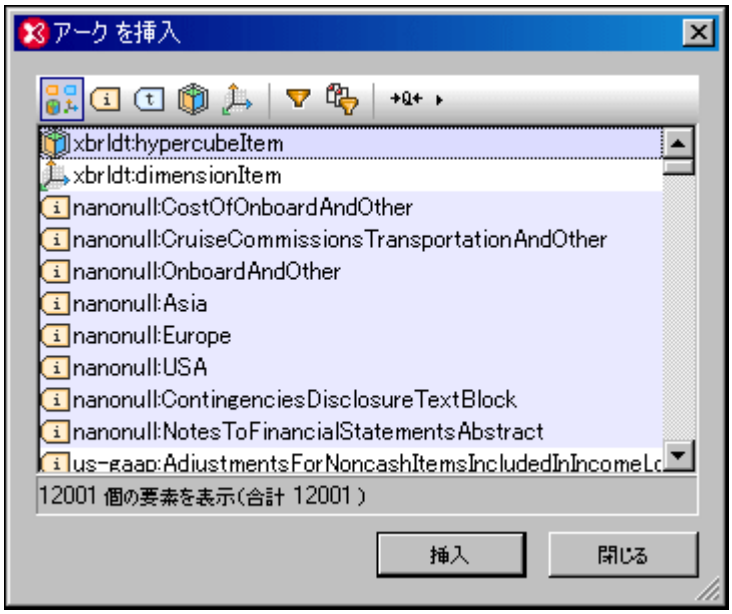
リンクロールへの要素参照やアークの挿入

リンクロール内で最初で作成される要素から、他の要素へのリレーションシップが作成されます（以下のスクリーンショットを参照）。この要素は、通常 Abstract 要素となり、（バランスシートの要素のように）他の要素をグループ化するために使用されます。この要素からアークが作成されるため、アークロール列へのエントリは存在しません。アークロールは、アークを送られる側の要素にて表示されます。

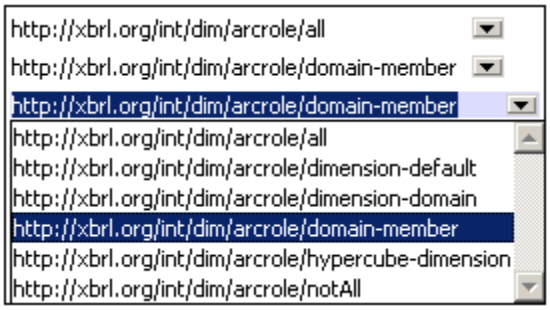


上のスクリーンショットでハイライトされている要素は、挿入された要素参照です。3つのアークがあり、1つがハイパーキューブ要素へ、2つがアイテム要素へのものです。各アークには、それぞれアークロールがあり、それらアークロールは「from-to」リレーションシップを定義し、これら3つの要素は各アークの「to」側の要素となります。

アークを要素参照または要素に挿入するには、「from」要素を右クリックし、コンテキストメニューから「アークの挿入」を選択します。この操作によりアークを挿入ダイアログが表示されます。アークの「to」側に作成する要素を選択します。このダイアログに表示されている内容を絞り込むには、フィルターの条件を指定してください（フィルタリングについてはXBRL ビュー内の入力ヘルパー 251に関する記述を参照ください）。



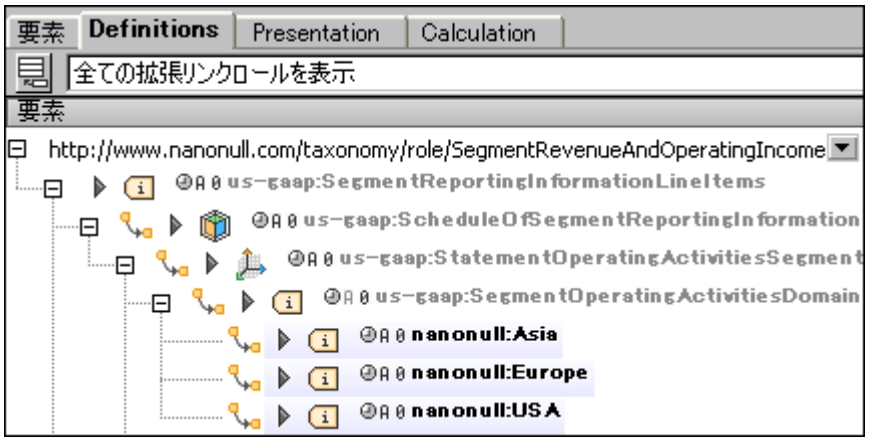
デフォルトのアークロールと共に要素が挿入されます。（以下のスクリーンショットにあるように）アークロールのドロップダウンリストから別のアークロールに変更することもできます。



メモ: 要素はアークと共にグローバル要素入力ヘルパーからドラッグすることで追加することもできます。

例題: ステップ7

以下のスクリーンショットにあるように定義リレーションシップを作成します。

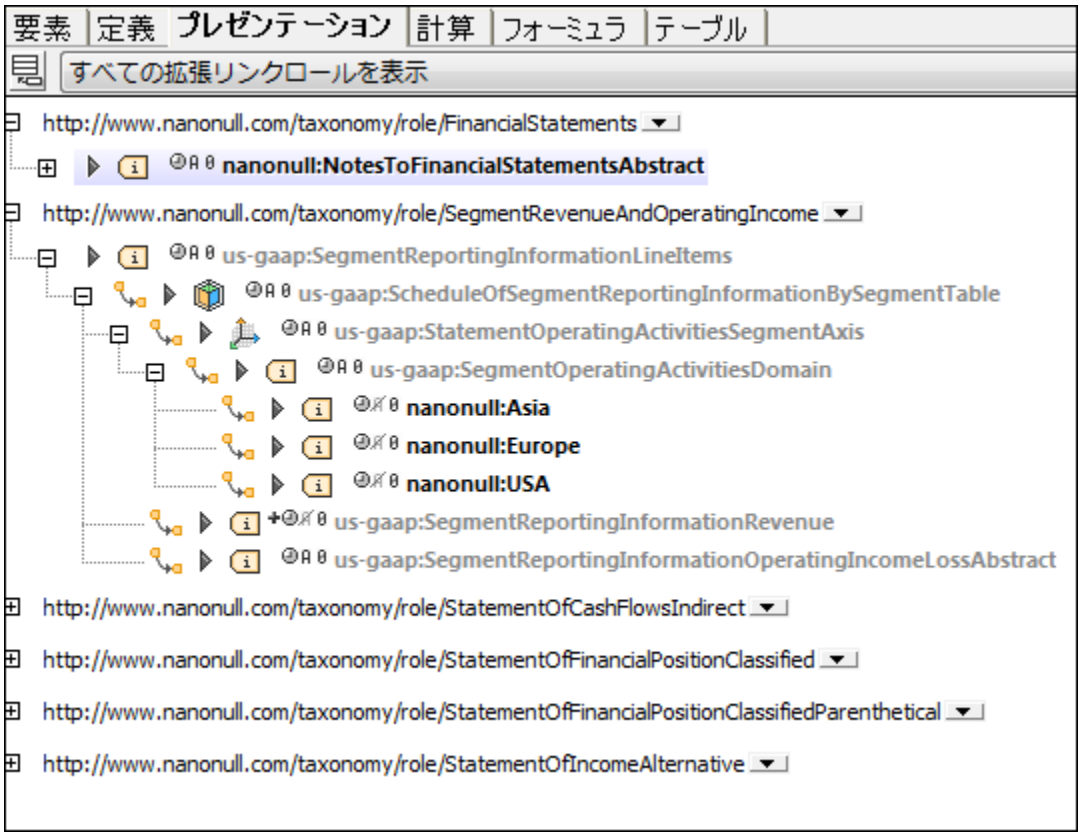


上のスクリーンショットにはアークと共に追加される要素が示されます。以下のスクリーンショットには新たに追加された要素のアークロールを示します。

作成したタクソノミとXMLSpyパッケージに含まれているものを比較することもできます。提供されるタクソノミ(`nanonull.xsd`)は`C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント\Altova\XMLSpy2021Examples\XBRLExamples\Nanonull`以下のフォルダーに含まれます。

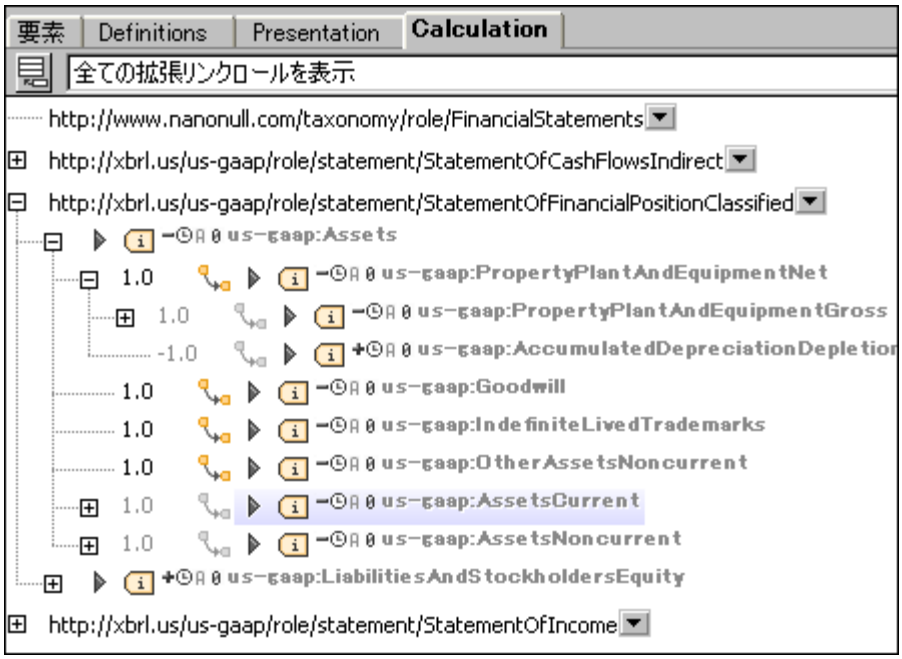
17.2.10 リレーションシップの作成：パート2

以前のセクションのリレーションシップの作成：パート1 757では、Definitions リレーションシップの作成方法がデモされました。Presentation リレーションシップ（以下のスクリーンショット）ならびに検証リレーションシップも同じような方法で作成されます。唯一の違いは、これらのリレーションシップにはアークロール列が存在しないことです。



以下の点に注意してください：

- プレゼンテーション ならびに 計算 リレーションシップは、親子関係のような2つの要素間の単純なアークとして見ることができます。アークアイコンによりこの関係が強調されます。要素にアークを挿入することは、グラフィカルに子要素を作成するのと同様な操作と考えることができます。従って、アークを使用することで階層構造を構築することができます。
- グローバル要素入力ヘルパーから直接ツリーに要素をドラッグすることもできます。これら要素は常にアークの「to」側に配置されます。要素がドロップされると、その位置に矢印が表示されます。
- Calculation アークには、アークにより「to」要素の値が合計されるときに使用される `weight` 属性があります（以下のスクリーンショット参照）。例えば、`weight` の値が `+1.0` であれば、要素の `100%` の値が「from」（または `summation`）要素へ加えられます。属性の値が `-1.0` であれば、`100%` の値が `summation` 要素から引かれます。`weight` 属性をダブルクリックすることで、異なる値を入力することができるようになります。



`weight` 属性は(以下のスクリーンショットにあるように)詳細入力ヘルパーからも変更することができます。

アーク使用の禁止

それが定義、Presentation、Calculation であるかに関わらず、すべてのアークには、`optional` または `prohibited` の値を持たせることができます。`prohibited` の値が使用されると、アークは無効になります。

表示色とコンテキストメニュー

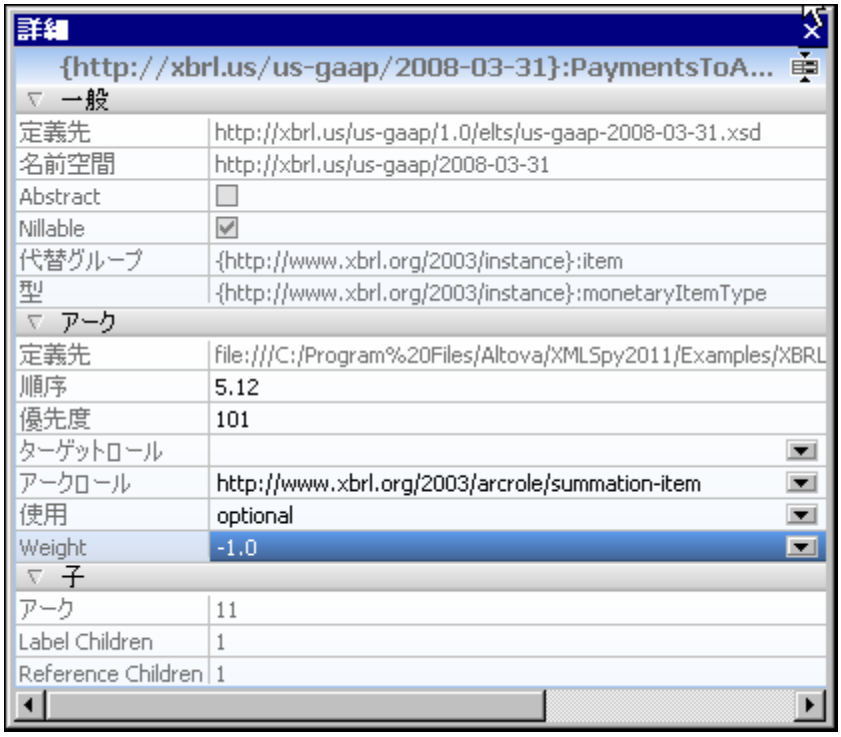
現在のタクソノミ内に要素が作成され、編集することができるのであれば、要素は黒色で表示されます。そうでない場合(それら要素がインポートされたタクソノミ内にあり、編集することができない場合)、要素はグレーで表示されます。

リレーションシップタブのコンテキストメニューには以下のエントリが表示されます。

- *要素参照の挿入* : 拡張リンクロールで利用可能。リンクロール以下に、アークの「from」側となる要素を作成します。
- *要素参照の削除* : リンクロール直下の要素参照にて利用可能。
- *アークの挿入* : 要素にて利用可能。アークの to 側となる要素を選択するためのダイアログが表示されます。
- *ターゲットロールの設定* : 選択された要素にターゲットロールをセット。
- *ラベルリンクロールの追加* : 選択された要素にラベルリンクロールを追加。
- *参照リンクロールの追加* : 選択された要素に対して参照リンクロールを追加。
- *アークのオーバーライド* : アークの `use` 属性の値を `optional` から `prohibited` に置き換え、アークを無視します。
- *アークの削除* : 選択されたアークを削除。
- *グローバル要素で表示* : 選択された要素をグローバル要素入力ヘルパーにてハイライトします。

詳細入力ヘルパー

リレーションシップ内の要素が選択状態であれば、詳細入力ヘルパーを使用してアーク属性を編集することができます(以下のスクリーンショットを参照)。



`order` や`priority` といったメインウィンドウのグラフィカルな表示からは編集できない属性は、詳細入力ヘルパーにて編集を行うことができます。

(C) 2015-2021 Altova GmbH

17.3 追加プロシージャ

追加プロシージャセクションでは、役に立つ他の機能が提供されています:

- 優先されるラベル763
- 型付けされているドメイン764
- 重複の検知と重複の解除765
- インライン XBRL766

17.3.1 優先されるラベル

リレーションシップ アークは複数のラベルを持つことができ、これらのラベルの 1 つのみが使用されている場合、以下のメカニズムが使用されます:

- 優先されるラベルが定義されると、このラベルが使用されます。
- 優先されるラベルが存在しない場合、デフォルトのラベルが使用されます。

Generic Preferred Label 1.0 Recommendation アドオン仕様により、`gpl:preferredLabel` 属性が定義、計算、フォーミュラ、およびテーブルリンクベース内のアーク上で定義することができます。属性の値は、ターゲット ノードのラベルを選択するために使用されるラベル ロールです。この結果、優先されるラベルは、(前の仕様で有効化されている) プレゼンテーション リレーションシップだけではなく、(検証リレーションシップなどの) 他のリレーションシップのためにもサポートされます。

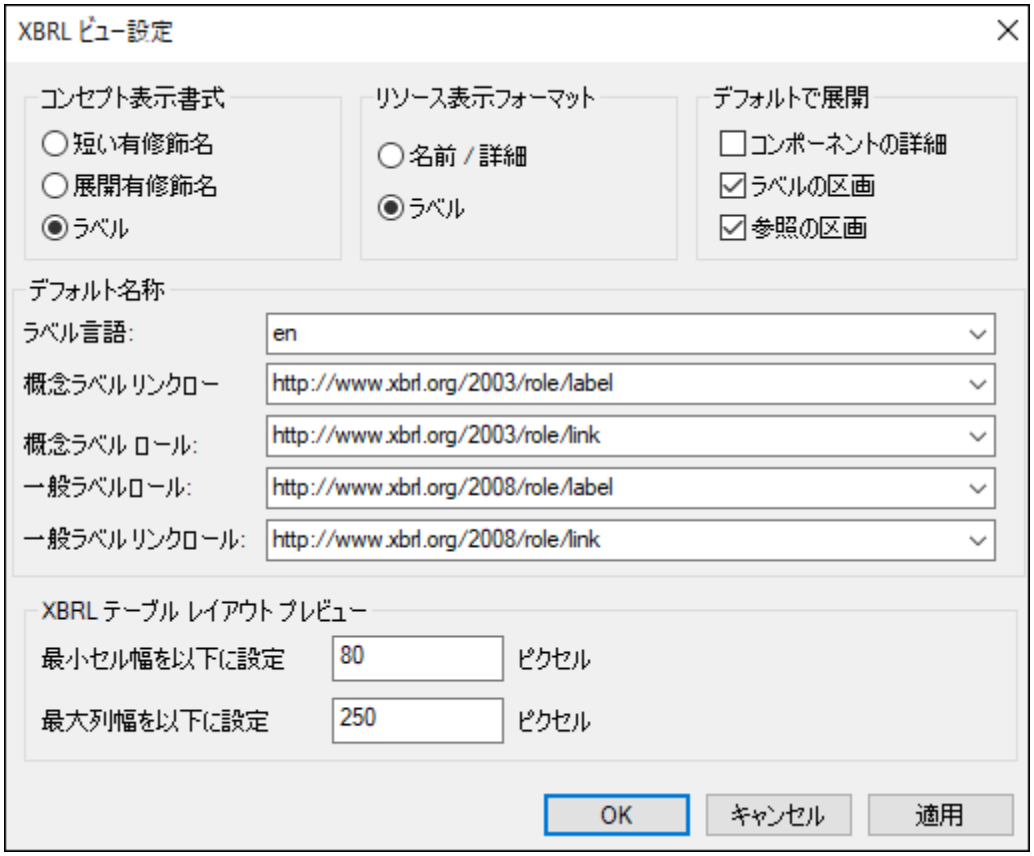
優先されるラベル
メイン タブ内でリレーションシップを選択し、詳細入力ヘルパー内で希望するラベル ロールを *優先されるラベル* プロパティの値として選択して、優先されるラベルを定義することができます (*下のスクリーンショットを参照*)。下のスクリーンショットでは、2 番目の検証リレーションシップには、定義されている 2 つのラベルが存在することを確認できます。詳細入力ヘルパー内では、優先されるラベルは `urn:mylabel` に設定されています。これはリレーションシップのラベルとして使用されるラベルです。(XMLSpy 内では、XBRL ビューの設定1401 内で、(名前の代わりに) ラベルがコンセプトおよび/またはの表示内で表示されるように指定することができます)。

メモ テーブルリンクベース内では、*優先されるラベル* プロパティは、これらのノードのためにラベルを指定することができないためリレーション シップ ノード、アスペクト ノード、マージされたルール ノードのために表示されます。

デフォルトのラベル
コンセプトまたはソースの *優先されるラベル* プロパティが設定されていない場合、コンセプトとソースのデフォルトのラベルが XBRL ビューの設定1401 ダイアログ内で指定されます。

XBRL ビューの設定内のラベル設定

XBRL ビューの設定 ダイアログはメニュー コマンド「XBRL | ビューの設定」によりアクセスすることができます (*下のスクリーンショット*)。

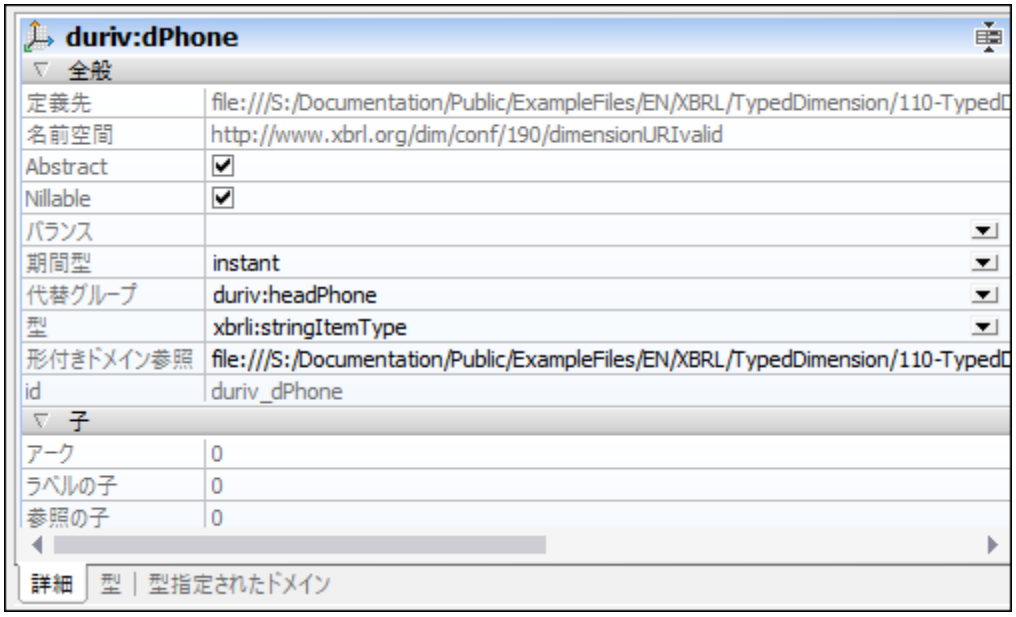


このダイアログ内ではラベルに関連する2つの設定が存在します:

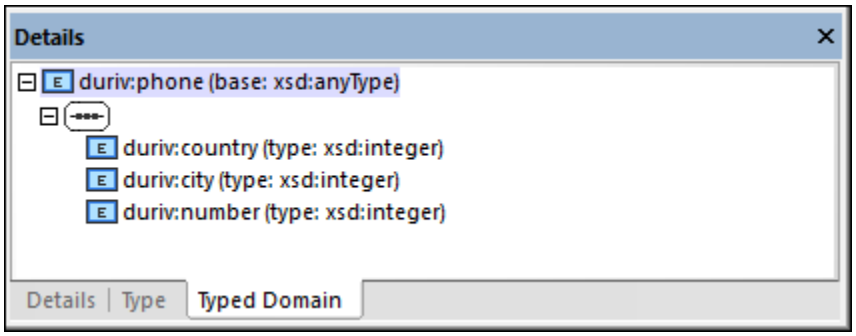
- コンセプトおよび/または リソースは名前の代わりにラベルと共に表示されます。
- (リソースのために使用される)コンセプトとジェネリックなラベルのためにデフォルトを設定することができます。

17.3.2 型付けされているドメイン

型付けされているドメインは型付けされたディメンションにより参照されている要素の宣言です。例えば、下の詳細入力ヘルパーのスクリーンショットで、型付けされたドメイン `duriv:phone` を参照する型付けされているディメンション `duriv:dPhone` を確認することができます。



参照されている型付けされているドメインに関する情報を確認するには、型付けされたドメインタブをクリックします（*下のスクリーンショットを参照*）。型付けされたドメインタブは、型付けされているドメインを参照する型付けされているディメンションが選択されている場合のみ表示されます。



17.3.3 重複の検知と重複の解除

XBRL インスタンスドキュメント内のファクトの重複の処理のために以下のサポートを使用することができます:

- XBRL インスタンスドキュメント内では、重複するファクトのリストを検知し、メッセージウィンドウ内で表示することができます。メニューコマンド「**XBRL | 重複の検知**」または「**XBRL | サーバー上で重複を検知する（ハイパフォーマンス）**」を実行します。詳細に関しては、コマンドの説明[1408]を参照してください。
- XBRL インスタンスドキュメントを検証する際、計算に影響を与える重複ファクトが検証の際に無視されるように指定することができます。この機能を有効化する設定はXBRL 検証オプション[1485]として使用することができます。
- フォーミュラが実行されると、テーブルが生成され、インライン XBRL が変換され、存在する重複が無視されます。この必要条件は、XBRL 処理オプションダイアログ[1411]内で指定することができます。

XBRL とインライン XBRL 1.0 内で重複するファクトを扱う方法 仕様内で設定されるルールを基に重複は決定されます。

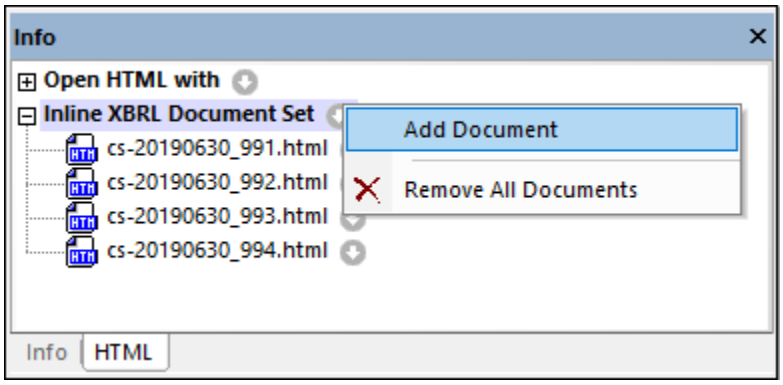
17.3.4 インライン XBRL

XBRL データを含むインライン XBRL ドキュメントは(XBRL タグを使用してマークアップされている) HTML ドキュメントです。アクティブなインライン XBRL ドキュメントを(「**XML | 検証**」) を使用して検証、および(「**XBRL | インライン XBRL の変換**」) を使用して処理することができます。ドキュメントの処理中、HTML ドキュメントから XBRL データを抽出します。

複数のインライン XBRL ドキュメント

複数のインライン XBRL ドキュメント検証/処理することができます。これを以下のように行います:

1. アクティブなドキュメントにするために、メインインライン XBRL ドキュメントを開きます。 HTML タブが情報ウィンドウ[112]内に表示されます *(下のスクリーンショットを参照)*。



2. HTML タブ内で、*インライン XBRL ドキュメントセット* のメニューボタンをクリックし、「**ドキュメントの追加**」*(スクリーンショットを参照)* をクリックして追加するインライン XBRL ファイルを参照します。
3. 検証または処理コマンドを実行します。

17.4 XBRL フォーミュラエディター

XBRL フォーミュラ 変数とフィルター仕様は、XBRL ビジネス報告書内のデータから新しいファクト値を取得するために使用されることのできる、式ルールのための構文を提供します。ジェネリックなラベルおよび参照仕様は、異なる構成のXBRL をラベルするすべての方法をサポートします。 XBRL フォーミュラ のコンテキストでは、ラベルと参照は、ヒューマンドキュメントとフォーミュラを関連付けるために使用することができます。変数は、XBRL ビジネス報告書内のどのファクトがフォーミュラ内の評価の使用方法のために変数により選択されるかをフィルターします。 検証と3つのアサーション 仕様は、ビジネス報告書の予期されるコンテンツについてのルールの構文を、変数および変数とフォーミュラのセットにより定義します。XBRL フォーミュラの構文とセマンティクスの概要については、Working Draft of XBRL Formula Overview 1.0 で確認することができます。

XMLSpy のXBRL フォーミュラ エディターは、アプリケーションのXBRL タクソノミ エディターの一部として実装されます。 XBRL ビューのフォーミュラ タブで使用することができます（*下のスクリーンショット参照*）。

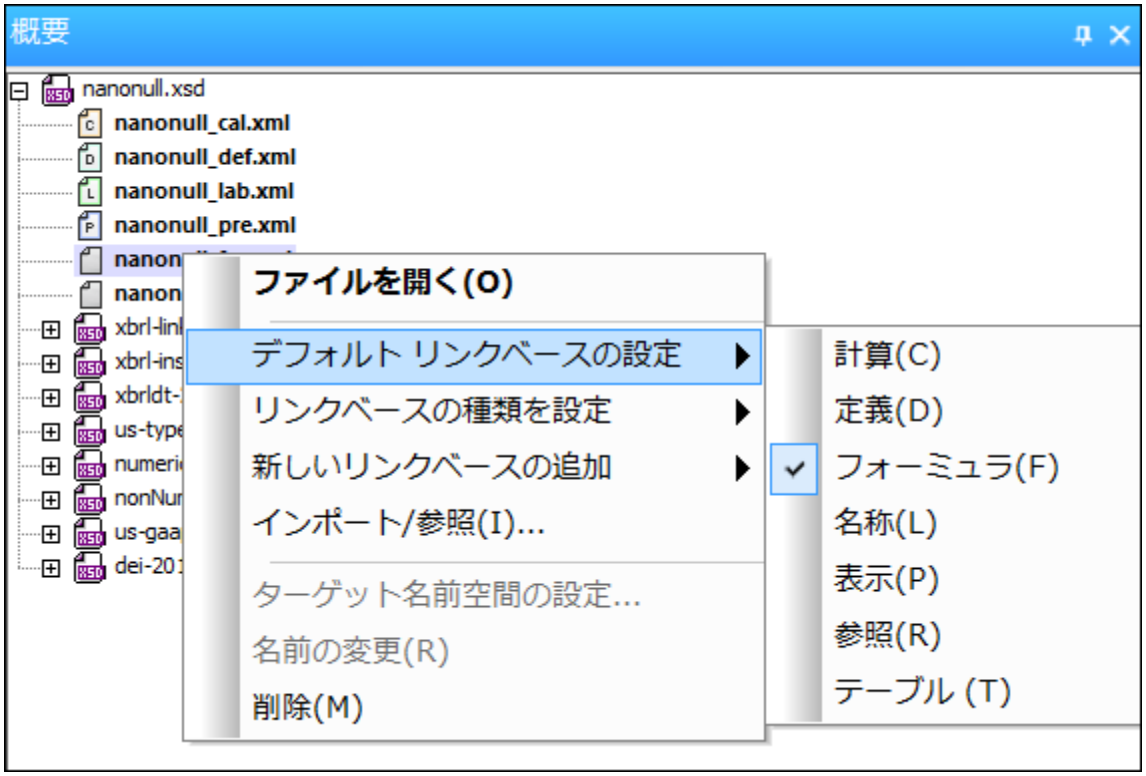
フォーミュラ タブ概要入力ヘルパーおよび詳細入力ヘルパーとともに使用され、フォーミュラを作成および編集します。概要入力ヘルパーは、XBRL フォーミュラのデフォルトのリンクベースを設定するために使用されます（フォーミュラ がデフォルトで保存されるファイルです）。詳細入力ヘルパーは、フォーミュラコンポーネントのコンテンツを編集するために使用されます。（このような編集は、フォーミュラ タブで直接行うこともできます）。

17.4.1 フォーミュラ リンクベース およびリンクロール

標準のXBRL リンクベース（定義、プレゼンテーション、計算）は、コンセプト間のルーションをロケーターと標準拡張リンク内の標準アークで定義しますが、フォーミュラ リンクベースは、フォーミュラ コンポーネント（フォーミュラ、変数、フィルター、アサーションなど）間のルーションを定義します。これらの定義は、リソースとジェネリックな拡張リンク内のジェネリックなアークにより指定されます。

フォーミュラリンクベースの追加

概要入力ヘルパー内(*下のスクリーンショット*) タクソノミファイルまたは、既存のリンクベースを右クリックして、「**新規リンクベースの追加 | フォーミュラ**」を選択します。追加されたリンクベースは、デフォルトのフォーミュラリンクベースファイルになります。他のフォーミュラリンクベースファイルをデフォルトのフォーミュラリンクベースにする場合は、右クリックして、「**デフォルトのリンクベースの設定 | フォーミュラ**」を選択します(*下のスクリーンショット*)。



デフォルトのリンクベースは、太字で表示されていて、保存されていない変更されたリンクベースは、アスタリスクでマークされていることにご注意してください。

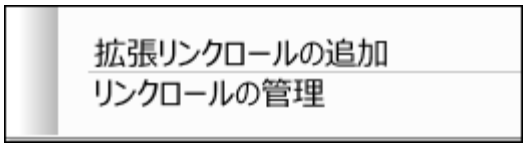
フォーミュラリンクベース フォーミュラ タブは、フォーミュラタブで表示されています。

メモ フォーミュラリンクベースが存在しない場合に[フォーミュラコンポーネントがタクソノミに追加された場合](#) 769、フォーミュラリンクベースは自動的に作成されます。

リンクロール

標準拡張リンクのケースと同様(定義、プレゼンテーション、計算)、ジェネリックなリンクは、同じ型のリレーションシップを区切り、結合が解除されたネットワークにするリンクロールの値を定義する必要があります。同じリンクを持つすべてのジェネリック名拡張リンクは、異なるリンクベースファイル内に存在する場合でも、フォーミュラタブ内のダイアグラム内で1つのリンクロールノードの下で結合されています。

ジェネリックなリンクロールは、背景エリアのコンテキストメニューを介したダイアグラム内で作成することができます。(*下のスクリーンショット*)。ですが、このコンテキストメニューは、フォーミュラタブ内のビューオプションコンボボックスが*すべての拡張リンクロールを表示*に切り替えられている場合のみ表示されることにご注意してください。

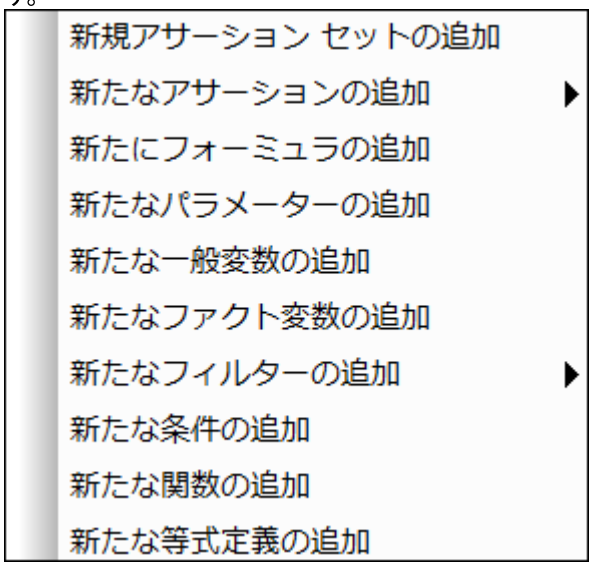


このメニューは、ツールバーアイコン *拡張リンクの追加/リンクロールの管理* を介して使用することができます。リレーションシップネットワークは、フォーミュラリンクベースにとって、リレーションシップネットワークはそれほど重要ではないため、デフォルトのフォーミュラタブのデフォルトの表示は、リンクロールを隠し、フォーミュラコンポーネントをリンクロール無しで表示する*拡張リンクロールを隠す*です。

デフォルトのフォーミュラリンクベースファイルがない場合、デフォルトのフォーミュラリンクベースが自動的に作成されます。また、リンクロールが作成された時点でデフォルトのリンクベースファイルにリンクロールが存在しない場合、デフォルトのリンクベースファイルが作成されます。

17.4.2 フォーミュラ コンポーネント

新しいフォーミュラコンポーネントは、リンクロールノードのコンテキストメニューを介して作成されます。(*下のスクリーンショット*)。または、 ツールバーアイコン、 *新しいフォーミュラコンポーネントの追加を介して、拡張リンクロールを隠す*に設定されたビューにより作成されます。



複数のコンポーネントを含むメカニズムについては、このセクションのサブセクションで説明されています。フォーミュラコンポーネントが追加された後、フォーミュラタブ内のダイアグラムに表示されます(*下のスクリーンショット参照*)。

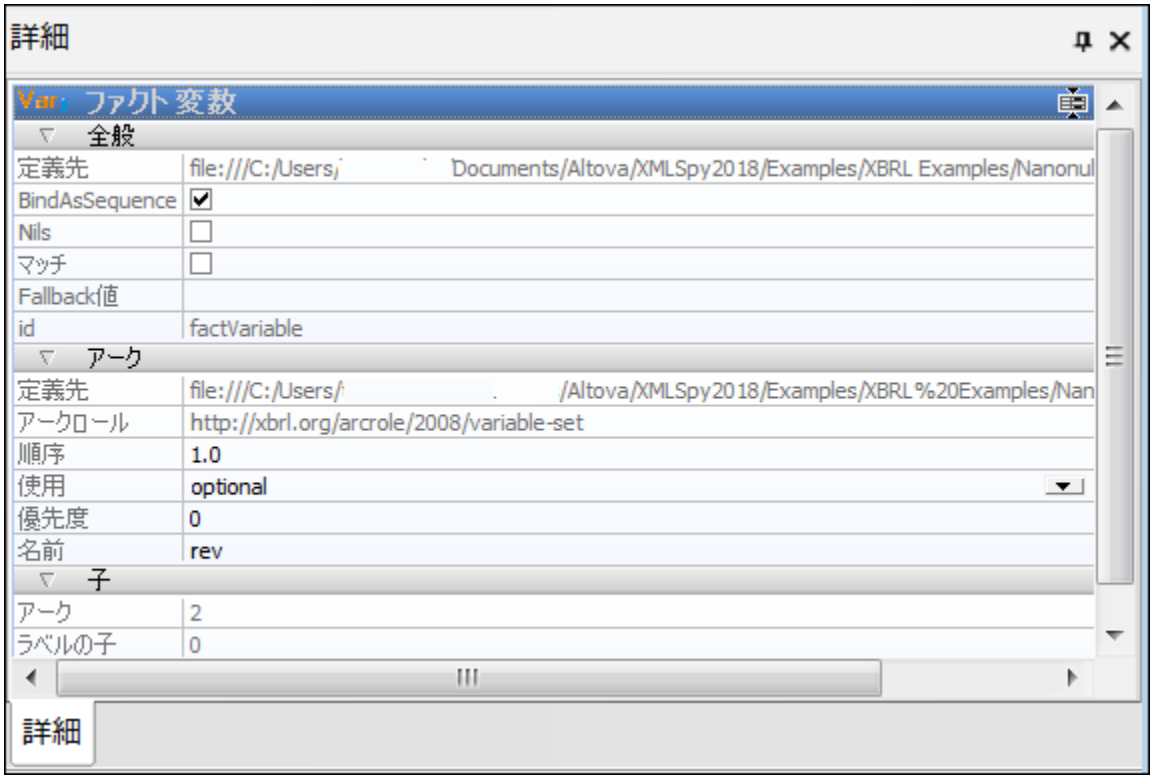
意味を明確にするために、フォーミュラコンポーネントは、*リレーションシップ*と他のコンポーネント（*アーク*）のセクションに分割されており、ツリー構造内に表示されています（*上のスクリーンショット参照*）。

コンポーネントのプロパティおよび*リレーションシップ*（*アーク*）は、アイコンコンポーネントのまたは*アーク*の左側のアイコンとして、ダイアグラム内に表示されています。（*下のスクリーンショット参照*）.

例えば、上のスクリーンショット内では、ファクト変数コンポーネントには3つのプロパティがあります:（S アイコンにより表示されている）`BindAsSequence`、`Nils` (N アイコン) および、`Matches` (M アイコン)。これらは、すべてboolean プロパティです。ダイアグラム内でアイコンからのラインを持たない（`BindAsSequence`）は、、`true` の値を持ちます。他の2つのプロパティは、（それぞれラインを介して表示された）`false` の値を持ちます。（変数の下の）アークは、2 つのプロパティを持ち、最初プロパティは、boolean `false`、二番目のプロパティはboolean `true` です。

ファクト変数の詳細入力ヘルパー（*下のスクリーンショット*）では、変数のプロパティは、全般的なセクションの下にリストされています。boolean プロパティの値は、`true` の場合はチェックボックスをチェック、`false` の場合はチェックを解除することにより指定します。

(C) 2015-2021 Altova GmbH



詳細入力ヘルパー内のアークのプロパティを確認するには、ダイアグラム内の to (destination) コンポーネントを選択します。アークのプロパティはアークセクションにリストされています。

フォーミュラ エディター内のコンテキスト メニュー

フォーミュラ コンポーネントのコンテキスト メニューは、コンポーネントの型により異なります。メニュー アイテムは、以下のセクションに整理されています:

- コンテンツの修正 (フォーミュラ、フィルターの一部、カスタム関数のため): 例えば *追加/追加 アスペクト ルール*
- リレーションの修正 (サブ アイテム のみのため): *オーバーライド/削除 アーク*
- *ラベル参照の追加*
- 新しい子コンポーネントの作成 (リレーションシップを含む): 例えば、*新しいフィルターの追加*
- コンポーネントの削除 (リレーションシップを含む)
- (コンポーネントの) *次の/前の発生を検索*

メモ コンテキスト メニューにより作成または削除することのできる*コンセプト アスペクト ルール*などのコンテンツ アイテムは、追加セクション内の詳細入力ヘルパーに表示されます。

17.4.2.1 アサーション と アサーション セット

3つの型のアサーションがあります:

- 値アサーション
- 存在アサーション

- 整合性 アサーション

値アサーション

値アサーションは、フォーミュラ リンクベース 機能で頻繁に使用され、入力 XBRL インスタンス ファクトを XPath 式に対して、確認する方法を提供します。プロパティ *アスペクト モデル* そして *明示的なフィルター* をアイコンとして提供します。プロパティ `test` の値は、XPath 式です。

存在アサーション

存在アサーションは、書式の型、会社のID、ファイルのID などの静的な存在をチェックするために役に立ちます。 これは、プロパティ *アスペクト モデル* および *明示的なフィルター* をアイコンとして提供します。プロパティ `test` の値は、XPath 式です。

整合性アサーション

整合性 アサーションは、関連するフォーミュラにより作成された出力ファクトが、入力インスタンス内のすべてのアスペクト一致したファクトに対して整合性を持つかを指定します。.これは、Boolean プロパティ `strict` をアイコンとして提供します。*絶対的アクセスを受け入れる半径* および *プロポーショナル 受け入れ半径* のプロパティの値は XPath 式です。

アサーションが成立した/成立していないメッセージ

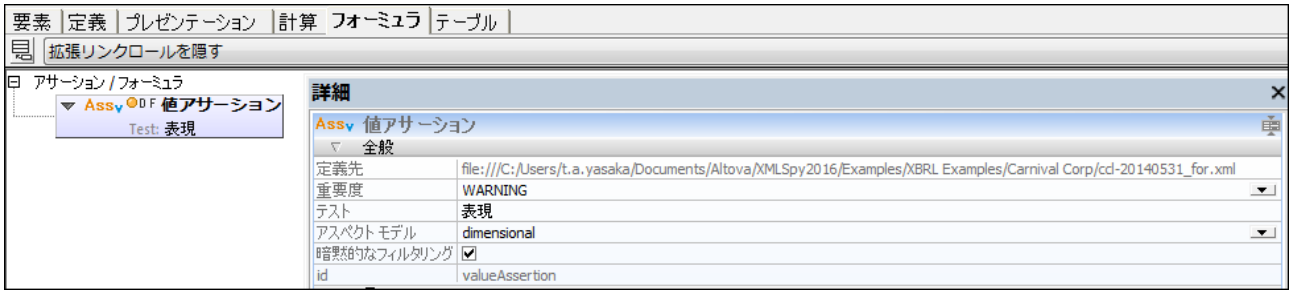
アサーションのこれらのサブコンポーネントは、アサーション評価をもつメッセージの関連を有効化します: 成立した メッセージ は、成功した評価が与えられ、成立していない メッセージ には失敗したメッセージが与えられます。これらの メッセージ は、各アサーションのコンテキスト メニューを介して追加されます。

アサーション-条件が満たされない-重要度 リレーションシップ

アサーションは条件が満たされるか、満たされないかです。ですが、アサーションには異なる重要度のレベルのルールがあり、満たされないアサーションはその特定の満たされないレベルの重要度に従い、満たされないアサーションは 分類されます。3つの標準の重要度のレベルがあります: エラー、警告、OK。デフォルトの重要度はエラーです。重要度は、アサーションが定義された重要度に関連しない場合、呼び出されます。

アサーション-重要度が満たされない-重要度 リレーションシップは、アサーションと定義された重要度リソースの1つの間のリレーションシップです。以下を持つ XLink アークと共に表示されます: (i) アークロールの値 `http://xbrl.org/arcrole/PR/2015-11-180/assertion-unsatisfied-severity`、(ii) 開始リソースとしてのアサーション、および (iii) 終了リソースとしての重要度リソース。

タクソノミーエディター内で、重要度リレーションシップはダイアグラム内のアサーションコンポーネントの 重要度 アイコンをクリックすることで指定することができます (*下のスクリーンショット参照*)。そして、表示されるポップアップから重要度のレベルを選択します。または、アサーションの詳細入力ヘルパー内で選択することもできます (*スクリーンショット参照*)。



アサーションセット

アサーションセットには、1 つまたは複数の アサーションが含まれます。アサーションセットのコンテキストメニューは、アサーションセットへの各アサーションの追加を許可します。

17.4.2.2 フォーミュラ

フォーミュラは、出力 XBRL ファクトを構築するためのルールのセットを、フィーミュラの変数セット内で評価された変数に値を変換することで示します。変数の値は、入力 XBRL インスタンス およびサポートされる DTS または フォーミュラを処理するアプリケーションからから取得されます。

値ルールは、ファクトに割り当てられる値を取得する XPath 式です。コンスタントなどの、シンプルな 式 であることができます。または、変数セット、他の変数セットからの値のチェーン、および/または カスタムおよび 内蔵された 関数から計算された値の変数および パラメーターを参照する用語を含むことができます。

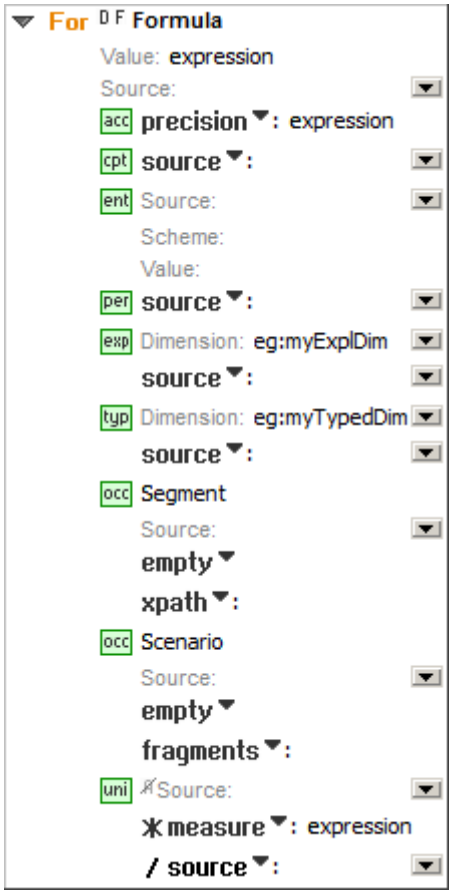
XBRL 内では、非-分数 数値 ファクトは、正確性/小数点 属性のかたちで正確性についての情報と共に報告されます。ですから、フォーミュラは、出力ファクトのために アサーとされる正確性の決定を左右する正確性 ルールを含む可能性があります。

出力ファクト 値と正確性を決定するルールと共に、フォーミュラは出力値を解釈するために必要なすべての出力アスペクトのためにアスペクト ルールを指定または暗黙に示します。出力コンセプト、出力コンテキストと(数値ファクトのための) 単位の出力ユニットはすべて異なる型のアスペクトルーツです。ト

(部分的または全体の) アスペクトが、ソースを指定することにより、評価の バインドされた変数から取得される可能性があります。ソースを指定されることができます。ソースはルールにより指定される、またはフォーミュラのフォーミュ(または タプル)ラ要素のソースから継承されることができます。on a ルール または 複数のソースが存在する場合、アスペクト ルールに近いものが優先されます。

フォーミュラが挿入されると、正確性 または アスペクト ルールを持ちません(*下のスクリーンショット*)。

正確性 と アスペクト ルールは、フォーミュラの コンテンツおよび追加された(または 削除された) コンテキスト メニューを介して定義されます。下のスクリーンショットは、可能性のあるすべての正確性とアスペクト ルールを持つフォーミュラを表示します。



追加セクション内の詳細入力ヘルパーに正確性とアスペクトルールは表示されています。

正確性 ルール

型: precision または decimals
値: XPath 式

アスペクトルール

アスペクトルールは、型によりグループ化されています。

コンセプト ルール
型: qname、expr、または source
値: コンセプトの QName、XPath 式、または ソース 変数（または カバーされていない QName）

エンティティ識別子 ルール
ソース: ソース 変数（または カバーされていない QName）
スキーム/値: XPath 式

期間 ルール
型: instant、duration、forever または source
値: 値の XPath 式、開始/終了/ソース、値無し または ソース 変数（または カバーされていない QName）

(C) 2015-2021 Altova GmbH

明示的なディメンション ルール
ディメンション: 明示的なディメンション ルールに影響されたディメンションのQName。
型: qname、exp、omit または source
値: メンバーのQName, メンバーのXPath 式, 値無し または ソース 変数（またはカバーされていないQName）

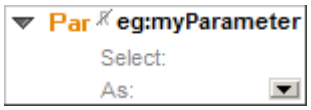
型指定された ディメンション ルール
ディメンション: 明示的なディメンション ルールに影響されたディメンションのQName。
型: xpath、value、 omit または source
値: XPath 式, XML 要素, 値無し または ソース 変数（またはカバーされていないQName）

オープンコンテキスト コンポーネント ルール
OCC ルールは、型によりセグメント OCC ルールおよびシナリオOCC ルールなどにグループ化されています。
ソース: ソース 変数は、最初のOCC ルール内で定義されています。
For each OCC ルール
型: empty、fragments、または xpath
値: 値なし、XML 要素、またはXPath 式

ユニット ルール
Boolean *フラグ 引数* は、*ソースアスペクト* 値が使用されるかを指定します。
For each ユニット 乗算/除算 ルール
型: *measure、 /measure または *source、/source
値: メジャーのXPath 式 または ソース 変数（またはカバーされていないQName）

17.4.2.3 パラメーター

パラメーターは、XPath 式内で参照されることができます。これは、*必要とされる フラグ*を提供します。設定されると、パラメーターは必須です。値は、処理アプリケーションにより与えられる必要があります。パラメータが必須ではない場合、また処理アプリケーションより値が与えられない場合、与えられた値 は、プロパティ *Select* 内で与えられたXPath 式 を使用して計算される可能性があります。オプショナルプロパティ *As* は、パラメータにより必要とされるデータ型を指定します。



17.4.2.4 変数

変数は、入力データのバインドの方法、通常はファクト アイテムを、変数名により参照される、アサーションまたは フォーミュラ 式などの名前に対して宣言します。 入力 ファクト アイテムをバインドする変数は、ファクト変数であり、フィルターを使用して入力内で何がバインドされるかを宣言します。一般 変数は、中間の式の結果と他の種類の処理に使用されます。

一般 変数

一般 変数は、Boolean プロパティのシーケンスとしてバインドをアイコンとして提供します。プロパティの選択の値は、XPath 式です。

ファクト変数

ファクト変数は、Boolean プロパティのシーケンスとしてバインド、およびアイコンとして一致を Nils として提供します。プロパティフォールバック値 の値は、XPath 式です。

17.4.2.5 フィルター

フィルターは、入力 XBRL インスタンス内の変数が評価される際に使用される XBRL インスタンスであるファクトのための選択条件を定義します。フィルターは、入力 ファクトに適用される条件を示します。 フィルターの一部には、サブラインに表示された XML コンテンツを持ちます。

Aspect Cover
これらのフィルターは、フィルタリングを行わず、明示的な XPath 式を持ちません。これらは、(コンセプト および ディメンションなどの) 他のフィルタの後に処理または適用され、他のフィルターのアプリケーションにより引き起こされるアスペクトのカバー状態をオーバーライドします。
1 つまたは複数のアスペクトアイテム
型: aspect, dim-qname/excl-dim-qname または dim-exp/excl-dim-exp
値: アスペクト 型 (enum), ディメンションの QName または XPath 式
アイテムは、入力ヘルパー詳細追加セクションに表示されています。

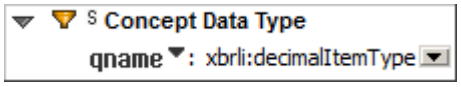
Boolean Filter
Boolean フィルターは、サブフィルターに関連付けられています。
and-filter は、サブフィルターにより表された条件をベースにしたファクトに一致します。

or-filter は、サブフィルターにより表された条件をベースにしたファクトに一致します。

Concept Name
Concept Name フィルターは、コンセプトの名前をベースにしたファクトに一致します。

1 つまたは複数のコンセプト:
型: qname または exp
値: コンセプトの QName または XPath 式
コンセプトは、入力ヘルパー詳細追加セクションに表示されています。

Concept Data Type
Concept Data Type フィルターは、スキーマデータ型をベースにしたファクトに一致するために使用されます。

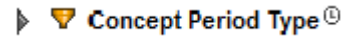


Boolean フラグ: "strict" は、ファクトのデータ型が割り出されるかを指定します。
型: qname または exp
値: データ型の QName または XPath 式

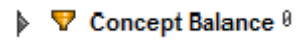
Concept Substitution Group
Concept Substitution Group フィルターは、XML スキーマ代替グループをベースにしたファクトに一致するために使用されます。

Boolean フラグ: "strict" は、ファクトのコンセプトが自身の @ substitutionGroup 属性内の要素を直接指定するかを指定します。
型: qname または exp
値: 代替グループの QName または XPath 式

Concept Period Type
Concept Period Type フィルターは、@ xbrli:periodType 属性により決定されたとおり、期間型またはインスタント型コンセプトのための値を報告するかどうかをベースにしてファクトに一致するために使用されます。



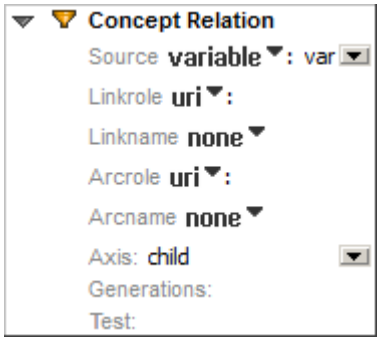
Concept Balance
Concept Balance フィルターは、@ xbrli:balance 属性を持つかどうか、また出金および入金の値があるかをベースにしたファクトに一致するために使用されます。



Concept Custom Attribute
Concept Custom Attribute フィルターは、存在または各コンセプトの宣言内の属性の値をベースにしたファクトに一致するために使用されます。

型: qname または exp
値: 属性の QName または XPath 式

Concept Relation
Concept Relation フィルターは、効果的なリレーションシップの指定されたリンクロールURI 内ネットワーク内の、指定されたアークロール URI の、指定された軸の、指定された生成を含む、オプションのテスト式を満たす、ソースコンセプトに対してのコンセプトの効果的なリレーションシップをベースにした、ファクトに一致するために使用されます。

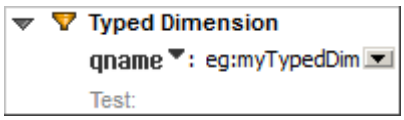


ソース 型 = 変数、qname または exp
リンクロール: 型 = uri または exp
リンク名: 型 = none、qname または exp
アークロール: 型 = uri または exp
アーク名: 型 = none、qname または exp

Explicit Dimension
フィルターディメンションの有効なメンバーのすべてのドメインの集合体内のすべてのドメインメンバーのセットとして与えられたDTS のコンテキスト内で明示的なディメンション のドメインが定義されています。 Explicit Dimensionフィルターは、明示的ディメンション値を持つ明示的ディメンションのドメイン内のドメインメンバーを持つファクトに一致するために使用することができます。

ディメンション 型: qname または exp
1 つまたは複数のメンバー:
型: 変数, qname または exp
メンバーは、入力ヘルパー詳細追加セクションに表示されています。

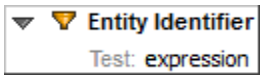
Typed Dimension
Typed Dimension フィルターは、型指定されたディメンションのための値をベースにしたファクトに一致するために使用されます。



ディメンション 型: qname または exp

(C) 2015-2021 Altova GmbH

Entity Identifier
Entity Identifier フィルターは、エンティティ識別スキームおよびまたはエンティティ識別値の特性をベースにするファクトに一致するために使用されます。



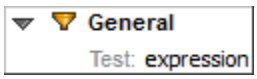
Specific Entity Scheme
Specific Entity Scheme フィルターは、スキーム識別フィルターのための報告値をベースにするファクトに一致するために使用されます。

Regular Expression Entity Scheme
Regular Expression Entity Scheme フィルターは、正規パターンエンティティ内のスキームをベースにしたファクトに一致するために使用されます。

Specific Entity Identifier
Specific Entity Identifier フィルターは、フィルターにより与えられたエンティティ識別し、値を使用する報告値をベースにするファクトに一致するために使用されます。

Regular Expression Entitiy Identifier
Regular Expression Entity Identifier フィルターは、エンティティ識別子の値のテキスト内の正規パターンをベースにしたファクトに一致するために使用することができます。

General
General フィルターはアスペクトをカバーしません。



Match Concept
Concept Matching フィルターに一致するコンセプトは、同じコンセプトのレポート値を持つファクトを選択するために使用されます。

Match Location
Location Matching フィルターは、同じ親の要素を持つファクトに一致するために使用されます。

Match Unit
Unit Matching フィルターは、同じユニットを持つファクトに一致するために使用されます。

Match Entitiy Identifier
Entity Matching フィルターは、同じエンティティ識別子とともにファクトを選択するために使用されます。

Match Period
Period Matching フィルターは、同じ期間を持つファクトを選択するために使用されます。

Match Dimension
Dimension Matching フィルターは、指定されたXBRL ディメンションのための同じ値を持つファクトを選択するために使用されます。

Match Complete Segment
Complete Segment Matching フィルターは、 XBRL ディメンション仕様をベースにして解釈されないセグメントのコンテンツがある同じセグメントを持つファクトを選択するために使用されます。

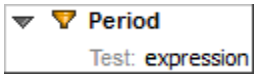
Match Non-XDT Segment
Non-XDT Segment Matching フィルターは、比較からXBRL ディメンション仕様 コンテンツを除いた後の同じセグメントを持つファクトを選択するために使用されます。

Match Complete Scenario
Complete Scenario Matching フィルターは、をベースにせずに解釈されたシナリオのコンテンツがある同じシナリオを持つ、ファクトを選択するために使用されます。

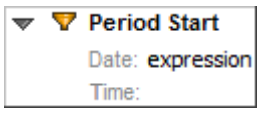
Match Non-XDT Scenario
Non-XDT Scenario Matching フィルターは、比較からXBRL ディメンション仕様 コンテンツを除いた後の同じセグメントを持つファクトを選択するために使用されます。

Period
Period フィルターは、計られたまたは期間に関連する条件の広い範囲をベースにした、ファクトに一致するために使用されます。

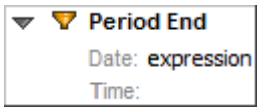
(C) 2015-2021 Altova GmbH



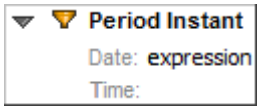
Period Start
Period Start フィルターは、計られたまたは期間の開始をベースにしたファクトに一致するために使用されます。



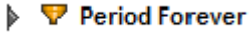
Period End
Period End フィルターは、計られたまたは期間の終了をベースにしたファクトに一致するために使用されます。



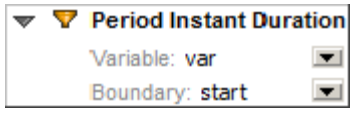
Period Instant
Period Instant フィルターは、計られたインスタントをベースにしたファクトに一致するために使用されます。



Period Forever
Forever フィルターは、繰り返し期間とともに報告されるファクトに一致するために使用されます。

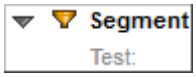


Period Instant Duration
Instant Duration フィルターは、他のファクトが報告された期間の開始または終了に一致するインスタントのあるインスタントでレポートされるファクト一致するために使用されます。

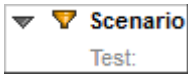


Relative
相対的なフィルターは、相対的なフィルターによりカバーされたアスペクトのためにファクトを選択するために使用されます。相対的なフィルタに一致するファクトは、評価される変数セット内の他のファクト変数の評価の結果である必要があります。

Segment
Segment フィルターは、指定された制約を満たす 非-XDT コンテンツを持つ、ファクトに一致するために使用されます。非-XDT コンテンツは、XBRL ディメンション使用で定義されている明示的なまたは型指定されたディメンションをベースにしないセグメントコンテンツを参照します。



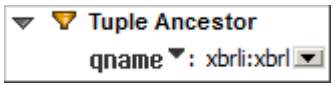
Scenario
Scenario フィルターは、指定された 制約を満たす、非-XDT コンテンツを持つファクトに一致するために使用することができます。非-XDT コンテンツを参照します。ディメンション仕様で定義されていない、明示的なまたは 型指定された ディメンションをベースとしないシナリオコンテンツを参照します。



Tuple Parent
Parent フィルターは、 指定された 親要素を持つファクトを選択するために使用することができます。

型: qname または exp

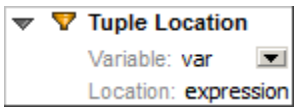
Tuple Ancestor
Ancestor フィルターは、指定された祖先要素を持つファクトを選択するために使用することができます。



型: qname または exp

Tuple Sibling
Sibling フィルターは、他のファクトの兄弟であるファクトを選択するために使用することができます。

Tuple Location
Location フィルターは、他のファクトのロケーションに相対的な指定されたロケーションを持つ、ファクトを選択するために使用されます。

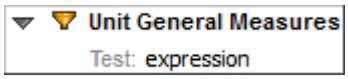


(C) 2015-2021 Altova GmbH

Unit Single Measure
Unit Single Measure フィルターは、単一の方法により指定されるユニットと共にレポートされているファクトに一致するために使用することができます。

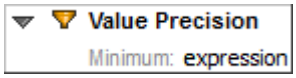
型: qname または exp

Unit General Measure
Unit General Measure フィルターは、ユニットメジャーの数値を含む条件をベースにした、ファクトを選択するために使用することができます。



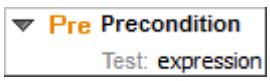
Value nil
Nil フィルターは、nil とレポートされているファクトに一致するために使用することができます。

Value Precision
Precision フィルターは、最小の実際の正確性を持つことをベースにし、@ decimal 属性の値により示されることのできるファクトに一致するために使用することができます。正確性フィルターは、フィルターが永続の最小の必須の正確性を示す場合、ファクトを選択しないことに注意してください。フィルターは、非数値または nil 値と共にレポートされるファクトを選択しません。



17.4.2.6 事前条件

事前条件は、バインドされた変数のセットが、フォーミュラ値 および 出力ファクト または アサーション値テスト または存在数をアクティブ化を決定する方法を提供します。

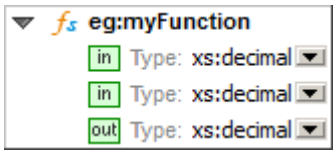


17.4.2.7 関数

カスタム関数は、XPath および XQuery 関数仕様、または、XBRL 関数 レジストリで定義されていない XPath 関数です。 カスタム関数は、XPath 式内で使用される可能性があります。

関数署名

関数署名 は、以下のスクリーンショットに表示されているとおりです。

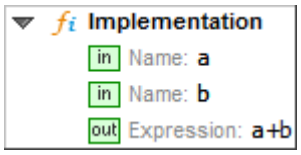


カスタム関数署名の子要素がある場合は、カスタム関数の 入力パラメーターのデータ型を指定します。カスタム関数の 入力パラメーターの順序は、カスタム関数署名の子要素のドキュメントの順序に一致します。

入力は、追加セクションの詳細入力ヘルパー内で表示されます。

関数の実装

関数の実装 は、以下のスクリーンショットに表示されているとおりです。



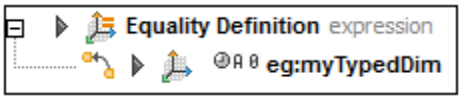
カスタム関数の実装（CFI）は、カスタム関数の実装を構成するXPath 式 を示し、カスタム関数の出力を定義するため、関数入力のための名前を定義する子要素のシーケンスを含みます。

関数実装 リレーションシップ は、カスタム関数署名 およびカスタム関数の実装間のリレーションシップです。関数の実装は、関数実装 リレーションシップのターゲットである必要があるため、対応する 関数署名の下に表示されます。リレーションシップが存在しない場合、（または署名異なるルクロールの下で定義されている場合） 実装は、*関数* セクションの下に直接表示されます。

入力とステップは、追加セクション内の詳細入力ヘルパーに表示されています。

17.4.2.8 等値定義

等値定義は、型指定されたディメンションドメイン内の2 つの値の等値の定義です。型指定されたディメンションドメイン 定義は、型指定されたディメンションを宣言する、XML スキーマ 要素上の `@xbrldt:typedDomainRef` 属性などにより識別された、型指定されたディメンションのためのコンテンツモデルを定義するXML スキーマ内の要素です。型指定されたディメンションドメイン 定義および等値定義間のリレーションシップである、等値-定義 リレーションシップが、等値定義 および対応する 型指定されたディメンション間の逆 リレーションとして、表示されます。

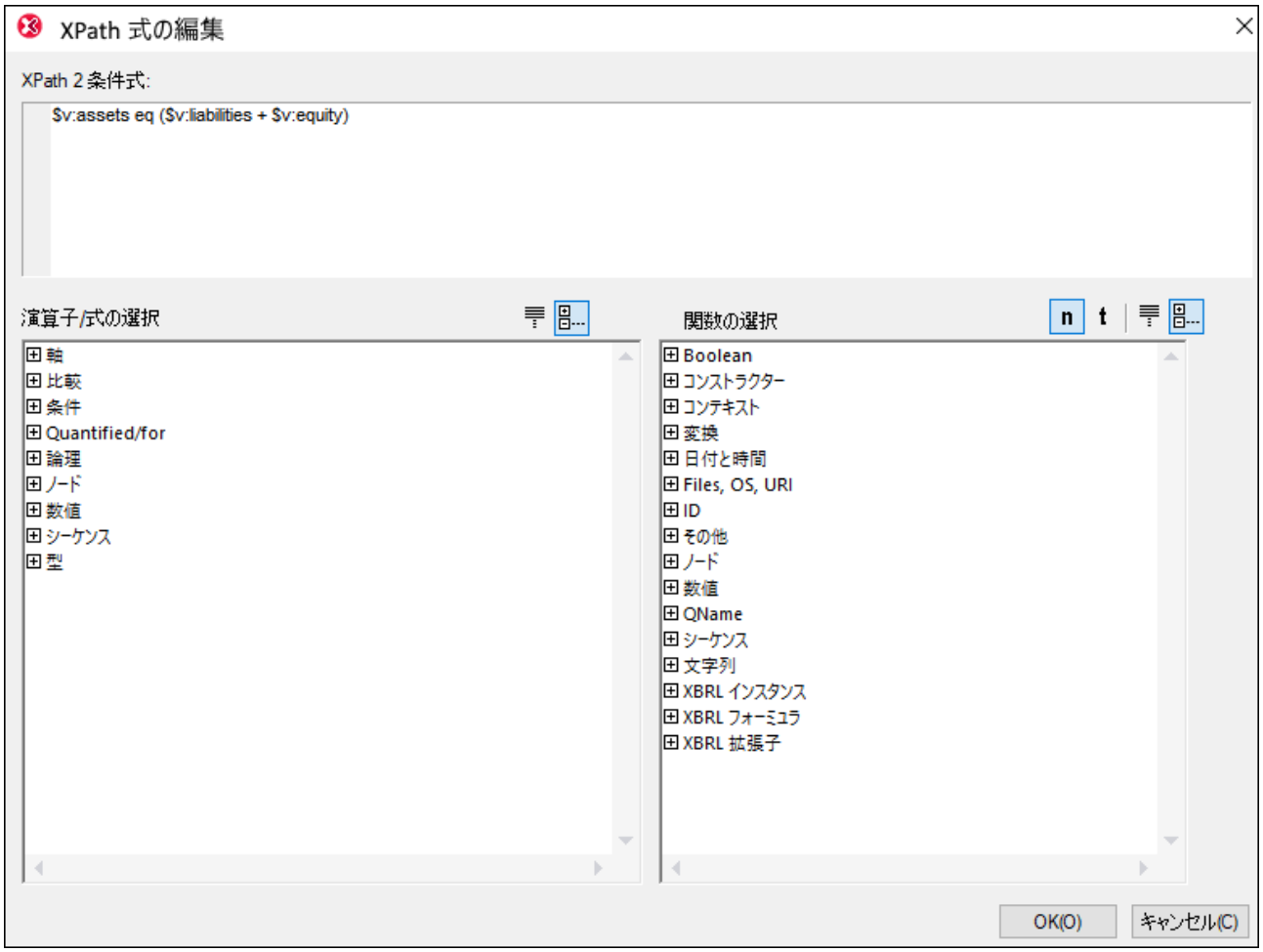


等値-定義 リレーションシップは、型指定されたディメンションをグローバル要素 入力ヘルパーから等値定義 コンポーネントへドラッグすることで構築することができます。等値定義または、まだ既存の等値-定義 リレーションシップ に含まれていない可能性があることにご注意してください。

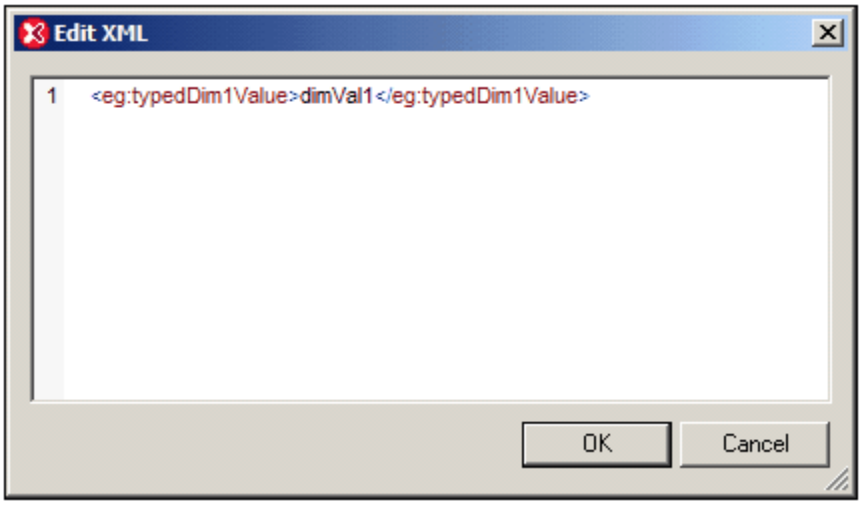
17.4.3 コンポーネント プロパティとコンテンツの編集

フォーミュラコンポーネンツのプロパティは、ダイアグラム内で、または詳細入力ヘルパー内で直接編集することができます。

ダイアグラム内で、コンポーネントに存在する場合は名前を入力すると適切な値のデフォルトのプロパティがプロパティ詳細テキストの横に灰色で表示されます。 コンポーネントをダブルクリックすると展開します。プロパティをダブルクリックすると、プロパティを編集モードにします。プロパティまたはコンテンツがXPath 式を含む場合、XPath 式の編集がポップアップします。



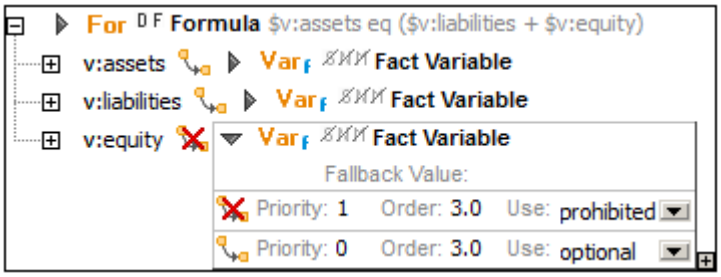
フォーミュラの型指定された ディメンション アスペクト ルールの内の*値のコンテンツ*または フォーミュラの OCC アスペクト ルール内の*フラグメントのコンテンツ*などのXML コンテンツの編集中、XML の編集ダイアログがポップアップします（*下のスクリーンショット*）。（フォーミュラを右クリックしてアスペクト ルールを追加します。）



XML テキストを入力して、「OK」をクリックすると、 XML テキストは、プロパティ内のコンテキストとして入力されます。XML テキストが整形式でない場合、この効果へのメッセージがポップアップしますが、テキストは許可されません。

17.4.4 フォーミュラコンポーネントリレーション

2 つのフォーミュラコンポーネント間のリレーションシップは、フォーミュラコンポーネントを他のコンポーネントにドラッグアンドドロップしてリンクすることにより行います。 リレーションシップはアークとともにダイアグラム内で、表示されています。(*下のスクリーンショット参照*)。



次の表示および編集の可能性があります:

- コンポーネントの子の順序は、ドラッグアンドドロップにより子を移動させることにより変更することのできるアークプロパティの *順序* の値によります。(*上のスクリーンショット参照*).
- (プロパティを含む)子コンポーネント リレーションをコピーまたは移動するために、コンポーネントをの上または下にドラッグします。
- 既存の(親) コンポーネントのコンテキストメニューを使用して、新しいコンポーネントを作成するには、(アークである)リレーションシップは自動的に生成されます。
- 子コンポーネントのコンテキスト メニュー内のコマンド「**アークのオーバーライドおよびアークの削除**」は、コンポーネントとその親の間のリレーションをオーバーライドまたは削除します。
- コンセプトリレーションに関しては、オーバーライドされたリレーションの複数のアークがサブラインに表示されます(*上のスクリーンショット参照*)。

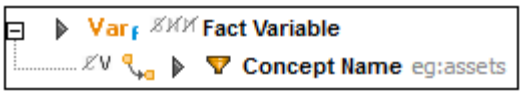
メモ フォーミュラコンポーネントリレーションのアークロールを変更することはできません。

変数セット リレーションシップ

変数セット リレーションシップは、(i) 変数セット リソース(値 アサーション、存在アサーション、またはフォーミュラ) 、(ii) 変数 (ファクト変数 または 一般変数) またはパラメーター間のリレーションシップです。変数の名前 またはパラメーターがアークアイコンの前に表示されます(下のスクリーンショット)。

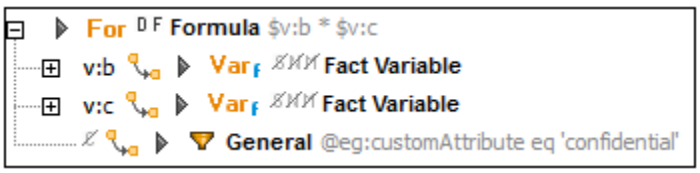
変数フィルター リレーションシップ

変数フィルター リレーションシップは、ファクト変数とフィルター間のリレーションシップです。Boolean *プラグ コンプリメント* (ダイアグラム内の、C アイコン) が設定され、リレーションシップは、変数フィルター リレーションシップにより補修されます。Boolean *プラグ カバー* (ダイアグラム内の、V アイコン) が設定されると、リレーションシップ は、カバーする変数フィルター リレーションシップはとなります(*下のスクリーンショットに表示*)。この場合は、フィルターは、フィルターされるファクトのアスペクトをカバーします。



変数セット-フィルター リレーションシップ

variable-set-filter リレーションシップ (*下のスクリーンショット参照*) は、変数セット リソースとフィルター間のリレーションシップです。variable-set-filter リレーションシップに含まれるフィルターは、基本的に、変数セット内のファクト関数に関連しており、関連するリソースにより定義されています。Boolean *プラグ コンプリメント* は、変数がフィルターコンプリメントを使用するか指定します。variable-set-filter リレーションシップによりファクト変数と関連する場合、通常アスペクトをカバーしません。



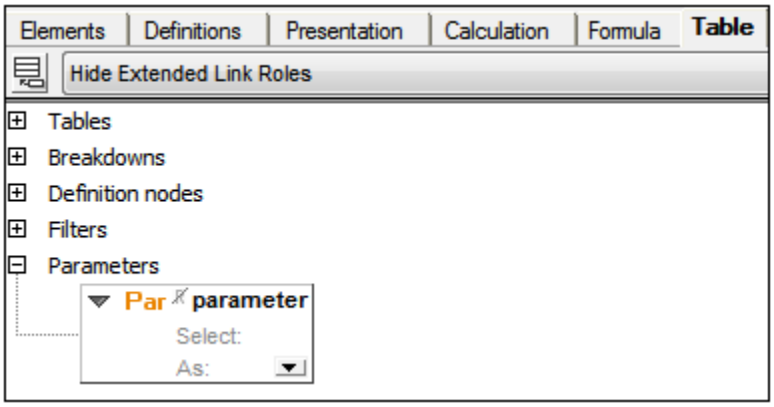
テーブル レイアウト プレビュー内で視覚的にフォーミュラをバインドする

XBRL タクソノミの開発者は、XBRL フォーミュラをポイントしてクリックすることにより作成する XBRL テーブル プレビューを活用することができます。 この機能については、テーブル レイアウト プレビュー内でフォーミュラをバインドする 818 セクションで説明されています。

17.4.5 フォーミュラ パラメーター

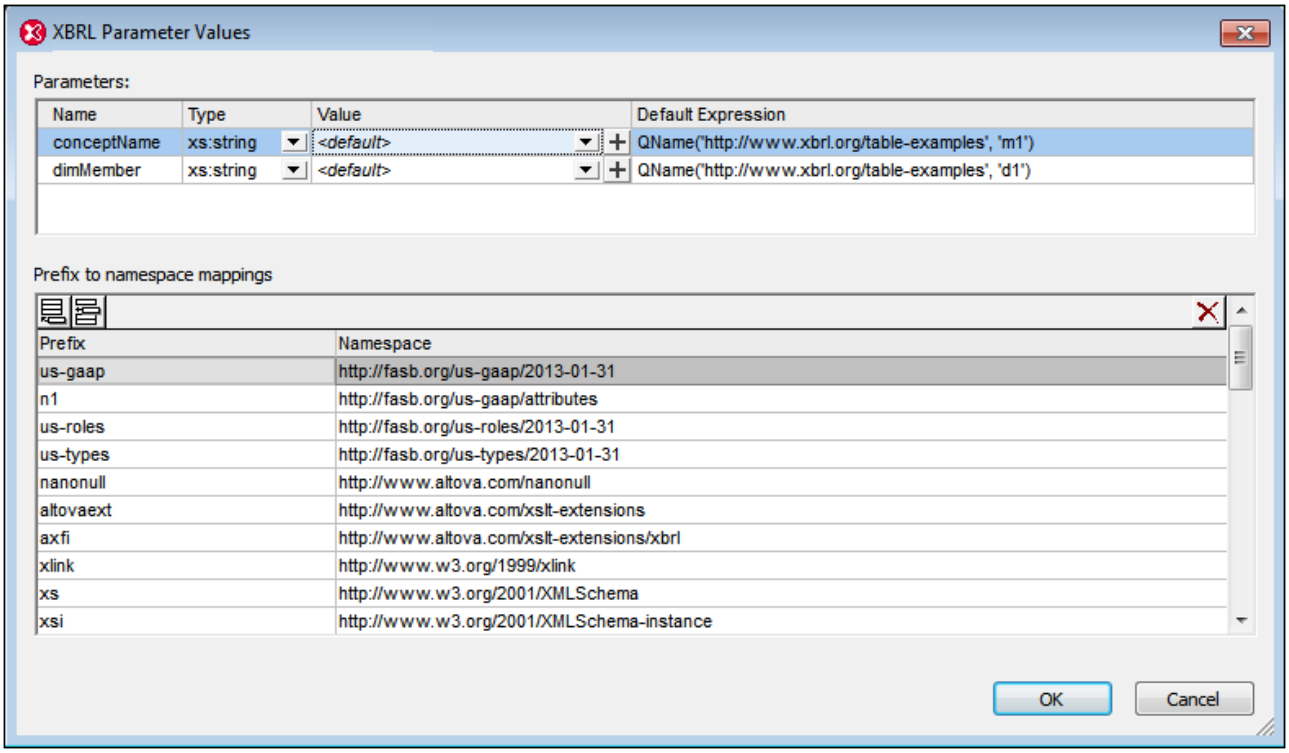
XBRL パラメーターは、フォーミュラ および テーブル定義内の XPath 式 で使用することができます。フォーミュラ パラメーターとして使用される、(フォーミュラ リンクベース内に存在する) パラメーターは、フォーミュラ タブで作成されます。(テーブル リンクベース内に存在する) テーブル パラメーターはテーブル タブ内で作成されます。フォーミュラ パラメーター および テーブル パラメーターは、ローカル または グローバル であることができます。ローカル パラメーターは、基本的には、作成時に対応するコンポーネント (フォーミュラ または テーブル) にリンクされたグローバル パラメーターです。ローカル パラメーターは、コンポーネント (フォーミュラ または テーブル) を右クリックして「**新しい パラメーターの追加**」を選択することにより、作成されますが、グローバル パラメーターは、対応するタブの空白のエリアを右クリックして「**新しい パラメーターの追加**」を選択し

て作成されます。これにより、*parameter*という名前の新しい*パラメーター*がダイアグラム内に追加されます（*下のスクリーンショット グローバルパラメーターを表示しています*）。*パラメーター*名を変更するには、名前をダブルクリックして、編集します。



すべての*パラメーター*には*必須*のフラグがあります。設定される場合、*パラメーター*は、必須で、処理アプリケーションにより値が提供される必要があります。*パラメーター*が必須ではなく、に処理アプリケーションより値が提供されていない場合、与えられた値は、プロパティ内 *Select* で与えられているXPath 式 を使用して計算される可能性があります。XPath 式を入力するには*Select* フィールド内をダブルクリックします。この値が*パラメーター*のデフォルトの値になります。オプショナルプロパティ*As* は、*パラメーター*により必要とされるデータ型を指定します。コンボボックスのドロップダウンリストからデータ型を選択します。

テーブル*パラメーター*として使用される*パラメーター*の場合、*パラメーター*のデータ型を編集し、デフォルトの値をオーバーライドする*パラメーター*値 を提供することができます。これを行うには「**XBRL | パラメーターの値**」をクリックします。そして、表示されるダイアグラム内（*下のスクリーンショット*）に*パラメーター*値を入力します。この値がデフォルトの値をオーバーライドします。テーブル*パラメーター*として使用される、*パラメーター*は、複数の値をとることができ、*パラメーター*のための追加の*パラメーター*の値を*値* 列内の「 + 」アイコンをクリックすることにより追加できます。



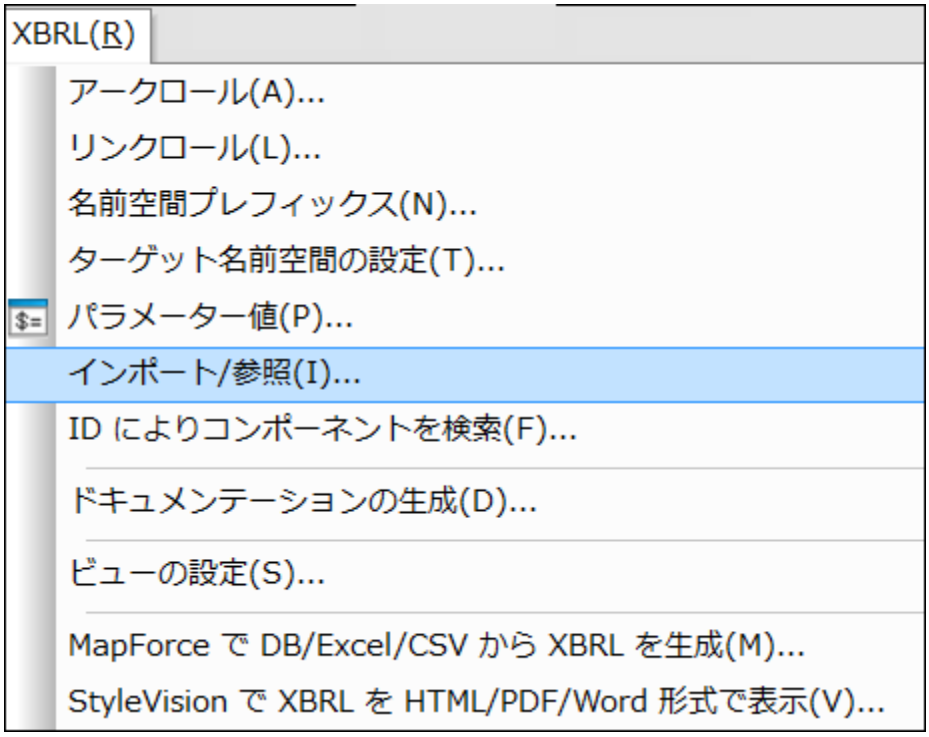
このダイアログで割り当てられたグローバル パラメーターの値は、テーブル パラメーターのみのために評価されます。フォーミュラで使用されるパラメーターの値は、このダイアログで編集されることはできません。

17.4.6 フォーミュラ コンポーネントの検索

フォーミュラ コンポーネントは、ID を使用して、または、ドキュメント内のコンポーネントの発生をナビゲートすることにより検索することができます。

ID によりフォーミュラ コンポーネントを検索する

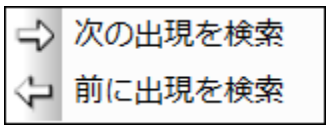
同じ型の複数のコンポーネントを含む大きなフォーミュラ リンクベースを持つタクソノミは、ID によりコンポーネントを検索する際に役に立ちます。メニューコマンド「XBRL | ID によるフォーミュラ コンポーネントの検索」 はID による検索を有効化します。



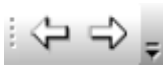
コマンドをクリックすると検索するID を問うダイアログがポップします。

コンポーネントの発生を検索する

フォーミュラコンポーネントのほとんどは、フォーミュラリンクベースダイアグラム内に複数回表示されています: (i) 適切な セクション ノードの下に直接存在する 定義、(ii) コンポーネントへのすべての参照 (リレーションシップを介して)。コンポーネントのコンテキスト メニュー内のコマンド「**次の発生を検索**」および「**前の発生を検索**」は、フォーミュラコンポーネントが参照されているすべての箇所にナビゲートします (*下のスクリーンショット*)。



これらのコマンドは、をツール バー アイコン介してアクセスすることができます。(*下のスクリーンショット*).



コンポーネントの定義に達成すると、その効果が表示されます。

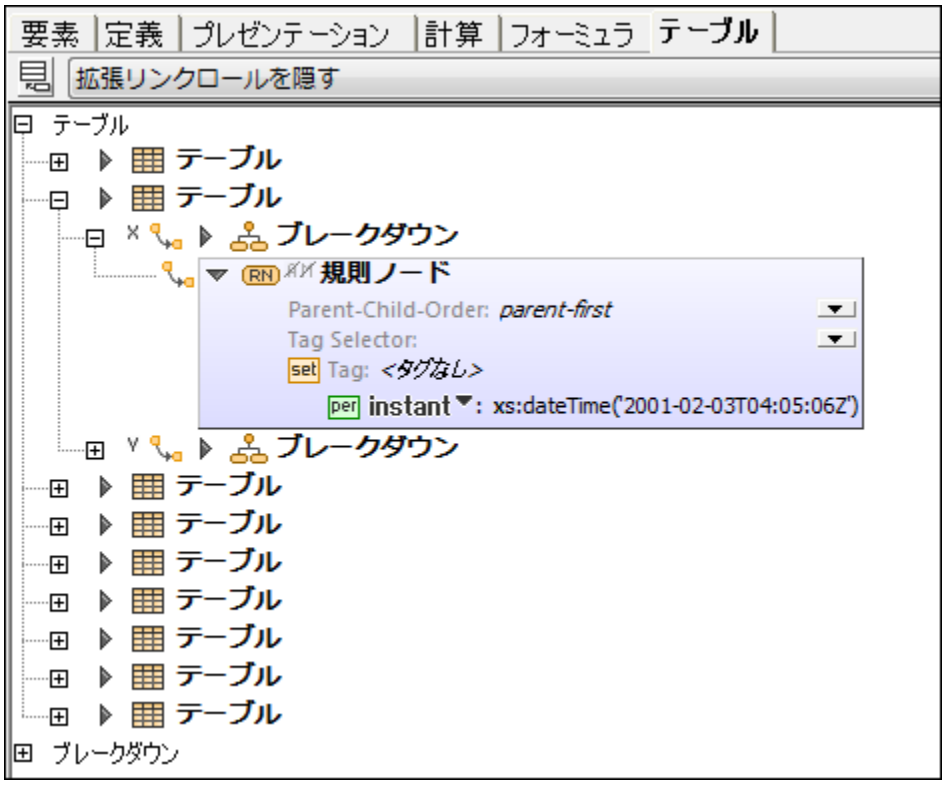
17.5 XBRL テーブル定義エディター

XBRL 仕様は*リンクベース*の表示を補完する*テーブルリンクベース*を提供します。テーブルは、XBRL *タクソノミ*内で定義される*コンセプト*のビューを定義する他の方法を与えます。*プレゼンテーションリンクベース*がするように、コンセプトを階層構造で表示するのではなく、複数の軸を使用してテーブルを定義します。軸のコンポーネントは、個別のアイテムに制限されておらず、ディメンション、期間、単位、エンティティ、タクソノミにより表されることのできる金融ファクトとして使用することのできる他のプロパティの組み合わせとして定義されることもできます。XBRL テーブルリンクベースの構文とセマンティックの概要情報は、XBRL テーブルリンクベース概要 1.0 および2014 年 3 月 18 日 版テーブルリンクベース1.0 勧告により見つけることができます。

XMLSpy は、2014 年 3 月 18 日 版テーブルリンクベース1.0 勧告に従い、名前空間 http://xbrl.org/2014/テーブル を使用します

標準のXBRL リンクベース（プレゼンテーション、計算、定義）は、標準の拡張されたリンク内のロケーターと標準のアークを介して、コンセプト間のリレーションを定義しますが、テーブルリンクベースは、コンポーネント（テーブル、ブレークダウン、定義ノード、など）を含み、ジェネリックな 拡張されたリンク内のリソース および ジェネリックな アークを介してその関係を定義します。テーブルリンクベース 仕様は、3 つのモデルのシーケンスを定義し、各モデルの次のモデルへの変換を処理します。3 つの モデル 定義 モデル、構造モデル および レイアウト（または レンダリング）モデル。定義モデルは、テーブルリンクベースのセマンティックコンテンツのモデルです。 テーブルは、軸により定義されます。また、軸定義は、代わりに定義ノードのツリーにより作成されます。

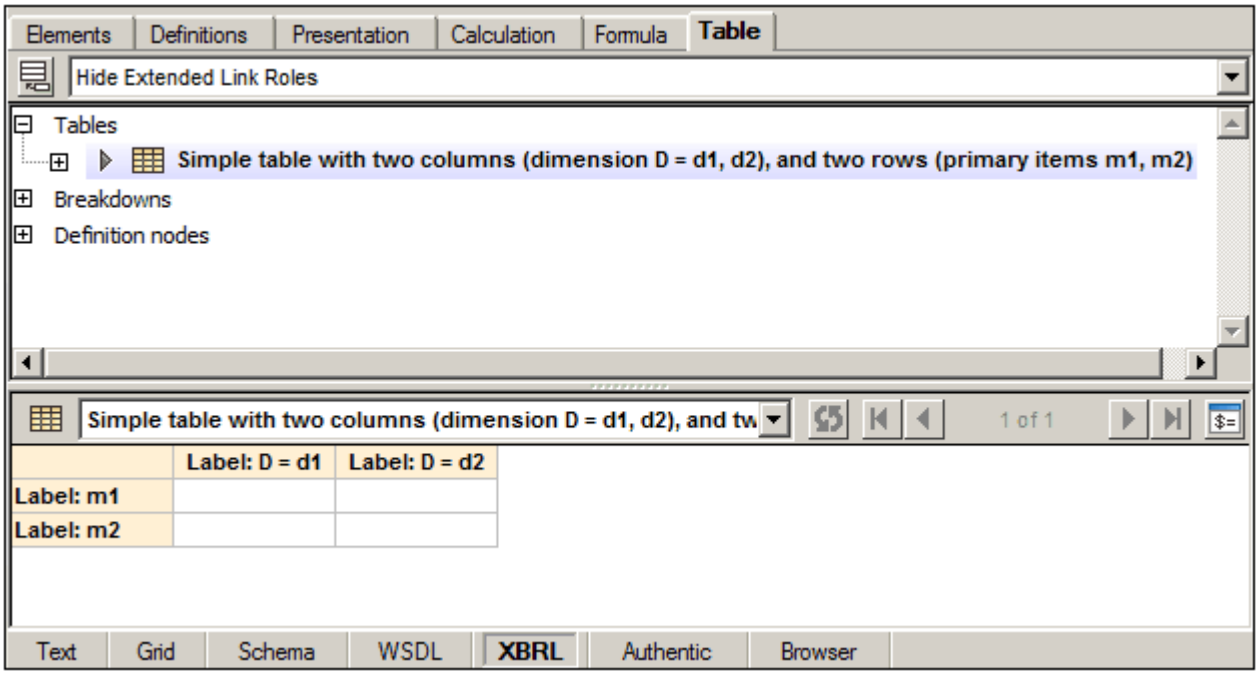
XMLSpy のXBRL テーブル定義 エディターは、アプリケーションのXBRL *タクソノミエディター*の一部として実装されます。XBRL ビューの**テーブル タブ** で使用することができます（*下のスクリーンショット参照*）。



テーブル タブは、概要 入力ヘルパーと詳細入力ヘルパーと共に、テーブル定義を編集するために使用されます。詳細入力ヘルパーは、テーブルコンポーネントのプロパティとコンテンツを編集するために使用されますが、テーブル概要 入力ヘルパーは、XBRL テーブルのためのデフォルトのリンクベースを設定するために使用されます（テーブル定義が保存されるファイルはデフォルトで保存されます）。 タブ自身がテーブル定義の直接の編集を有効化することができます。

XBRL テーブルレイアウトプレビュー

テーブル定義のレイアウトをプレビューするために、XBRL タクソノミエディターは、XBRL ビューのテーブルタブ内にXBRL テーブルレイアウトプレビューペインを与えます（*下のスクリーンショット参照*）。ダイアグラム内で、テーブルまたはテーブルコンポーネントが選択されると、テーブルのプレビューがダイアグラム下のテーブルレイアウトプレビューペインに表示されます（*下のスクリーンショット参照*）。代わりに、プレビューペインのコンボボックスのドロップダウンリストから、テーブルを選択することができます。これは、テーブルリンクベース内のテーブルのリストです。



プレビューの機能の詳細に関しては、次のセクションを参照してください：

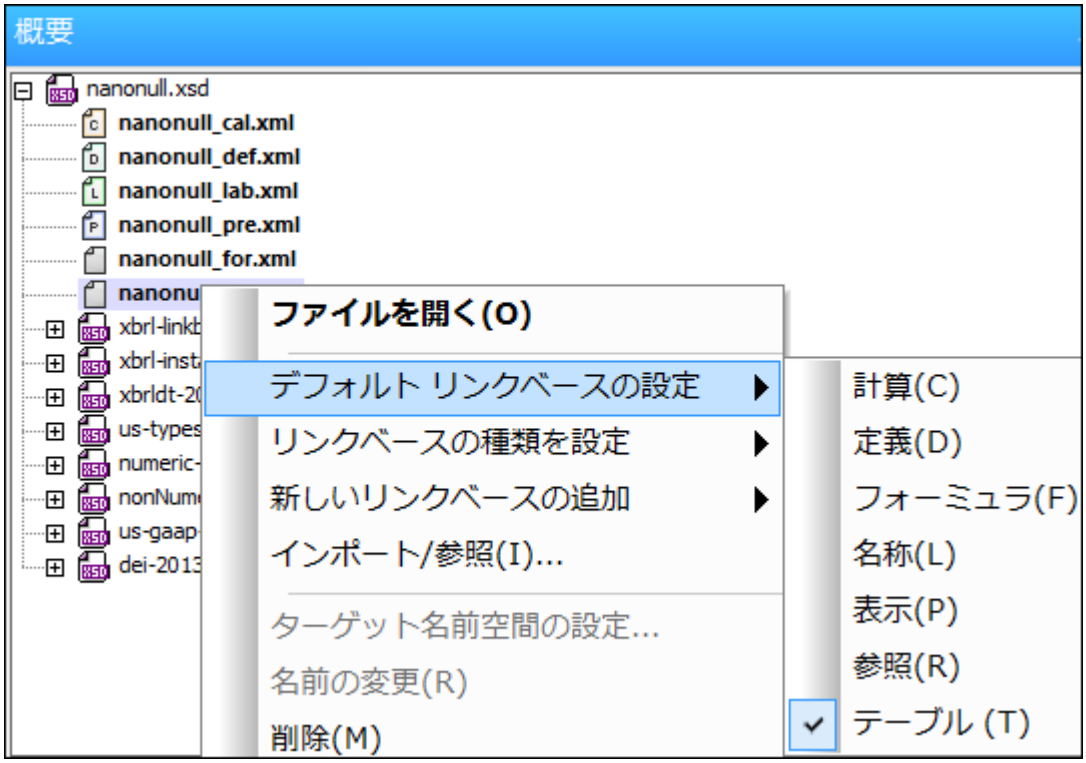
- テーブル構造 794
- テーブルパラメーター 814
- テーブルレイアウトプレビュー 817

17.5.1 テーブル リンクベース およびリンクロール

標準のXBRL リンクベース（定義、プレゼンテーション、計算）は、標準の拡張されたリンク内のロケーターと標準のアークを介してコンセプト間のルーションシップを定義しますが、テーブルリンクベースは、テーブルコンポーネント（テーブル、ブレークダウン、定義ノードなど）およびそのルーションシップを定義します。これらの定義は、ジェネリックな 拡張されたリンク内のリソースおよびジェネリックな アークを介して 指定されることができます。

テーブルリンクベースの追加

概要入力ヘルパー内（*下のスクリーンショット*）のタクソノミファイルまたは既存のリンクベースを右クリックして、「**新規リンクベースの追加 | テーブル**」を選択します。追加されたリンクベースは、デフォルトテーブルリンクベース ファイルとなります。デフォルトテーブルリンクベース ファイルは、タクソノミファイルが保存される際に保存される新しいテーブル定義が入るファイルです。他のテーブルリンクベース ファイルをデフォルトテーブルリンクベースにするには、右クリックして「**デフォルトリンクベースの設定 | テーブル**」を選択します（*下のスクリーンショット参照*）。



デフォルトリンクベースは、太字で表示され、変更されているが保存されていないリンクベースは、アスタリスクと共にマークされています。

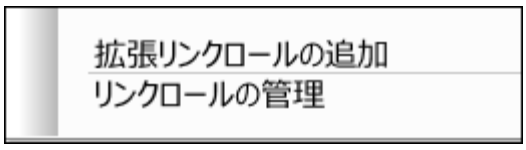
テーブルリンクベースは、テーブルタブ内に表示されます。

メモ テーブルリンクベースが存在しない場合にテーブルコンポーネントがタクソノミに追加[807]されると、テーブルリンクベースは自動的に作成されます。

リンクロール

標準の拡張されたリンクのケースと同様（定義、プレゼンテーション、計算のための）ジェネリックなリンクは、ネットワークを区切る同じタイプのリレーションシップを区切る拡張されたリンクロール値を定義する必要があります。同じリンクロールを持つすべてのジェネリックな拡張されたリンクは、*異なるリンクベース ファイルにそんざいする場合*でも、リンクロールノード テーブルタブ内のダイアグラムにの下に結合されます。

ジェネリックな リンクロールは、背景エリアのダイアグラム コンテキスト メニューを介して作成することができます（*下のスクリーンショット*）。しかしながら、このコンテキスト メニューは、テーブル タブの表示 オプション コンボボックスが、*すべての拡張されたリンクロールの表示*に切り替えられている場合のみ表示されます。場合のみ表示されます。



このメニューは、 ツール バーアイコン、*拡張リンクの追加/リンクロールの管理*、でも使用することができます。リレーションシップ ネットワークは、テーブルリンクベースにとってそれほど重要ではなく、テーブル タブのデフォルトビューは、ンクロールを非表示にする、*拡張リンクロールを隠す*です。代わりに、リンクロール無しのテーブルコンポーネントを表示します。

拡張されたリンクロールが作成された際にデフォルトテーブルリンクベースファイルが存在しない場合、デフォルトテーブルリンクベースファイルは自動的に作成されます。リンクロールの作成時にデフォルトリンクベースファイル内にリンクロールが存在しない場合、デフォルトリンクベースファイル内にリンクロールは自動的に作成されます。

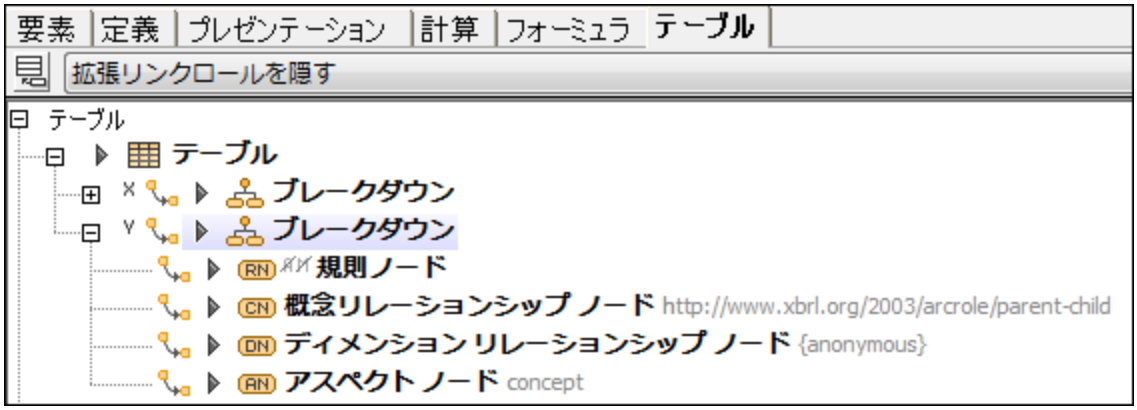
17.5.2 テーブル構造

テーブル定義内のテーブルの構造がテーブルの軸（X、Y、Z）により定義されている場合、それぞれ1つまたは複数のブレークダウンコンポーネントに対応します（*下のスクリーンショット参照*）。

- X とY 軸は、それぞれ生成されたテーブルの列と行に対応します。 X とY 軸[795]セクションで説明されています。
- Z 軸が定義されている場合、個別のテーブルとして表示されています。Z 軸[805]のセクションを参照してください。
- 各ブレークダウンコンポーネントは、複数のテーブル定義ノードを含むことができます（*下のスクリーンショット参照*）。これらは、定義ノードの異なる型です。

  - ルールノード（RN 下のスクリーンショットのアイコン）
  - コンセプトリレーションシップノード（CN）
  - ディメンションリレーションシップノード（DN）および
  - アスペクトノード（AN）。

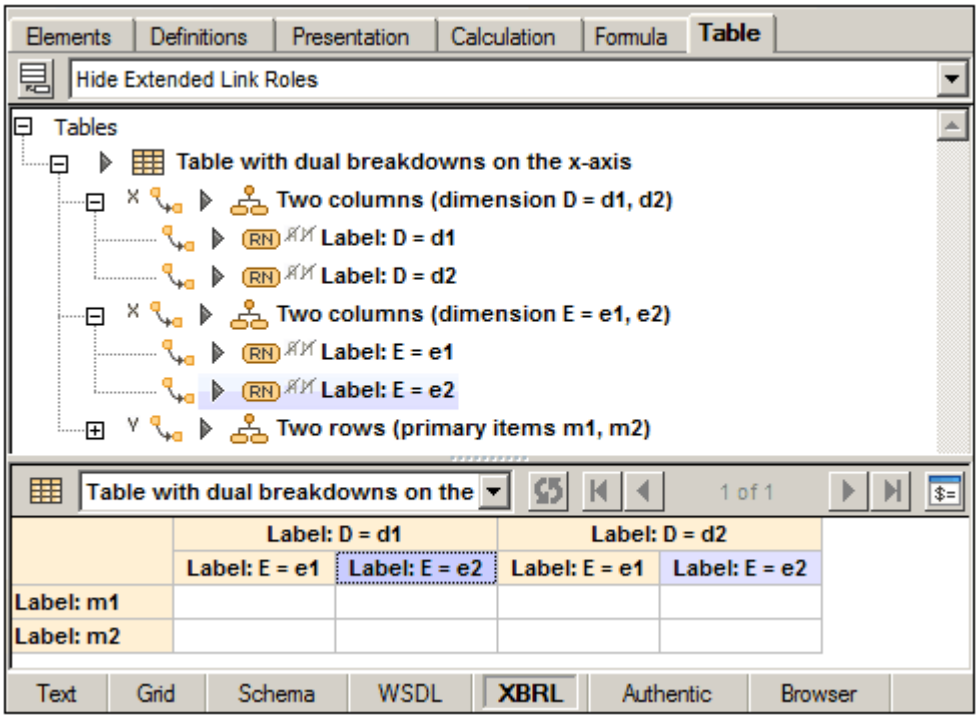
  定義ノードの構造プロパティの詳細に関しては、定義ノード[797]のセクションを参照してください。



複数のブレークダウンのためのプロジェクション

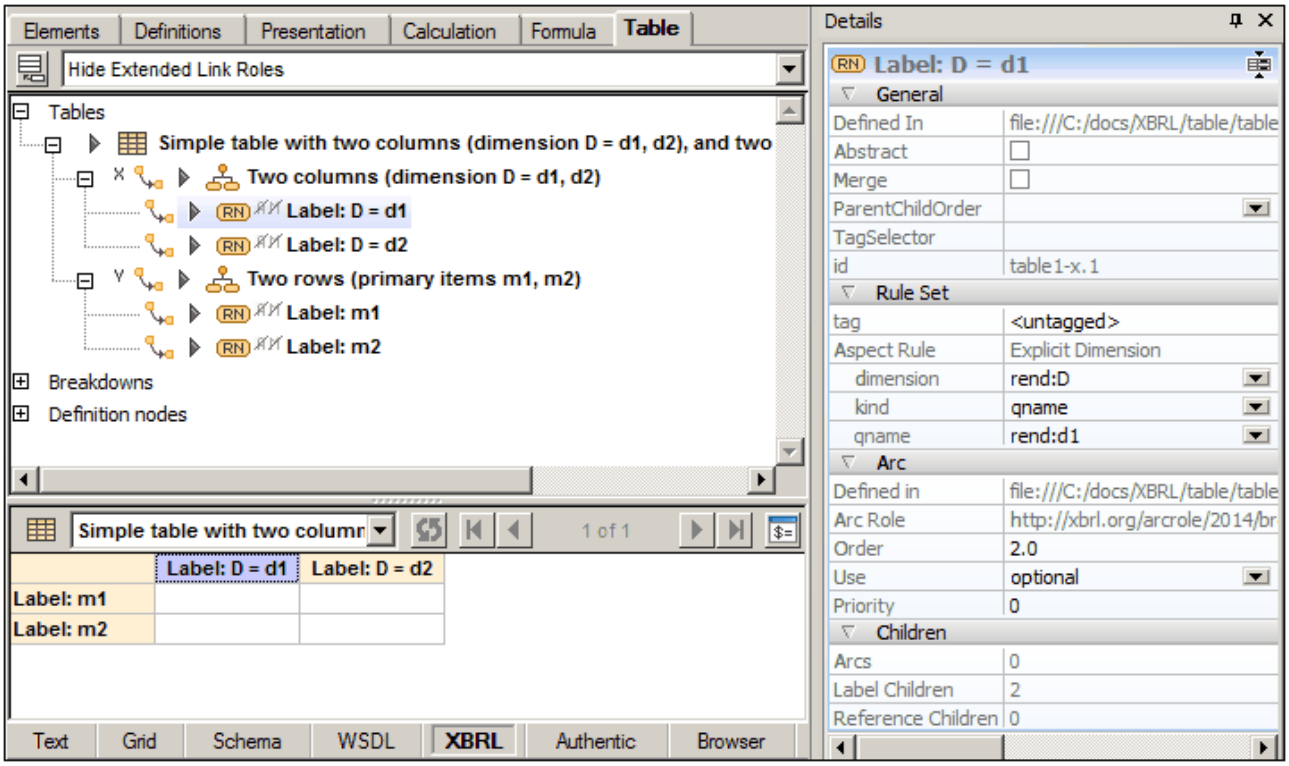
複数の独立したブレークダウンは、単一のテーブル軸に関連している可能性があります。複数のブレークダウンを単一の"効果的な"ブレークダウンに組み合わせるためのメカニズムはプロジェクションと呼ばれます。単一の軸のための複数のブレークダウンの相対的な優先順序は、各ブレークダウンの `@order` 属性により決定されます。ブレークダウンはツリーとしてビジュアル化されます。最初のブレークダウンの各リーフのため、2 番目のブレークダウン全体が添付され、再帰的に行われます。

下のスクリーンショットでは、例えば X 軸のために2 つのブレークダウンがあります: `dimension D` は、`dimension E` より高い優先順位でオーダーされます。ですから、`dimension D` (`d1` および `d2`) の各リーフには、`dimension E` のツリー全体が添付されます。X 軸は、列を生成するため、これらのブレークダウンはテーブルの列構造のためのプロジェクションを作成します。下のスクリーンショットのテーブルレイアウトプレビューを参照してください。



17.5.2.1 X と Y 軸

X と Y 軸 は、それぞれテーブルの列と行を定義します。それぞれの軸で、1 つまたは複数の階層的ブレークダウンが定義されます。(*下のスクリーンショット参照*)。ブレークダウン 対応する 単一の 軸は、単一の“効果的な” ブレークダウンに解決されます。軸のためのブレークダウンが 1 つしか存在しない場合、このブレークダウンは、効果的な ブレークダウンとなります。軸のために 複数のブレークダウンが定義されている場合、解決 メソッドについては、さらに詳しく複数のブレークダウンのためのプロジェクション 796で説明されています。



次の軸に関連するプロパティと編集機能に注意してください：

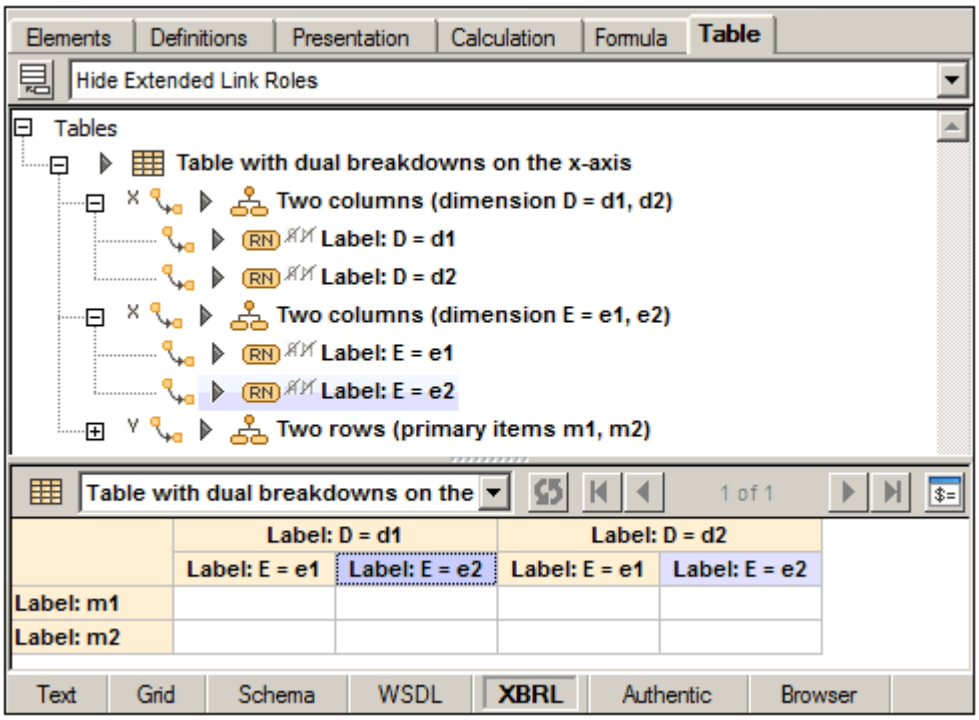
- テーブル定義内で、X 軸は、生成されたテーブルの列に対応しますが、Y 軸は、テーブルの行に対応します（*上のテーブルレイアウトプレビューのスクリーンショットを参照してください*）。
- それぞれの軸は、1 つまたは複数のブレークダウンを持つことができます（複数のブレークダウンのためのプロジェクション 796 を参照）。
- 生成されたテーブルの各セルの背景色は、オレンジ黄色です。テーブル定義内で、テーブル定義内で、セルは、軸のブレークダウン内の定義ノードに対応します。
- セルが選択されると、対応する定義ノードも選択されます。またその逆もあります。 選択されたコンポーネントのセルの背景色は、紫色です。
- コンポーネントが選択されると、詳細入力ヘルパーにプロパティが表示され、そこで編集されることができます。（*上のスクリーンショット参照*）.
- データセルには、背景色はありません。タクソノミ自身がファクトを含まないため、これらは常に空白です。
- セル制約 は、軸（存在する場合はタグセレクターを使用して） から、計算され、詳細入力ヘルパーの制約タブに表示されます。セクション内のスクリーンショット Z 軸 805 .を参照してください。

複数のブレークダウンのためのプロジェクション

複数の独立したブレークダウンは、単一のテーブル軸に関連している可能性があります。複数のブレークダウンを単一の"効果的な"ブレークダウンに組み合わせるためのメカニズムはプロジェクションと呼ばれます。単一の軸のための複数のブレークダウンの相対的な優先順序は、各ブレークダウンの `@order` 属性 により決定されます。ブレークダウンはツリーとしてビジュアル化されます。最初のブレークダウンの各リーフのため、2 番目のブレークダウン全体が添付され、再帰的に行われます。

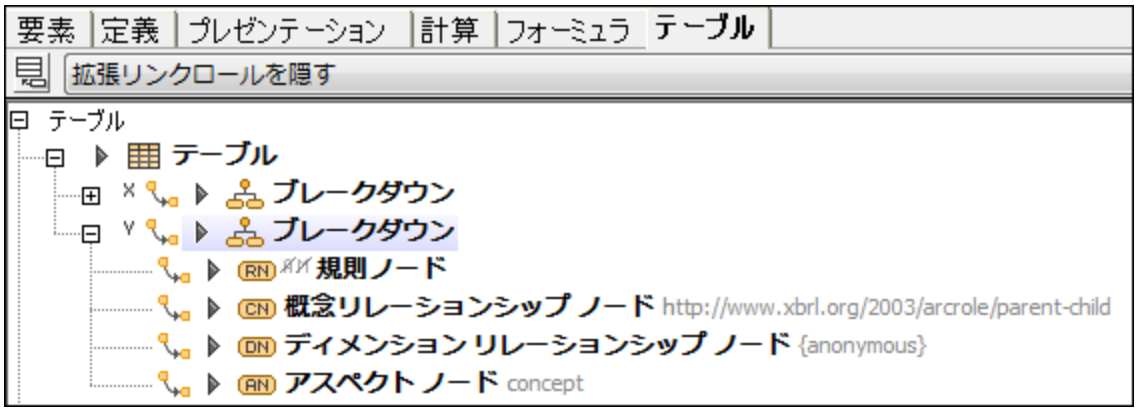
下のスクリーンショットでは、例えば X 軸のために2 つのブレークダウンがあります: **`dimension D`** は、**`dimension E`** より高い優先順位でオーダーされます。ですから、**`dimension D`** (**`d1`** および **`d2`**) の各リーフには、**`dimension E`** のツリー全体が添付されます。X 軸 は、列を生成するため、これらのブレークダウン はテーブルの列構造のためのプロジェクションを作成します。下のスクリーンショットのテーブル

レイアウトプレビューを参照してください。



17.5.2.2 定義ノード

各ブレークダウンコンポーネントは、複数のテーブル定義ノードを含むことができます（*下のスクリーンショット参照*）。



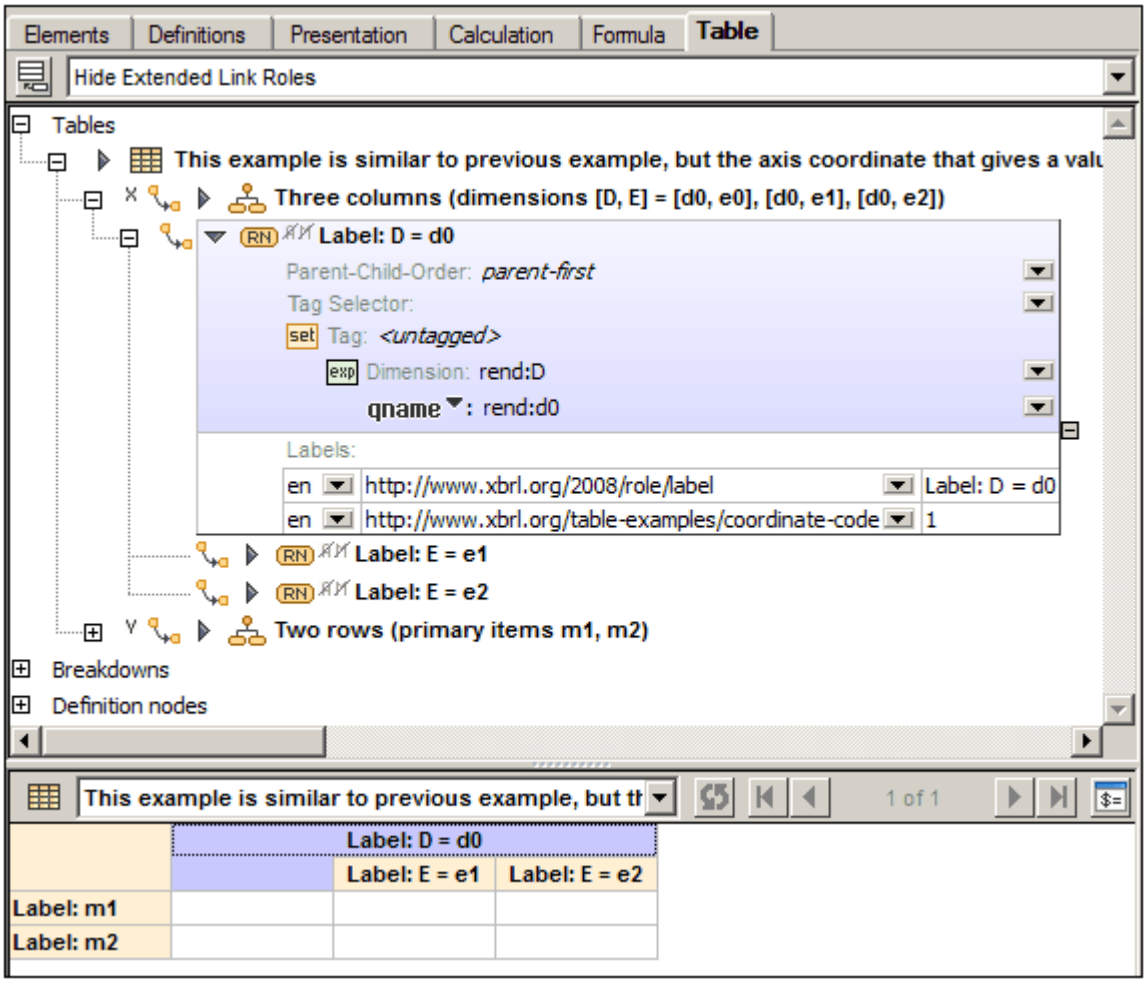
定義ノードの異なる型があります:

- ルールノード[798] (RN 上のスクリーンショット内のアイコン)
- コンセプトリレーションシップノード[802] (CN)
- ディメンションリレーションシップノード[802] (DN)

- アスペクトノード[804] (AN)

17.5.2.2.1 ルールノード

ルールノードは、1 つまたは複数のアスペクトのためのアスペクトルールを定義します。: コンセプト、期間、ユニット、エンティティ識別子、ディメンション、またはオープン コンテンツアスペクト。定義 ツリー内のコンポーネントは、ルールノードが抽象的または子のない場合、レイアウト内の 1 つのセルに対応します。その他の場合は、レイアウトは、ルールノードのプロパティ parentChildOrder の効果的な値により位置が決定される、追加ロールアップセルを含みます。

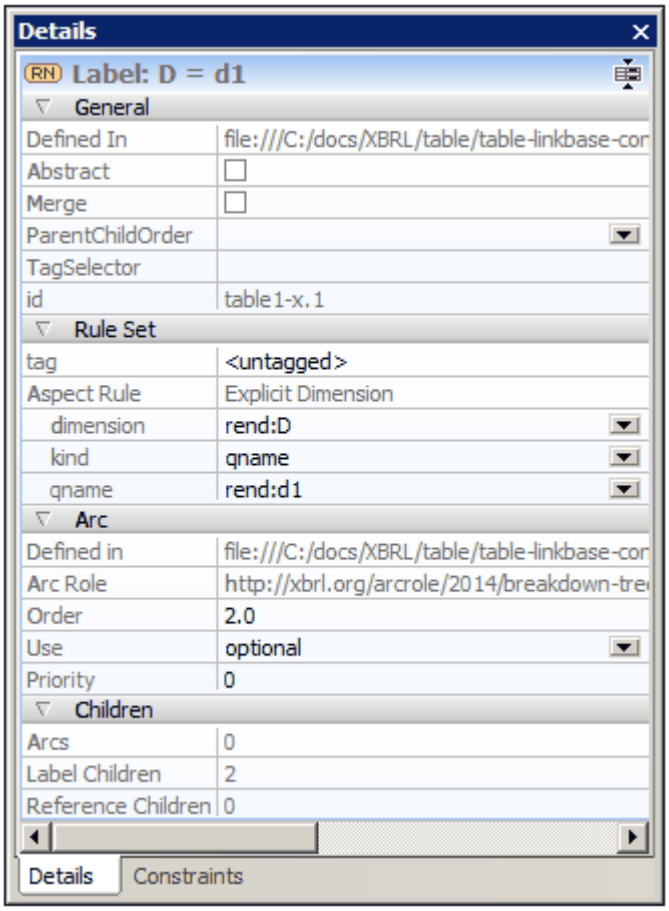


レイアウトセルのヘッダーは、ルールノードから次のように計算されます:

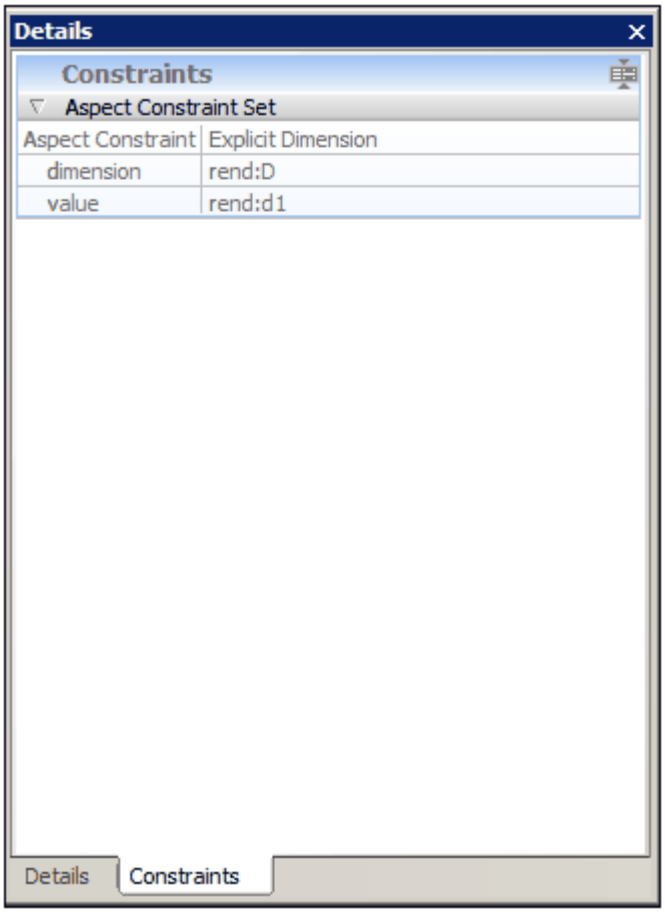
- ノードがユーザー定義 ラベルに関連付けられている場合、このラベルのテキスト が表示されます。
- 単一のアスペクト制約 を定義するラベルおよびノードが存在しない場合、(コンセプト、ディメンション、ユニット、エンティティ識別子、または期間) その値が表示されます (例えば、コンセプトの修飾名)。
- その他の場合は、静的テキスト `Rule node` が使用されます。

詳細入力ヘルパー

ノードのプロパティは、詳細入力ヘルパーの詳細タブに表示されています(左 *下のスクリーンショット*)。制約タブは、ルールノードのアスペクトルールから計算されたアスペクト制約セットの読み込み専用のビューを提供します(*右下のスクリーンショット*)。

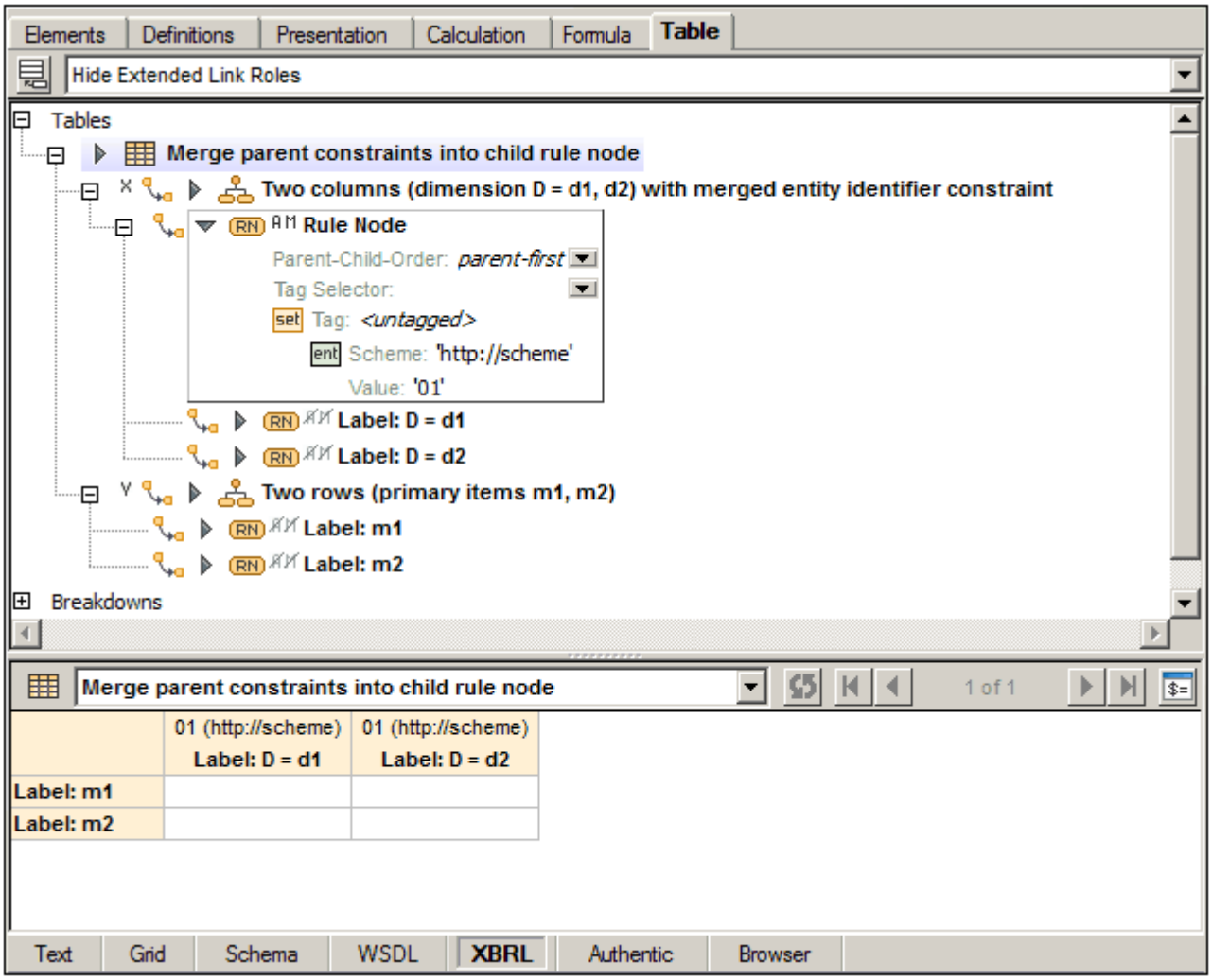


(C) 2015-2021 Altova GmbH



マージされたルールノード

マージされたルールノードは、子により作成されたすべての制約セットの制約すべてに与えられるすべての子に適用される追加プロパティを示します。(*下のスクリーンショット参照*)。



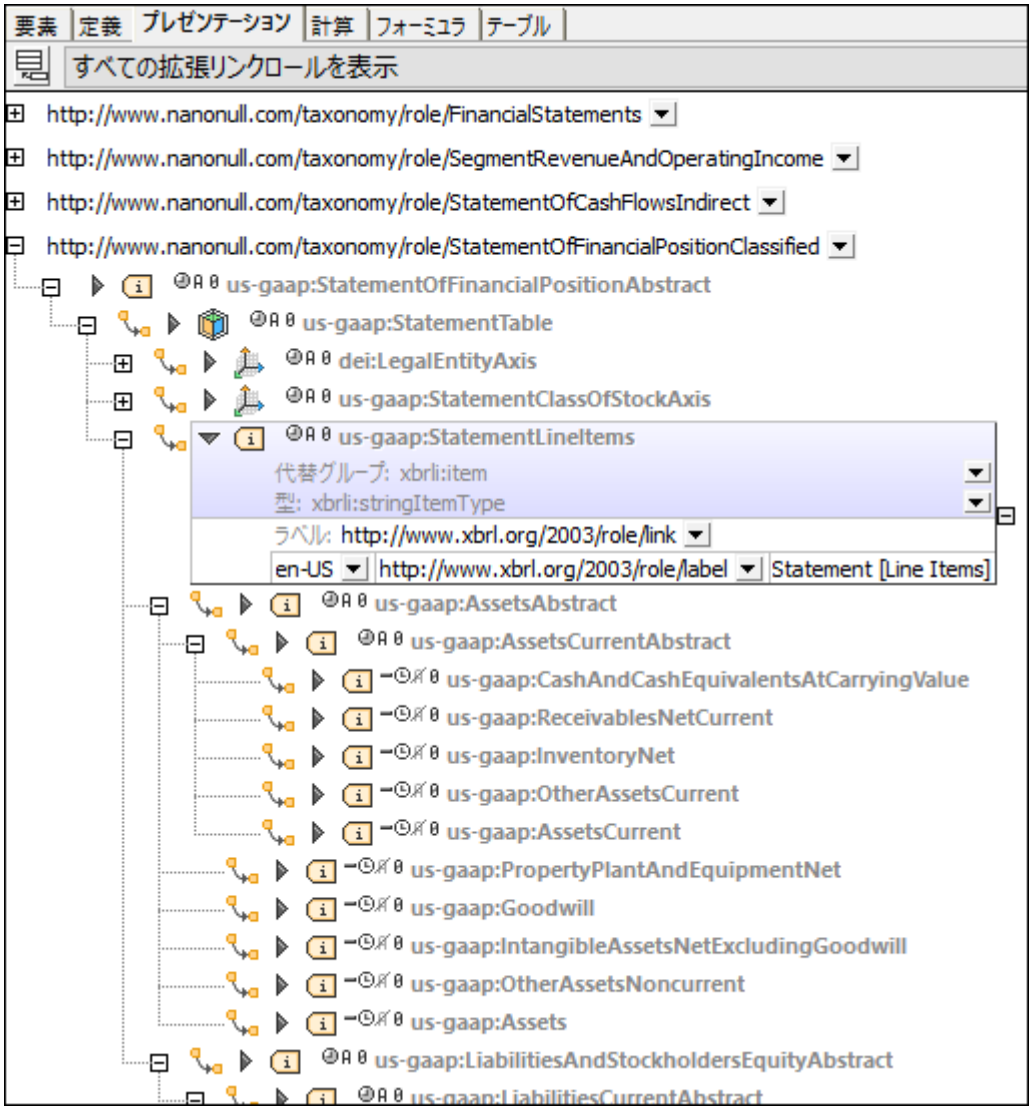
17.5.2.2.2 リレーションシップノード

コンセプト または ディメンション リレーションシップノードは、コンセプトのネットワーク または DTS 内の明確なディメンション メンバーにより定義された構造ノードのツリーになります。ですから、定義ツリー内のエンポーネントは、レイアウトのセルのブロックに対応します。

左のスクリーンショットは、をコンセプトリレーションシップノードを含むテーブル定義表示します。右のスクリーンショットは、コンセプトの対応するネットワークを表示します。この場合は、プレゼンテーション リンクベース内で定義されています。

104 - Statement - Nanonull & Consolidated Balance Sheets

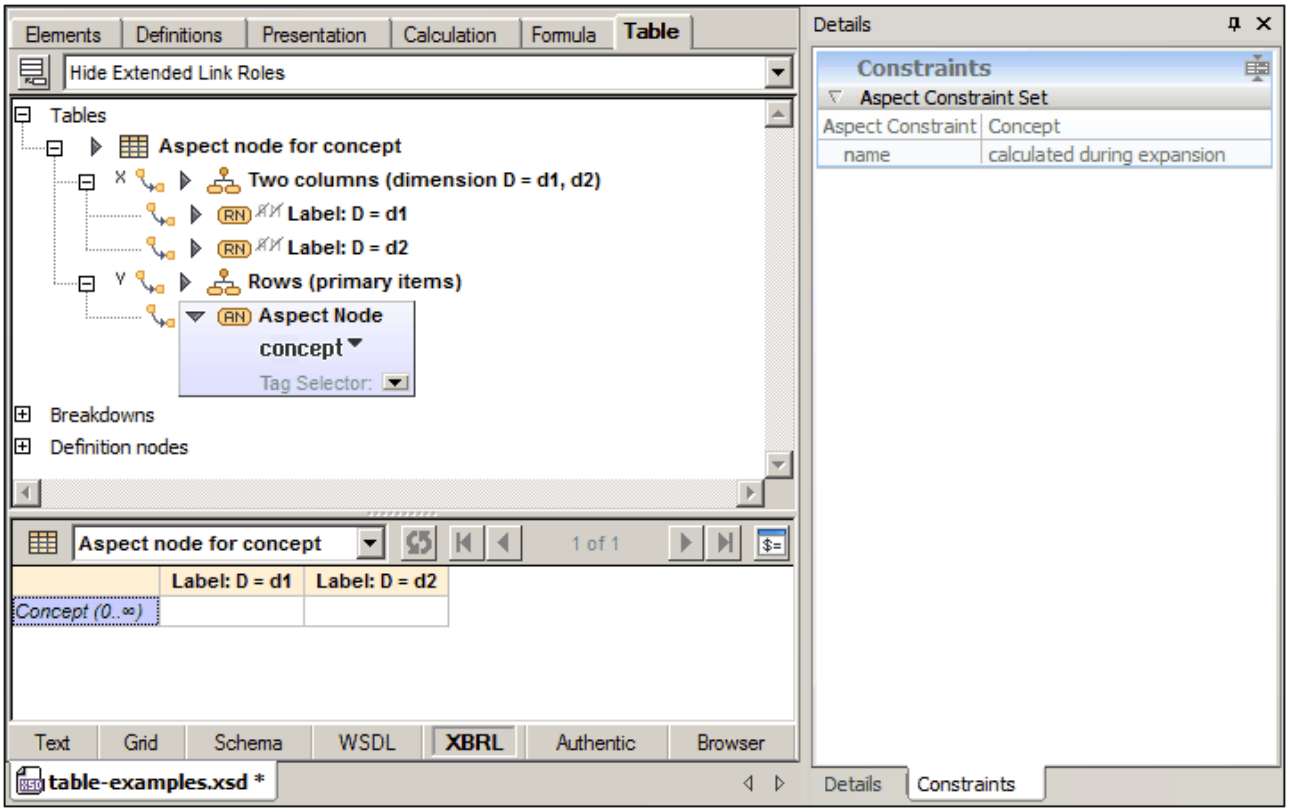
			2010-08-31	2009-11-30
ASSETS	Current Assets	Cash and cash equivalents		
		Trade and other receivables, net		
		Inventories		
		Prepaid expenses and other		
		Total current assets		
	Property and Equipment, Net			
	Goodwill			
	Other Intangibles			
	Other Assets			
	Assets, Total			
LIABILITIES AND SHAREHOLDERS' EQUITY	Current Liabilities	Short-term borrowings		
		Current portion of long-term debt		
		Accounts payable		
		Accrued liabilities and other		
		Customer deposits		
		Total current liabilities		
	Long-Term Debt			
	Other Long-Term Liabilities and Deferred Income			
	Contingencies (Note 3)			
	Shareholders' Equity	Additional paid-in capital		
		Retained earnings		
		Accumulated other comprehensive income (l...		
		Treasury stock, 39 shares at 2011 and 2010 of...		
		Total shareholders' equity		
		Common stock		
	Liabilities and Stockholders' Equity, Total			



リレーションシップノード には、1つのアスペクト制約があります: コンセプト または 明示的なディメンション。ですから、レイアウト セルのヘッダーは、(存在する場合) コンセプトのラベルまたはその修飾名です。詳細入力ヘルパーの詳細タブは、リレーションシップノードのプロパティを表示しますが、制約タブは、レイアウト内でフォーカスされたセルにより定義されたアスペクト制約 (セット) を提供します。

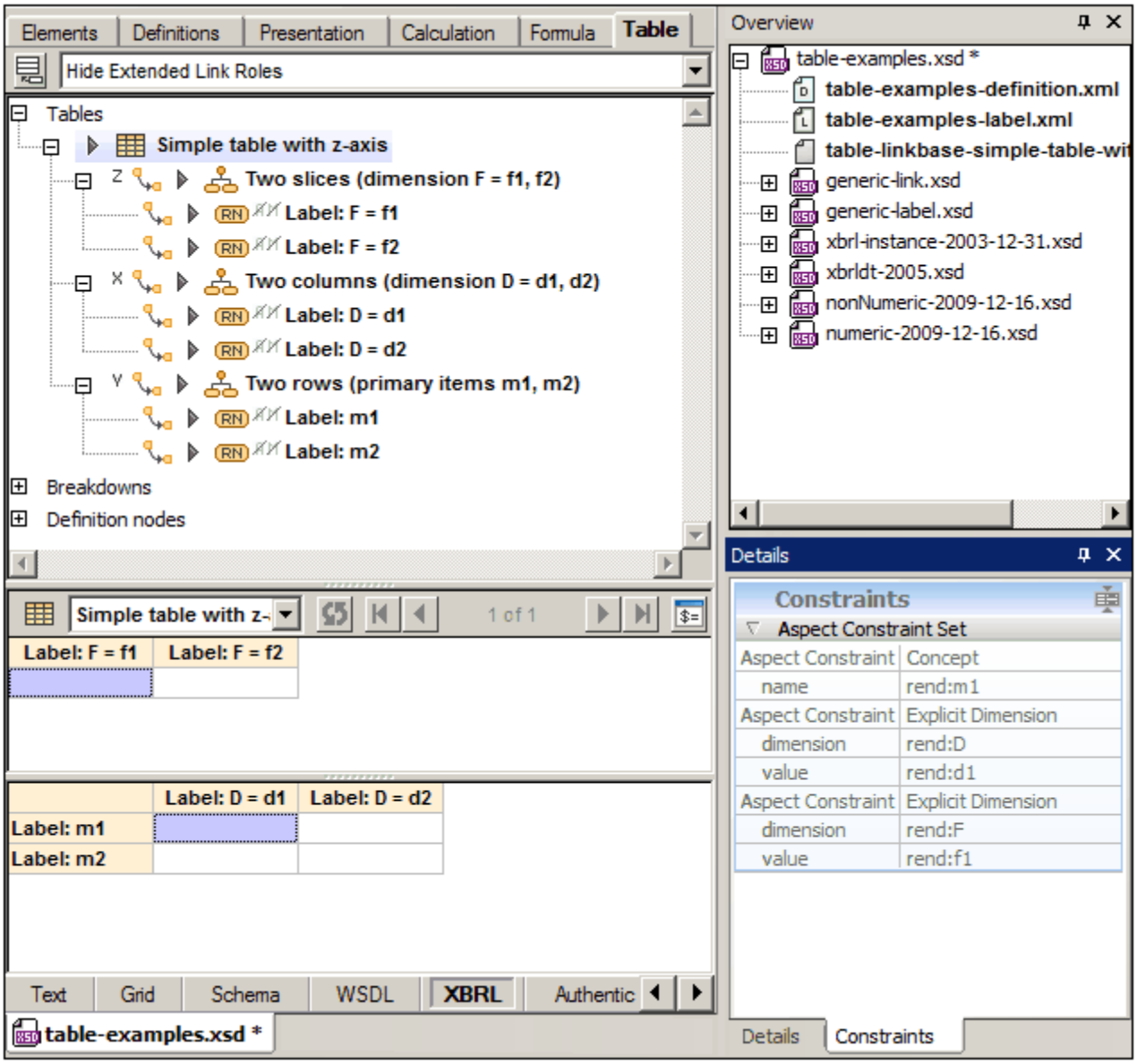
17.5.2.2.3 アスペクトノード

アスペクトノードは、単一の参加するアスペクトを直接指定するオープンな定義ノードです。レイアウトプロセス中、 XBRL インスタンス ファイルのファクト内に存在する、アスペクトノードは、参加するアスペクトの各明確な値のためのセルに拡張されます。アスペクト 値 制約は、ノードの定義とDTS により完全に決定されないためレイアウトプレビューは、プレースホルダを表示します(*下のスクリーンショット参照*)。



17.5.2.3 Z 軸

テーブル定義がZ 軸を含む場合、この軸は、複数の 2 次元のテーブルとして扱われます。テーブルレイアウトプレビュー内で、Z 軸 が個別のテーブルとしてのXY テーブル上に表示されます（*下のスクリーンショット参照*）。



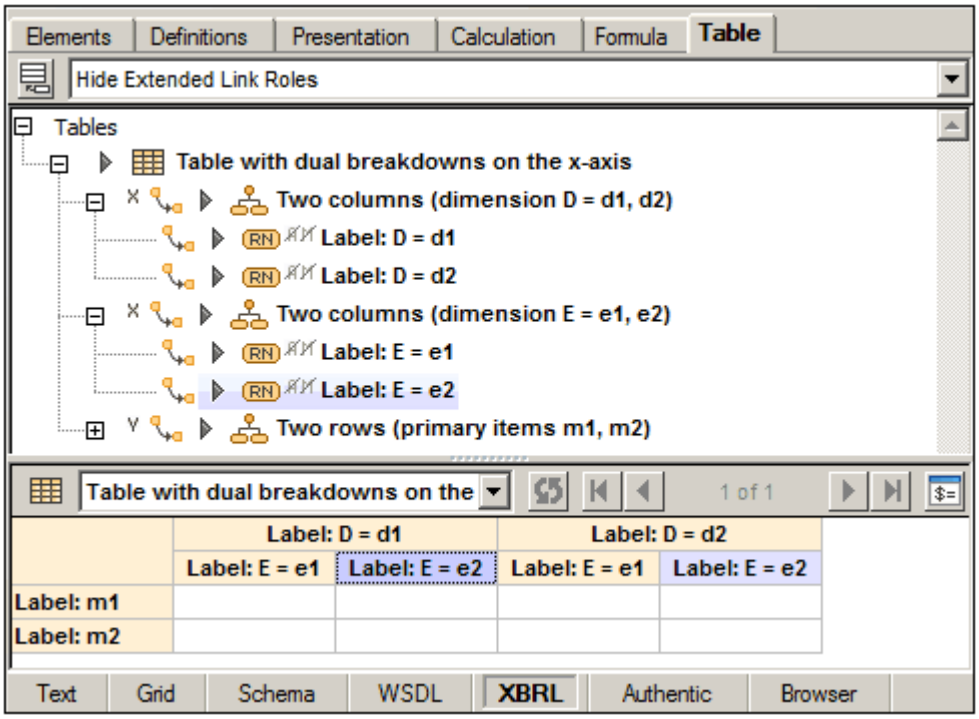
テーブル定義が、Z 軸を含むと、テーブルプレビュー内の焦点は、常に2 つの データセル(上のスクリーンショット参照) 当てられます。3本すべての 軸のための座標は、この方法により指定されます。この方法で XY テーブル内のX とY 座標およびZ テーブル内のZ の座標が決定されます。これは、上のスクリーンショットで表示されています。

軸に関連したプロパティおよび編集 機能は、X とY 軸[795]と同様です。 セル制約 は、(存在する場合はタグセレクターを使用して) 軸から計算され、詳細入力ヘルパーの制約タブ内に表示されます(上のスクリーンショット参照)。

複数のブレークダウンのためのプロジェクション

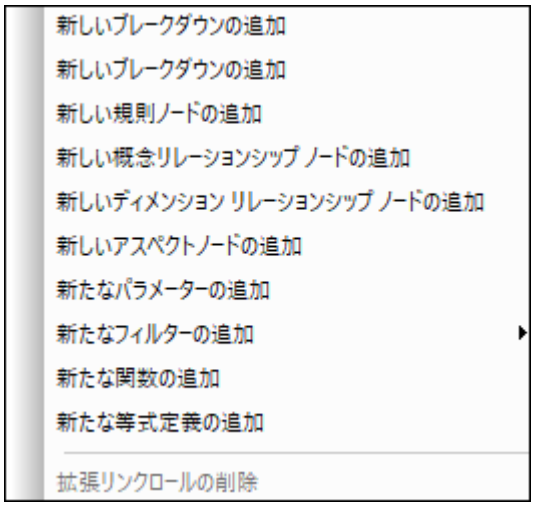
複数の独立したブレークダウンは、単一のテーブル軸に関連している可能性があります。複数のブレークダウンを単一の"効果的な" ブレークダウンに組み合わせるためのメカニズムはプロジェクションと呼ばれます。単一の軸のための複数のブレークダウンの相対的な優先順序は、各ブレークダウンの `@order` 属性 により決定されます。ブレークダウンはツリーとしてビジュアル化されます。最初のブレークダウンの各リーフのため、2 番目のブレークダウン全体が添付され、再帰的に行われます。

下のスクリーンショットでは、例えば X 軸のために2 つのブレークダウンがあります: `dimension D` は、`dimension E` より先高い優先順位でオーダーされます。ですから、`dimension D` (`d1` および `d2`) の各リーフには、`dimension E` のツリー全体が添付されます。X 軸 は、列を生成するため、これらのブレークダウン はテーブルの列構造のためのプロジェクションを作成します。下のスクリーンショット のテーブル レイアウト プレビューを参照してください。

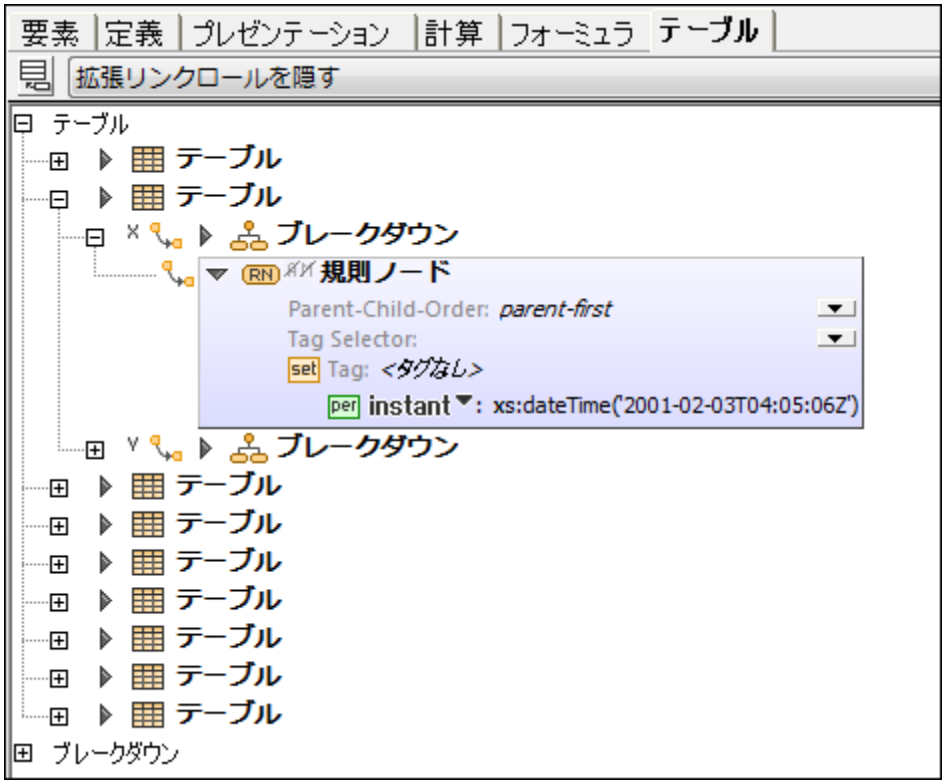


17.5.3 テーブルコンポーネント

新しいテーブルコンポーネントは、リンクロールノードのコンテキスト メニュー で作成されます（ *下のスクリーンショット*）。または、*拡張リンクロールを隠す*, に設定されたビューで、をツール バーアイコン *新しい テーブルコンポーネント* 介して作成されます。

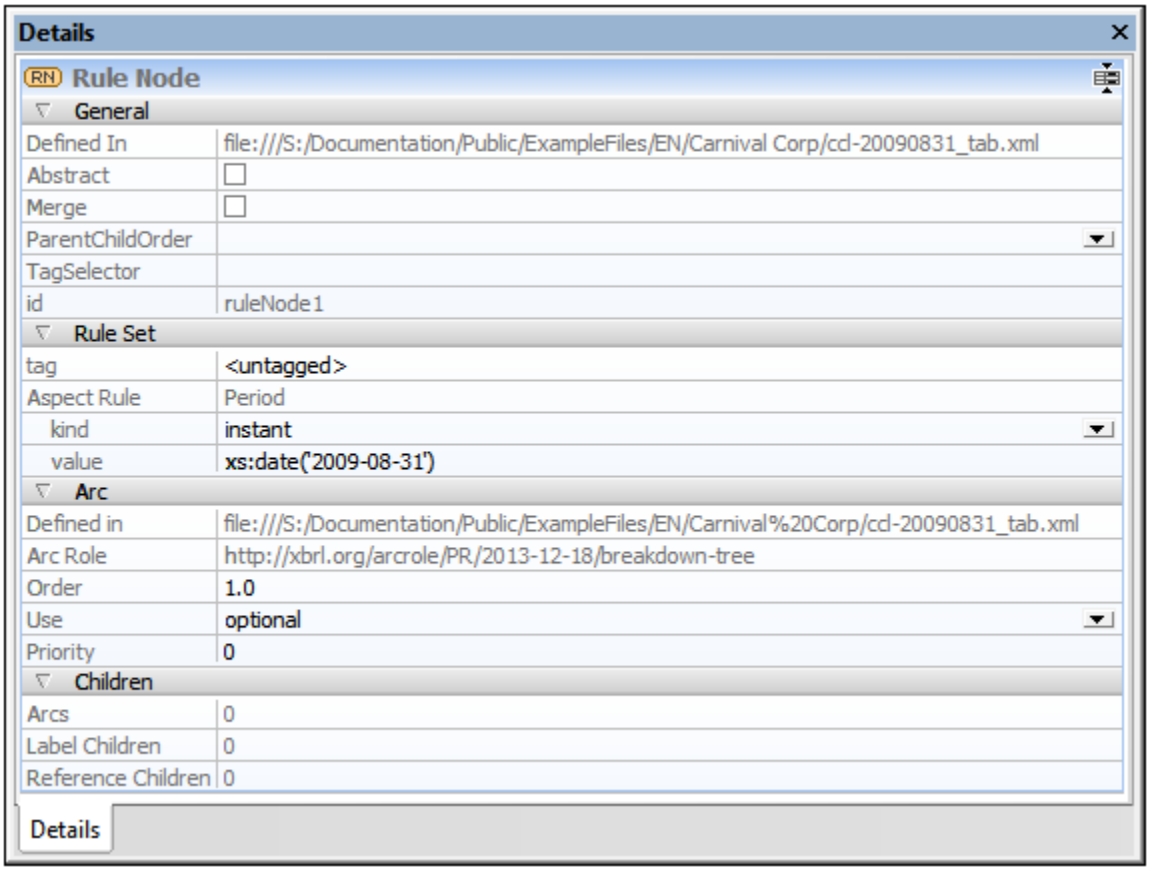


各種のコンポーネントに追加して、関係するメカニズムは、セクションのサブセクションで説明されています。テーブルコンポーネントが追加された後、テーブルタブ内のダイアグラムに表示されます（*下のスクリーンショット参照*）。



意味を明確にするために、テーブルコンポーネントは、ツリー構造内に表示されているリレーションシップ to 他のコンポーネント（アーク）に分けられています（*上のスクリーンショット参照*）。コンポーネントのプロパティとリレーションシップ（アーク）のアイコンとしてダイアグラムに、コンポーネント または アークの左横に表示されています（*上のスクリーンショット参照*）。

ルールノードの詳細入力ヘルパーは、上のスクリーンショットでハイライトされています。ノードのプロパティは、*一般* セクションの下にリストされています。プロパティのboolean の値は、チェックボックスがチェックされていると、`true` チェックが解除されていると`false` で示されています。追加セクションのは、ノードに関連する他の詳細をリストします。



詳細入力ヘルパーでアークのプロパティを確認するには、ダイアグラム内から `to`（ターゲット）コンポーネントを選択します。アークのプロパティはアークセクションにリストされます。

テーブルエディター内のコンテキストメニュー

テーブルコンポーネントのコンテキストメニューは、コンポーネントの型により異なります。 メニューアイテムは、以下のセクションに整理されています:

- リレーションの修正（サブアイテムのみのため）: *アークのオーバーライド/削除*
- コンテンツの変更（ルールノード、リレーションシップノード）: 例えば、*追加/追加 アスペクトルール*
- *ラベル参照の追加*
- 新しい子コンポーネントの作成（リレーションシップを含む）: 例えば、*新しいブレークダウンの追加*
- （リレーションシップを含む）コンポーネントの削除
- （コンポーネントの）*次の/前の発生を検索*

メモ コンテキストメニューのより作成または削除され、*たルール セット* などのコンテンツアイテムは、追加セクション内の詳細入力ヘルパーに表示されています。

17.5.3.1 テーブル

テーブルは、プロパティ parent-child-order (parent-first/children-first) を提供します。オーバーライドされないテーブル内のすべての クローズされた 定義ノードにより与えられたロールアップ のデフォルトの位置を定義します。

17.5.3.2 ブレークダウン

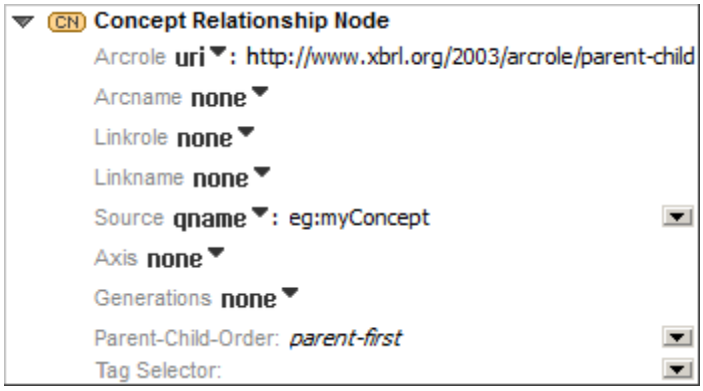
ブレークダウンは、プロパティ parent-child-order (parent-first/children-first) を提供します。ブレークダウン内のすべての閉じられた定義ノードにより与えられた、ロールアップノードのデフォルトの位置を定義し、テーブルからの継承された値をオーバーライドします。ます。

17.5.3.3 定義ノード: ルール

ルールノードは、アスペクト ルールを介して、制約を定義する閉じられた定義ノードです (フォーミュラコンポーネントを参照)。ルールノードは、ゼロまたはそれ以上のルールセットを定義します。すなわち、アスペクト ルールのセットです。各ルールセットは、タグを指定することができます。ルールセットの1つは、タグを省略することができます。このタグ付けされていない ルールセットは、常にタグ付けされたルールセットの前に表示されます。 空のタグ付けされていない ルールセットは、タグ付けされた ルールセットが存在する場合、表示されません。ルールノードは、以下の2 つのBoolean プロパティをアイコンとして提供します: abstract および merge。下のスクリーンショットは、アスペクト ルールなしのルールノードを表示します。

17.5.3.4 定義ノード: コンセプトリレーションシップ

コンセプトリレーションシップ ノードは、コンセプト XBRL 2.1 ネットワークのツリー ウォークを実行し、コンセプトを検索します。ツリー ウォークは、 ネットワーク および1 つまたは複数 リレーションシップソースにより一意に識別されます。コンセプトリレーションシップ ノードは、単一のネットワークを識別する必要があります。多くの場合、リンクロールおよびアークロールの組み合わせで、厳密にネットワークを識別することができますが、アーク名 または 拡張されたリンクの名前などの追加の情報を指定する必要があるかもしれません。

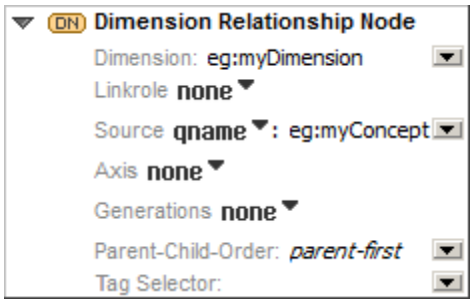



```
Arcrole:     Kind = uri | exp
Arcname:     Kind = none | qname | exp
Linkrole:    Kind = none | uri | exp
Linkname:    Kind = none | qname | exp
Source:      Kind = qname | exp
Axis:        Kind = none | value | exp
Generations: Kind = none | value | exp
```

コンセプトリレーションシップは、サブツリーを持つことができません。

17.5.3.5 定義ノード: ディメンションリレーションシップ

ディメンションリレーションシップノードは、明示的なディメンションメンバーのツリーをディメンショナルなリレーションシップセット（DRS）のツリーウォークをとして説明します。このツリーウォークは、1つまたは複数のリレーションシップソースにより一意に識別されます。



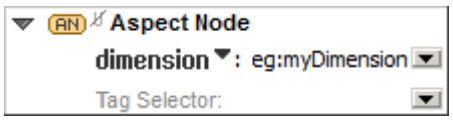

```
Linkrole:    Kind = none | uri | exp
Source:      Kind = qname | exp
Axis:        Kind = none | value | exp
Generations: Kind = none | value | exp
```

ディメンションリレーションシップノードは、サブツリーをもつことができません。

17.5.3.6 定義ノード: アスペクト

アスペクトノードは、1つのアスペクトを定義します。

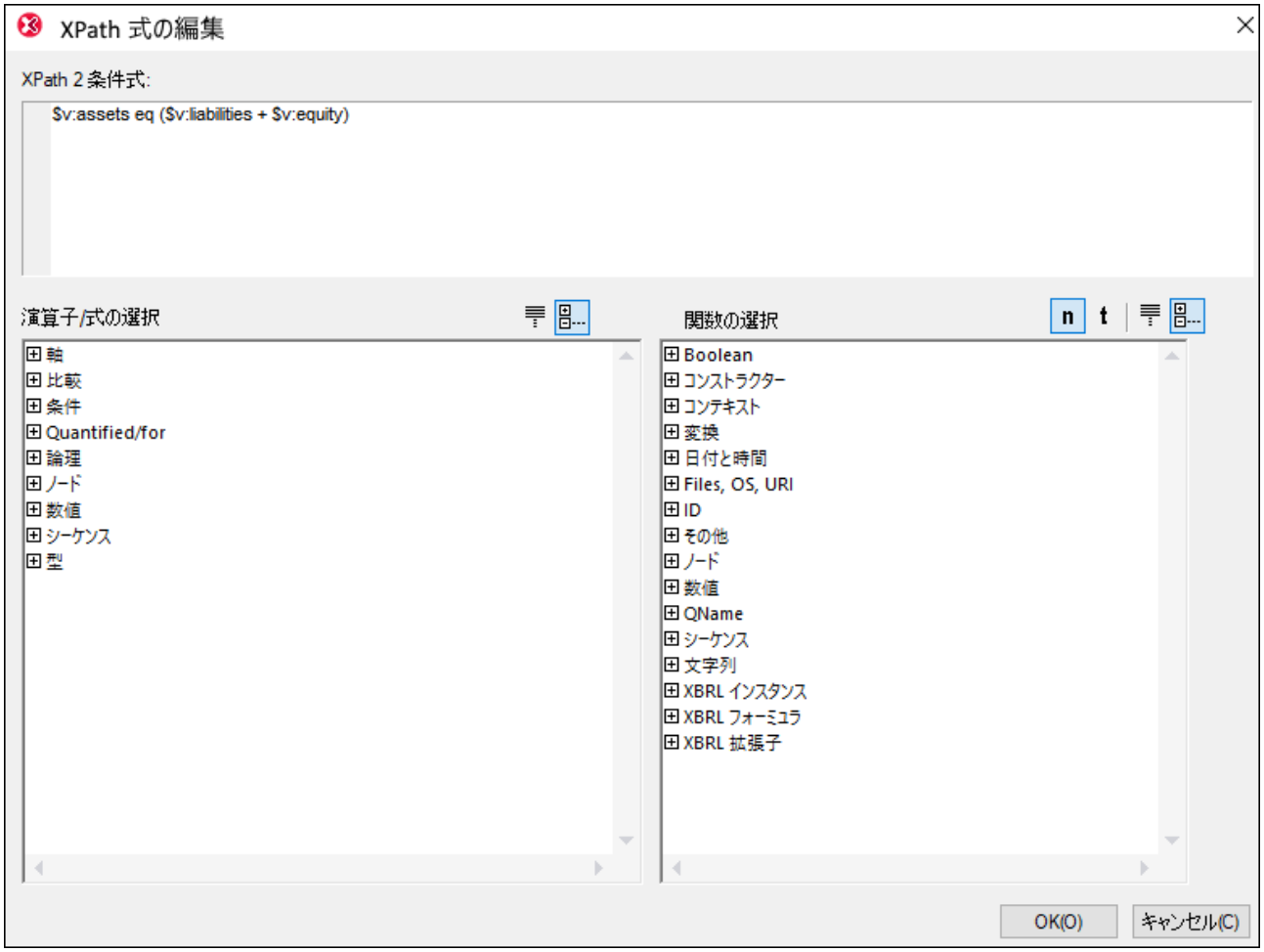
ディメンショナルなアスペクト仕様は、追加のBoolean プロパティ `include unreported value` をアイコンとしてとして与えます。



17.5.4 コンポーネントプロパティとコンテンツの編集

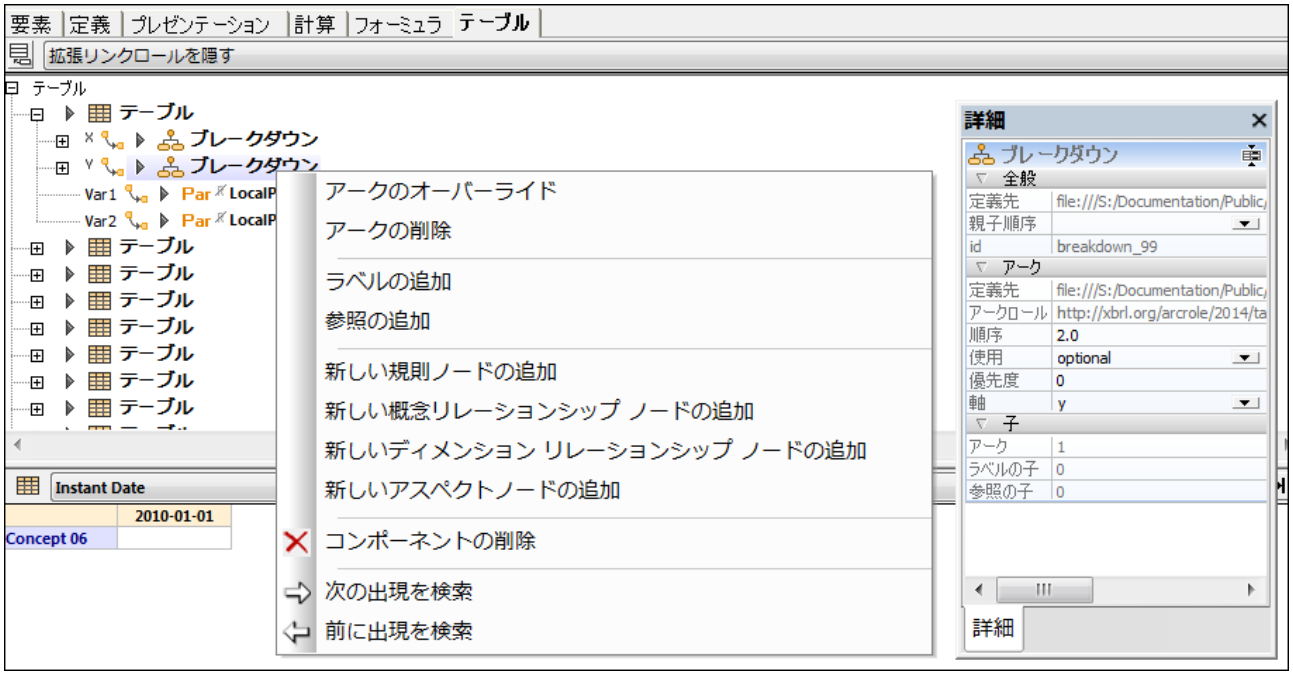
テーブルコンポーネントのプロパティは、直接ダイアグラムで、または詳細入力ヘルパーで編集することができます。

ダイアグラムで、コンポーネントが折りたたまれていると、（存在する場合は）名前 または、適切な デフォルトのプロパティの値が、コンポーネントの説明テキストの横に、灰色で表示されます。コンポーネントをダブルクリックすると、展開することができます。プロパティをダブルクリックすると、プロパティを編集モードにします。プロパティ または コンテンツが XPath 式を含む場合、XPath 式の編集がポップアップします。



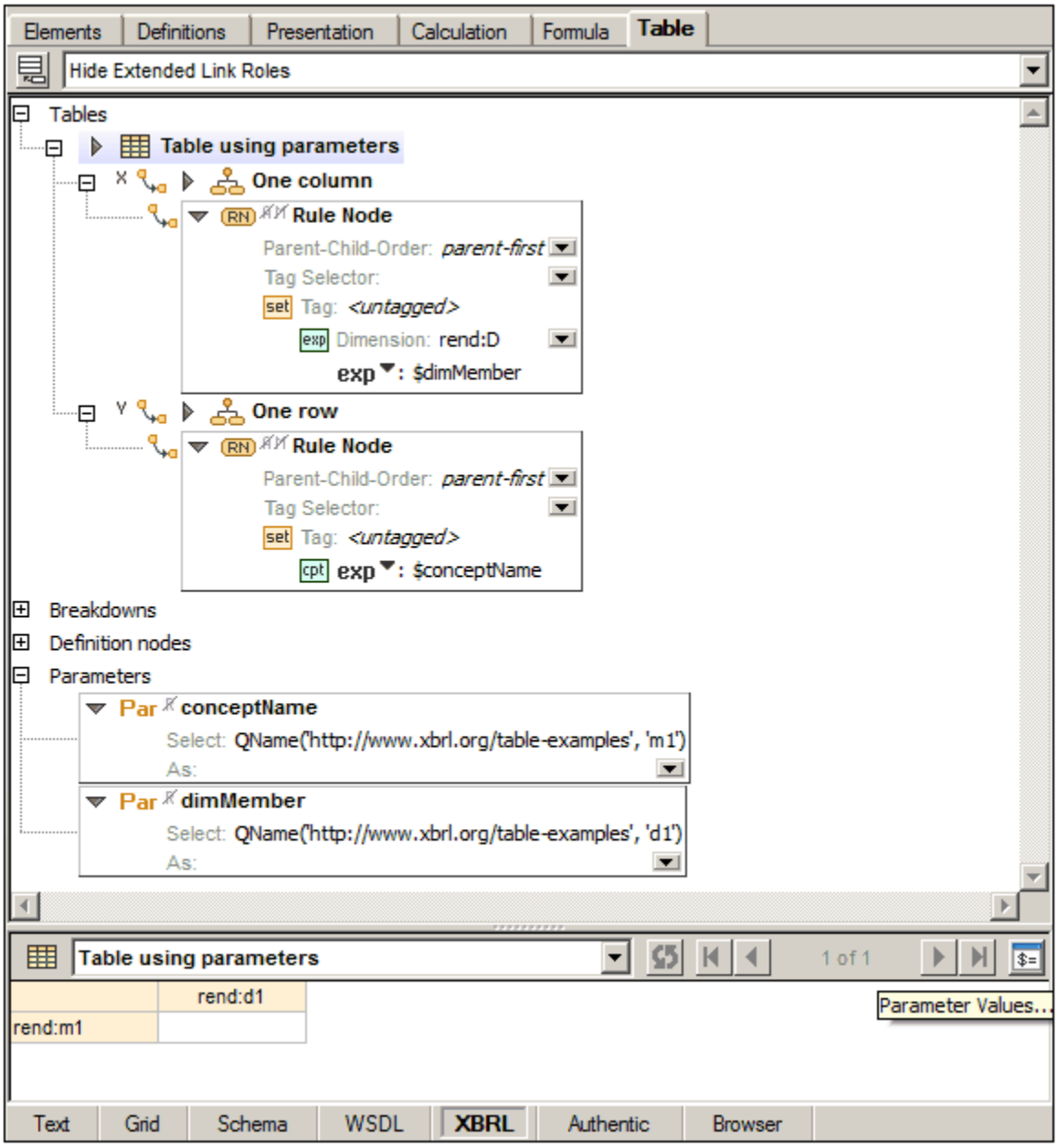
17.5.5 テーブルコンポーネントリレーションシップ

テーブルコンポーネント間のリレーションシップは、テーブルコンポーネントと他のテーブルコンポーネントにドラッグアンドドロップすることにより作成することができます。親 コンポーネントの子 の順序は、アークプロパティ `order` の値によります。この順序 は、子をドラッグアンドドロップして移動することにより変更することができます。(プロパティを含む)子コンポーネント リレーションをコピーまたは移動するために、コンポーネントをの上または下にドラッグします。



17.5.6 テーブルパラメーター

テーブルパラメーターは、テーブルの軸を定義するために使用することができます。例えば、下のスクリーンショット内で、選択されたテーブルの X-軸 は、パラメーター `$dimMember` により定義されます。Y-軸 は、パラメーター `$conceptName` により定義されます。2 つのパラメーターの定義は、テーブル定義の下の(グローバル)パラメーターリスト内に表示されます。テーブルレイアウトプレビューの下のペインは、生成されるテーブルを表示します。軸は、テーブルの行と列として作成されます。

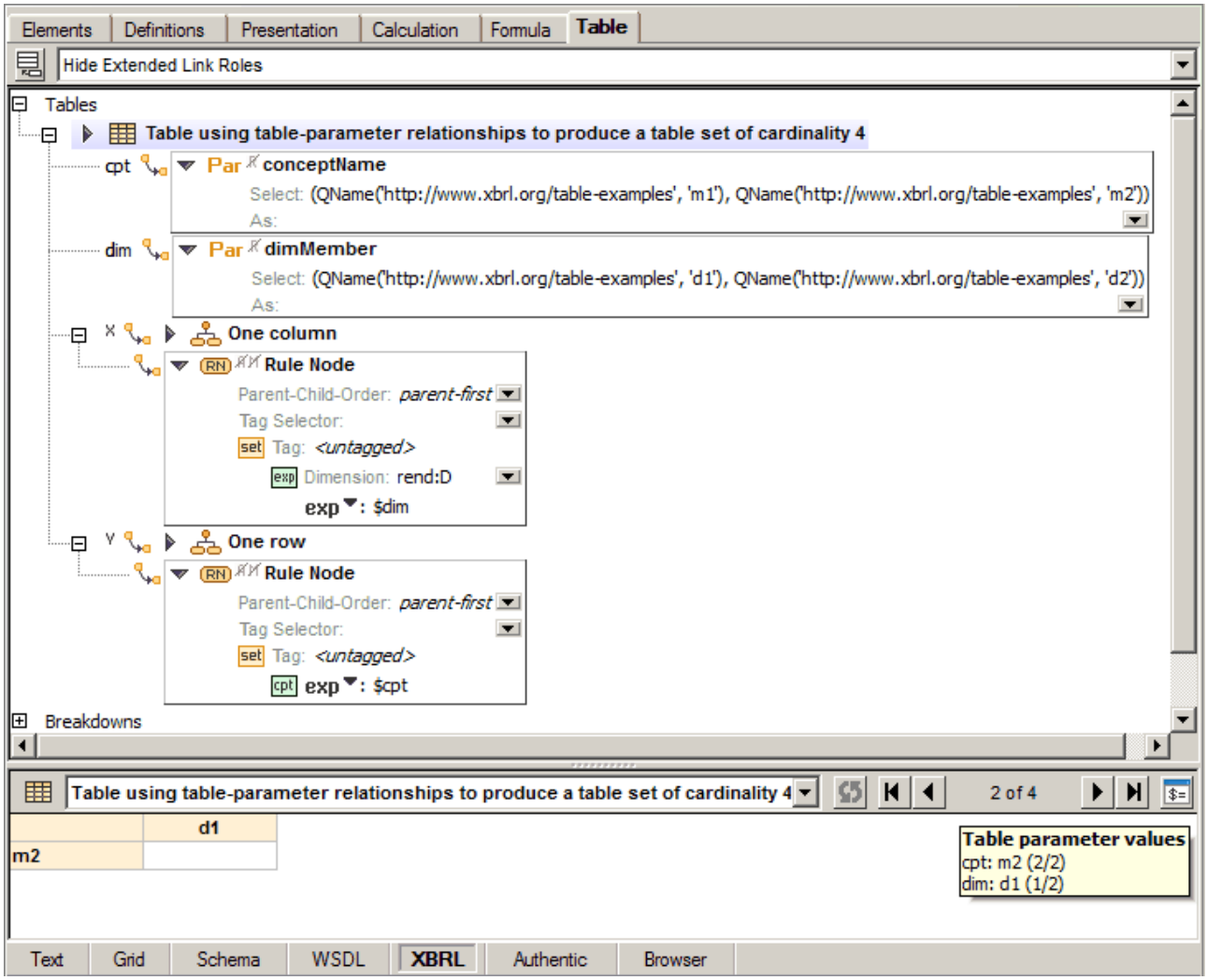


テーブル パラメーターにより単一のテーブル定義から、テーブルセットを作成する複数の関連するテーブルを作成することが許可されます。

- 単一の パラメーターが値のシーケンスを評価する場合、 テーブルセットは、結果シーケンス内の各アイテムのためのテーブルを1 つ含みます。
- 複数の パラメーターが値のシーケンスを評価する場合、テーブル セットは、評価 パラメーターにより取得され、順序付けされた、シーケンスのデカルト 製品に相対します。順序付けされたデカルト 製品は次の例で表示されています:

```
A × B = {1,2} × {3,4} = {(1,3), (1,4), (2,3), (2,4)}
B × A = {3,4} × {1,2} = {(3,1), (3,2), (4,1), (4,2)}
```

2 つのテーブル パラメーター(`conceptName` および `dimMember`) それぞれが2 つのQName のシーケンスを検証します (*Select プロパティの XPath 式のを参照してください*。)



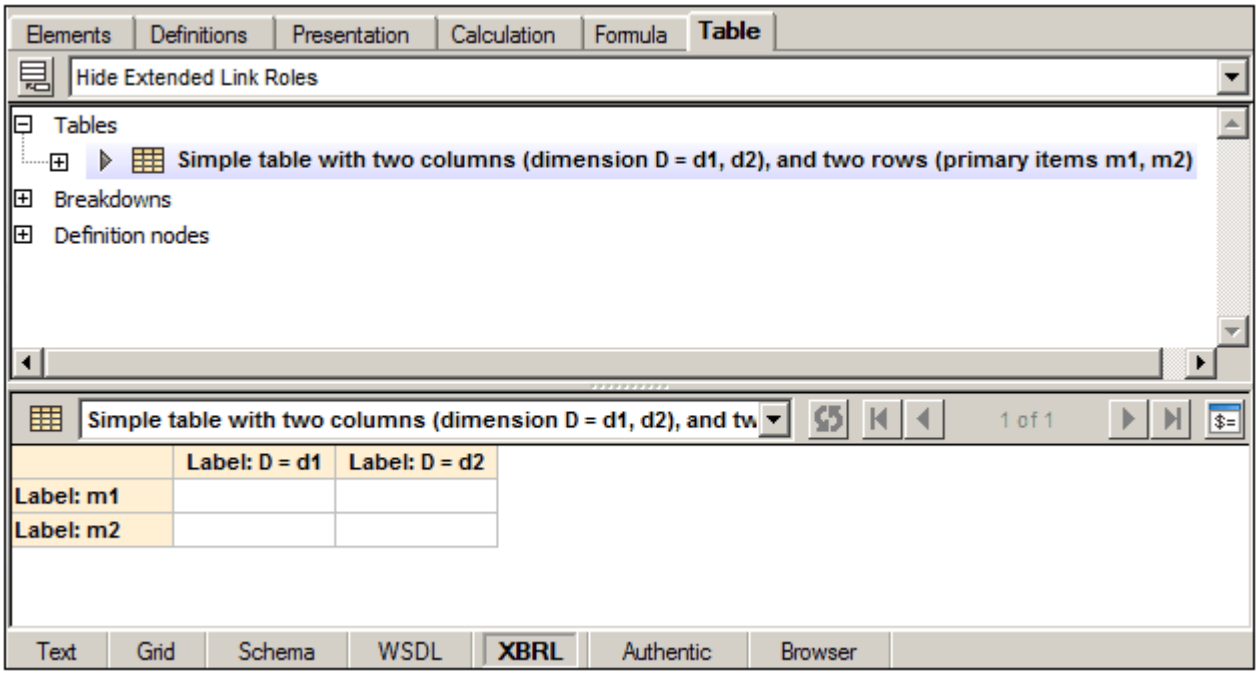
以下の点に注意してください：

- パラメーターは、ローカル パラメーターです。テーブル コンポーネントを右クリックして、 μ を選択することにより、この固有のテーブルが作成されます。これらは、最初のスクリーンショットで表示される グローバル パラメーターではありません。
- 2 つの QNames の 2 つのシーケンスの順序付けされたデカルトの製品は、4 つのテーブルを作成します: `dimMember × conceptName = {d1,d2} × {m1,m2} = {(d1,m1), (d1,m2), (d2,m1), (d2,m2)}`
- テーブル セットを説明するテーブル定義が、ダイアグラム内で選択された場合、テーブル レイアウト プレビュー内のナビゲーション アイコンが有効化され、テーブル セット内のプレビューを確認することができます。現在のプレビューされているテーブルは、ツール バー内のオーダーされたテーブル セット内のインデックスにより指定されます。上のスクリーンショットでは、現在のテーブルは、4 の 2 です。現在プレビューされているテーブルの パラメーターの値は、ポップアップで表示されます（*スクリーンショットを参照*）。
- テプレビューが現在の定義と同期されていない場合、例えば新しいコンセプトが追加された後、テーブル レイアウト プレビューの更新 ツール バー アイコンが有効化されます。
- テーブル レイアウト プレビューの パラメーターの値 ツール バー ボタンは、XBRL パラメーターの値 ダイアログを開きます。すべてのテーブル パラメーターの値とデータ型（グローバルおよびローカル）を編集することができます。

(C) 2015-2021 Altova GmbH

17.5.7 テーブル レイアウトプレビュー

XBRL テーブルレイアウトプレビューペインは、テーブル定義 ツリーの下のテーブルタブ内にあります（*下のスクリーンショット参照*）。レイアウトプレビューペイン内のコンボボックスは、*アクティブなタクソノミのテーブルリンクベース*内のすべてのテーブルをリストします。テーブルのレイアウトをプレビューするには、プレビューペインのコンボボックスのテーブルを選択します（*下のスクリーンショット参照*）。プレビューは*レイアウトのみ*をプレビューする点に注意してください。テーブルセルは、作成されません。これは、XBRL タクソノミ内にデータが存在しないからです。



テーブルレイアウトプレビューにより以下を行うことができます:

- テーブルレイアウトを、テーブル定義が変更されると自動的にアップデートするプレビューを使用してビジュアル化します。
- テーブルセルをクリックすることによりコンポーネントの定義に直接移動、または、テーブル定義ツリー内のコンポーネントをクリックしてテーブルセルに移動します。
- （パラメーターの値 ツールバーアイコンを介して）XBRL テーブル パラメーター ダイアログにアクセスして、テーブル パラメーターを管理します。

編集

テーブル定義とタクソノミの変更は以下のとおり処理されます:

- *テーブルの変更:* テーブル定義の構造が変更されると、（テーブルタブ内のダイアグラム または詳細入力ヘルパーを介して、）テーブルのレイアウトプレビューはすぐにアップデートされます。パラメーター定義 または パラメーターの値の変更もこのアップデートをトリガーします。
- *DTS の変更:* テーブルレイアウトプレビューは、Altova の XPath エンジンを定義ノード内の XPath 式 を評価するために使用します。XPath モデルは、XBRL ビューにタクソノミスキーマをロードする際に作成され、検証中にアップデートされます。基になる DTS が変更されると、（例えば、コンセプト または リンクベースの編集による）テーブルプレビューは、変更された DTS とはもう同期ではありません。プレビューのツールバー内のテーブルが同期されていないことを示すアイコンがこの状態を示し、ツールヒントがヒントを与えます: *プレビューは、更新 ボタンを使用して、手動で更新される必要があります。*「更新」コマンドは、DTS の再検出を引き起こします。ですから、これは、タクソノミの検証と同等です。

	テーブルがDTS と同期されていません。
	テーブルの更新

エラーの扱い

テーブルレイアウトプレビューに関連するエラーは、以下のように扱われます:

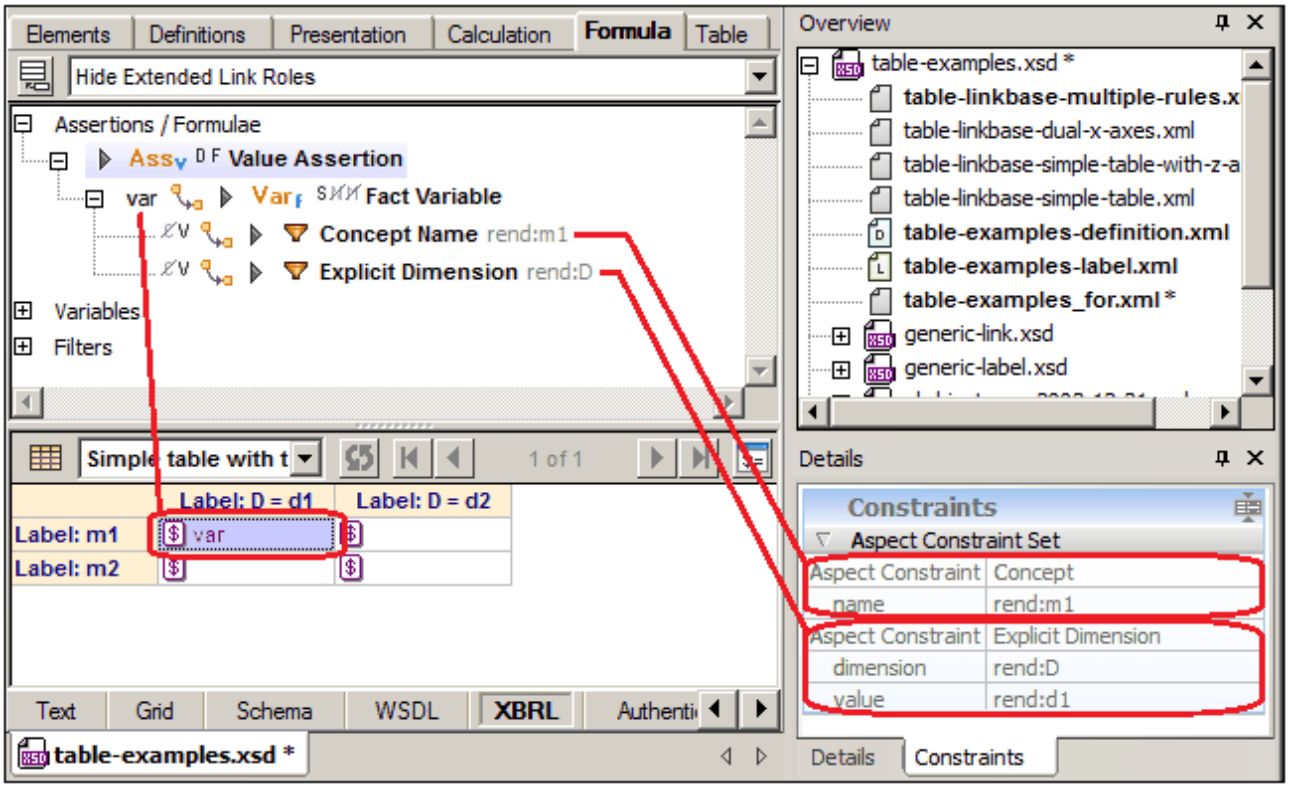
- *テーブル定義ノード内の無効な式*: テーブル定義ノードが解決することの出来ないXPath 式を含む場合、対応するレイアウト セルのヘッダーは、赤で表示されます。この場合は、無効なアスペクト制約は、詳細入力ヘルパーの制約タブ内でハイライトされて表示されます。
- *解決することのできないリレーションシップノード*: 無効なプロパティまたは無効なDTS のために、リレーションシップノード[798]が解決されない場合、ハイライトされたエラーテキストと共にプレースホルダセルを表示します。
- *子ノード無しのマージされたルールノード*: マージされたルールノード[798] が子ノードを持たない場合、レイアウトは、エラーテキストと共にプレースホルダセルを表示します。
- *無効なDTS*: タクソノミが無効な場合、 XBRL タクソノミ エディターにタクソノミスキーマをロード、または検証 の後は、XPath モデルを使用することができません。テーブルレイアウトプレビューは、テーブルがDTS と同期されていませんのツールバーアイコンを表示し、エラー状態になります。.これに反して、レイアウトはまだ作成されることができる可能性があります。XPath 式は、しかしながら評価されることはできません。ツールバーアイコンのツールヒントはこの問題の解決方法をアドバイスします(検証 エラーおよび再検証 タクソノミをフィックスすること。

17.5.7.1 テーブル レイアウトプレビュー内でのフォーミュラの作成

テーブルレイアウトプレビューにもフォーミュラ タブは、表示されます(*下のスクリーンショット参照*)。変数セットの下(すなわち、フォーミュラまたは 値/存在 アサーション下) のファクト変数の作成をサポートします。この場合は、フォーミュラ リンクベース内で対応するテーブル定義ノードが非表示のため、軸内のセルを選択することはできません。この一方、フォーミュラ ツリー 内の変数セットが選択されるとすぐに有効化される**「ファクト変数の追加」**アイコンを表示します(*下のスクリーンショット参照*).

選択された フォーミュラ または アサーションへのファクト変数の追加

変数セットにファクト変数を追加するには、フォーミュラ タブ内の変数セットを選択します。(変数セットは、フォーミュラ または 値/存在 アサーションです。下のスクリーンショット内では、 選択された 変数セットは、値アサーションです。) テーブルレイアウトプレビューのセル内で、**「ファクト変数の追加」**アイコンが有効化されます。 変数セットに変数を追加するためにアイコンをクリックします。実行中、データ セルにより定義される、各アスペクト制約のための適切な フィルターを含む新しいファクト変数は、選択された 変数セットの下に作成されます(すなわち、フォーミュラ または アサーション)。



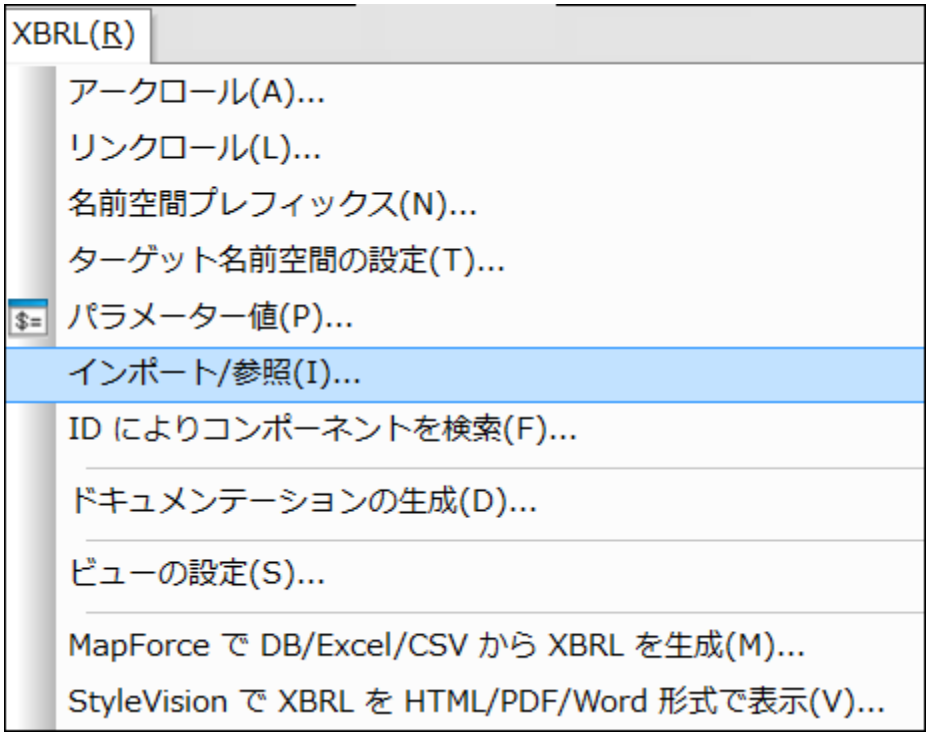
選択されたフォーミュラまたはアサーションが、プレビュー内のデータセルのアスペクト制約に一致するフィルターを含む変数を持つ場合、変数名 がデータセル内に表示されます。これは、「ファクト変数の追加」アイコンを介して、新しいファクト変数を作成した後の特徴です（*上のスクリーンショット参照*）.

17.5.8 テーブルコンポーネントの検索

テーブルコンポーネントは、以下の方法で検索することができます（i）ID を使用、および（ii）ドキュメント内のコンポーネントの発生のナビゲート。

ID によりテーブルコンポーネントを検索する

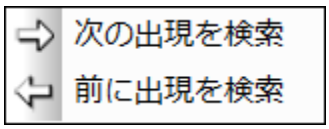
同じ型の複数のコンポーネントを含む、フォーミュラまたはテーブルリンクベースを持つ、タクソノミ内では、（例 アサーション、フィルター、テーブルコンポーネントを ID により検索することが役にたちます。メニューコマンド「XBRL | ID によりテーブルコンポーネントを検索する」により ID による検索が有効化されます。



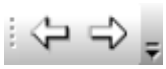
コマンドをクリックすると、検索するID を問うダイアログがポップアップします。

コンポーネントの発生を検索する

フォーミュラコンポーネントのほとんどは、フォーミュラリンクベースダイアグラム内に複数回表示されています: (i) 適切な セクション ノードの下に直接存在する 定義、(ii) コンポーネントへのすべての参照 (リレーションシップを介して)。コンポーネントのコンテキスト メニュー内のコマンド「**次の発生を検索**」および「**前の発生を検索**」は、フォーミュラコンポーネントが参照されているすべての箇所にナビゲートします (*下のスクリーンショット*)。



これらのコマンドには、ツール バーアイコンを介してもアクセスすることができます (*下のスクリーンショット*)。



コンポーネントの定義に到達すると、効果に関するメッセージが表示されます。

17.6 XULE

(XBRL ルールからの) XULE は XBRL レポートとタクソノミをクエリするための言語です。言語の主目的はデータの品質を保証するために入力の前にレポートをクエリしチェックする機能を提供することです。 XULE によりレポートを大まかに2つの方法でチェックすることができます:

- *レポート内のデータからの出力を提供する方法:* レポート内のデータをクエリし、レポート内のデータから結果を計算します。出力を品質のために評価します。
- *アサーションの作成:* レポート内のデータはこれらのアサーションに対してテストされ、結果に従い適切なアクションを取ることができます。

インターネットリンク

- XULE ホームページ (XULE の大まかな概要が含まれています)
- XULE 言語構文ドキュメント

XMLSpy 機能

XMLSpy は次の XULE 機能を提供します:

- XBRL インスタンスドキュメントに対して XULE ドキュメントと式を処理する**ビルトイン XULE プロセッサー**。
- XULE 適合ドキュメントの作成。
- XULE 仕様に対して正確な構文のために XULE ドキュメントを検証する機能。
- XULE ドキュメント内の構文色分け
- XULE ドキュメントの編集時の XULE 言語の自動補完コンストラクト
- XBRL インスタンスドキュメントをインタラクティブにクエリするための特別な **XULE ウィンドウ**。
- XBRL インスタンスを単一の XULE ドキュメント、または zip アーカイブに保管されている XULE ドキュメントのセットに対して処理する機能。処理は XMLSpy エンジン、または RaptorXML(+XBRL) Server 957 エンジンにより実行することができます。
- XULE ドキュメントとドキュメントセットの実行の XMLSpy プロジェクト 950 内での統合。

このセクションのサブセクションで機能の詳細を確認してください。

Altova RaptorXML+XBRL Server XULE

Altova RaptorXML+XBRL Server コマンドラインからパワフルな Python API を含む複数のサーバーとエンジン APIs を使用して XULE ドキュメントを処理することを可能にするカスタム化可能で高速の処理機能を提供します。詳細に関しては以下を確認してください:

- Altova Web サイト内での RaptorXML+XBRL ページ
- Altova RaptorXML+XBRL の DQC 証明書
- RaptorXML+XBRL 製品ドキュメント
- RaptorXML Python API ドキュメント

17.6.1 XULE ドキュメント

XMLSpy には XULE ドキュメントの作成、検証、および実行をサポートする機能が搭載されています。このセクションではこれらの機能について説明されています。

XULE 適合ファイル(.xule ファイル)

.xule ファイル型はXULE 適合としてXMLSpy 内で既に定義されています。これは.**xule** ファイルがXMLSpy 内で開かれており、構文色分けと自動補完機能をヘルプとして使用が可能なことを意味します。 オプションダイアログ内のファイル型セクション[1458] 内で、他のファイル拡張子がXULE 適合になるように指定することもできます。

XULE ドキュメントセット

複数のXULE ドキュメントをzip アーカイブ(通常 **.zip**) にパッケージすることも可能です。このzip アーカイブはXULE ドキュメントセット(またはXULE ルールセット)です。実行するXULE ファイル[827] としてzip ファイルを指定し XBRL インスタンス上でセットされているXULE ドキュメント全体を実行することができます。

zip アーカイブはすべての構造を持つことができます。アーカイブの全てのレベルでのXULE ファイルはXULE 実行中に使用されます。XULE ファイルではないファイルはフィルターで除外されます。

構文色分け

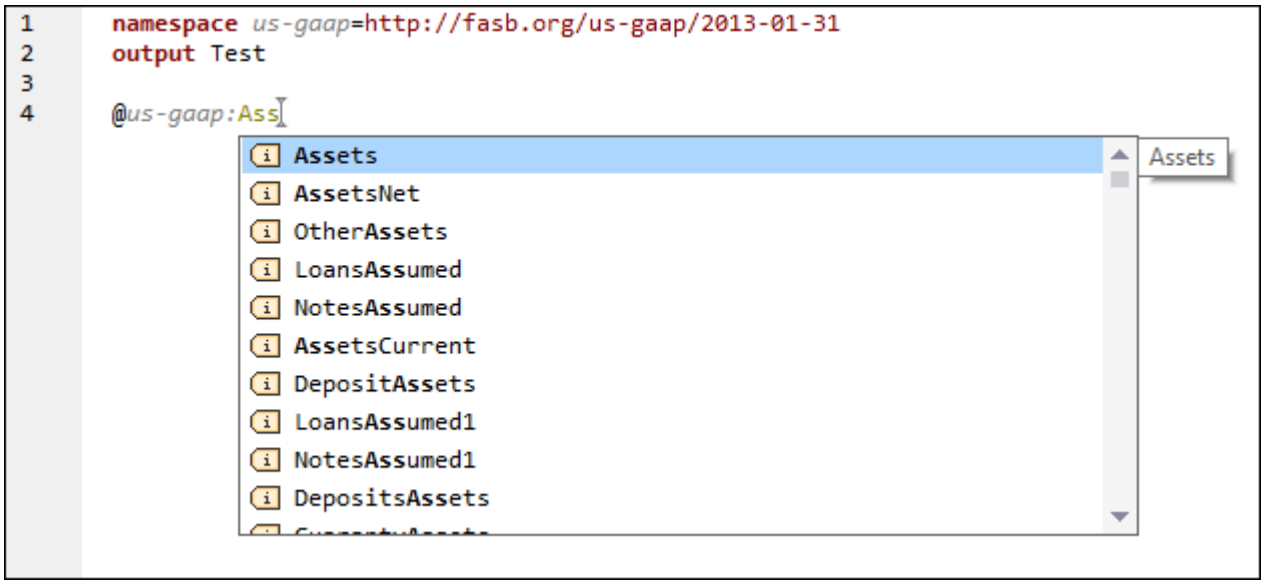
XMLSpy のテキストビュー内でXULE ドキュメントを編集することができます。下のスクリーンショットはサンプルXULE ドキュメントのデフォルトの構文色分けを示しています。オプションダイアログの「フォントと色 | テキストビューセクション」[1470] 内の構文色分けをカスタム化することができます。

```
1   namespace us-gaap=http://fasb.org/us-gaap/2013-01-31
2
3   // Balance sheet test
4   assert balance-test unsatisfied
5
6   @us-gaap:Assets#assets == @us-gaap:LiabilitiesAndStockholdersEquity#liabilities
7
8   message
9
10  "Balance sheet is unbalanced in period {$assets.period}, {$assets} != {$liabilities}"
```

自動補完

XULE ドキュメント内にルールを入力すると、2種類の自動補完ヘルプを使用することができます:

- XULE 言語構文に関連した自動補完
- 選択されたXBRL タクソノミの構造に関連した自動補完



情報ウィンドウ内で使用するタクソノミを選択することができます（下のスクリーンショットを参照）。XULE タブ内で、タクソノミアイテムの横のアイコンをクリックします。表示されるメニューから「タクソノミの選択」をクリックします。タクソノミを参照し選択します。一度に1つのタクソノミしか追加することができません。新規のタクソノミを追加すると、前のタクソノミは置き換えられます。または、新規のタクソノミを追加する前にタクソノミを削除することができます（下のスクリーンショットを参照）。

メモ XULE ドキュメントに必要とする全てのタクソノミコンポーネントの名前空間宣言を追加する必要があります（上の XULE ドキュメントスクリーンショットを参照してください）。名前空間プレフィックスはタクソノミに一致する必要はありませんが、同じ名前空間プレフィックスを保持し混乱を避けてください。名前空間宣言を追加する必要がない場合、タクソノミコンポーネントの自動補完は作動しません。

XULE の検証

XULE ドキュメントは「XML | 検証 (F8)」コマンドを使用して XULE 言語使用に対して正確な構文のために検証することができます。

XMLSpy プロジェクト内での統合

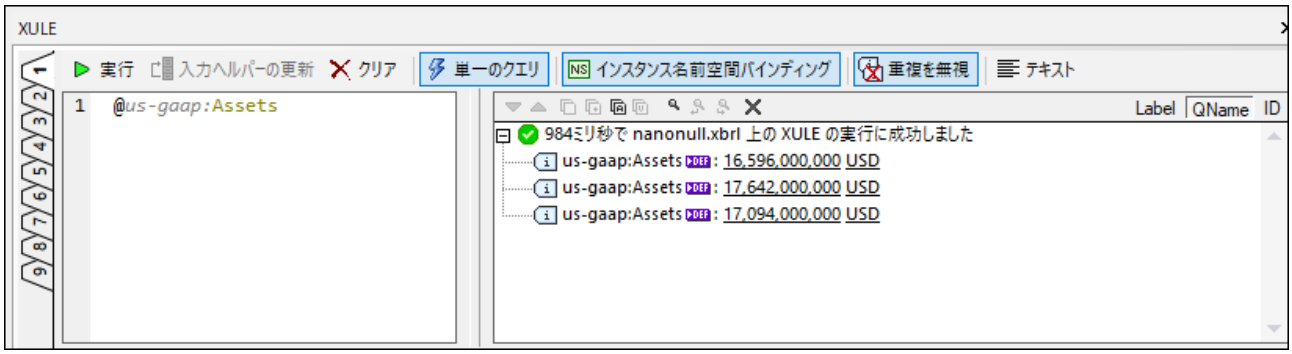
XMLSpy プロジェクト内で XULE ドキュメントを以下のように統合することができます:

1. XULE ドキュメントを XMLSpy プロジェクト 950 内の適切なフォルダーに追加します。
2. そのフォルダー内のプロジェクトプロパティ 1194 内で、XULE ファイルを作動する XBRL インスタンスファイルを入力します。
3. プロジェクト内で XULE ドキュメントを右クリックし、「XBRL | XULE の実行」コマンドを選択します。XULE ドキュメントはフォルダーのために設定されている XBRL インスタンス上で実行されます。メッセージウィンドウ内または、新規のドキュメント内で結果は表示されます（XULE 実行オプション 1488 を参照してください）。

17.6.2 XULE ウィンドウ

XULE ウィンドウ (*下のスクリーンショット*) は出力ウィンドウ[121]です。これによりアクティブな XBRL インスタンスドキュメントをインタラクティブにクエリし、クエリの結果を確認することができます。

XULE ウィンドウはXMLSpy GUI の下の[108] メインウィンドウの下にデフォルトで配置されています。9つのタブが存在し、2つのペインに分割されています: (i) アクティブなドキュメントを実行する XULE 式 (または XULE ルール) を入力する XULE 式ペイン。(ii) 実行の結果を表示する結果ペイン。



アクティブな XBRL インスタンスドキュメント上で XULE 式をインタラクティブに実行するには、以下を行います:

1. クエリする XBRL インスタンスドキュメントをメインウィンドウでアクティブなドキュメントにします。
2. XULE 式ペイン (*左側のペイン*) 内に XULE 式を入力します。構文色分けと自動補完を含む式ペインの機能を編集します。
3. 式を実行するために**ウィンドウツールバー**内の「**実行**」をクリックします。
4. 実行の結果は結果ペイン (*右側のペイン*) 内に表示されます。結果内のリンクをクリックして XBRL インスタンスドキュメント内の対応するノードに移動することができます。

メモ XULE のための構文色分けは (「フォントと色 | テキストビュー」[1470] セクション内の) オプション[1454] ダイアログ内でカスタム化することができます。自動補完に関する詳細は、「**入力ヘルパーの更新**」ツールバーコマンドの説明を参照してください。

ツールバー: コマンドとオプション

XULE ウィンドウのツールバーはコマンドをオーバーライドし、XULE 式を作成し実行するためのコマンドと役に立つオプションを提供します。

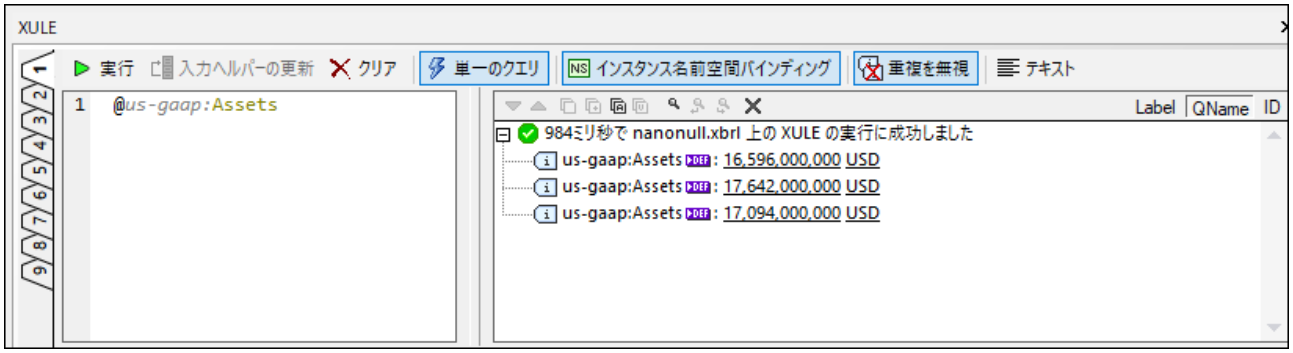
実行
XULE 式を実行するために「**実行**」をクリックします。

入力ヘルパーの更新
XULE 式ペインは入力中に2種類の自動補完を提供します: (i) XULE 構文に関連する自動補完 (ii) アクティブな XBRL インスタンスドキュメントの構造に関連した自動補完。XBRL インスタンスの構造を考慮する場合、XULE ウィンドウは XBRL インスタンスにより参照される XBRL タクソノミを読み取る必要があります。 XBRL インスタンスに関連するタクソノミを静的にロードするために「**入力ヘルパーの更新**」をクリックします。タクソノミがロードされた後、ドキュメント構造に関連した自動補完を使用できるようになり、(この XBRL インスタンスドキュメントではもう必要とされない) このツールバーボタンが無効化されます。「**実行**」がクリックされる都度タクソノミも読み取られることに注意してください。

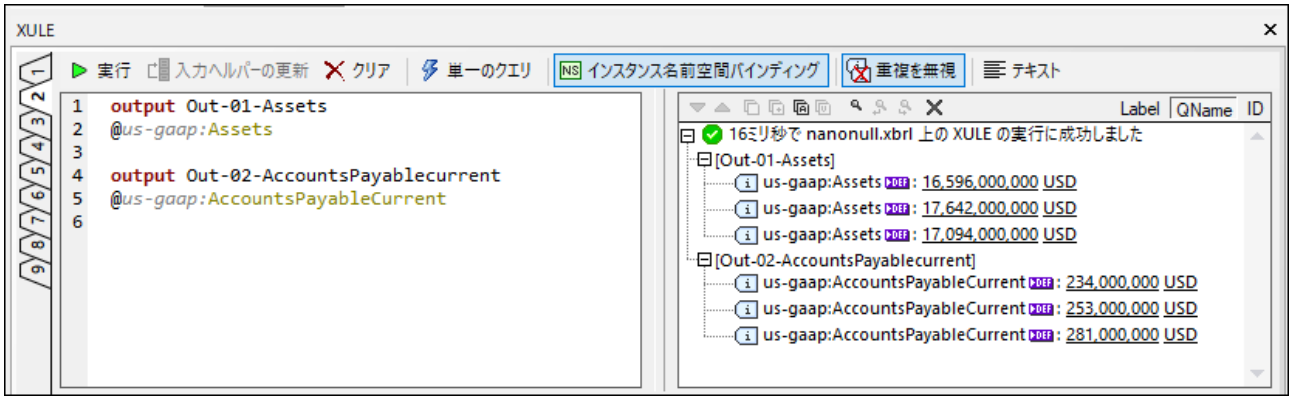
クリア
式をクリアするために「**クリア**」をクリックします。

単一のクエリ
単一クエリモードはXMLSpy XULE ウィンドウに固有の機能です。以下を行うことができます: (i) `output` キーワードを持たない単一クエリとして入力することができ (ii) 単一出力として結果を生成することができます。有効な XULE 構文は`output` キーワードを必要としますが、インタラクティブにXBRL ドキュメントを素早くクエリする場合、`output` キーワード無しに単一クエリを入力できることは有益です。

下の2枚のスクリーンショットは単一クエリと複数クエリの使用法を表示しています。

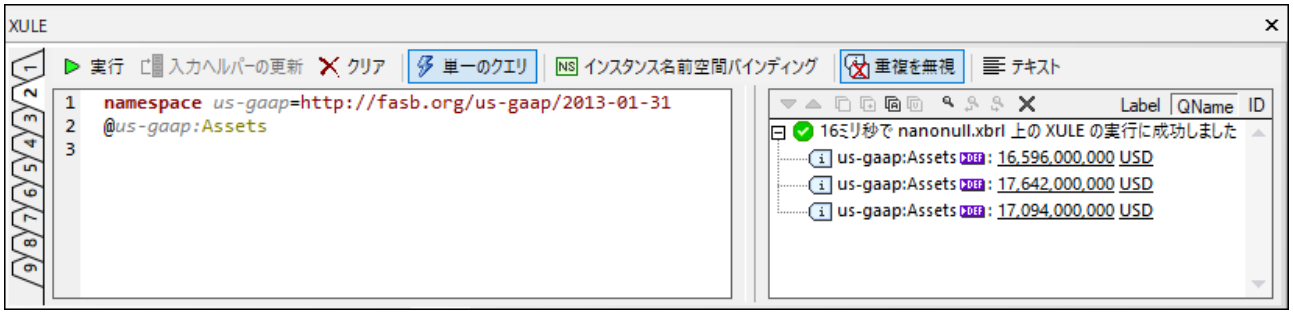


単一クエリモードが有効化されている状態: 「出力」キーは必要ありません。



単一クエリモードが無効化されている状態: 「出力」キーワードと複数の出力を持つ使用した「出力」クエリ

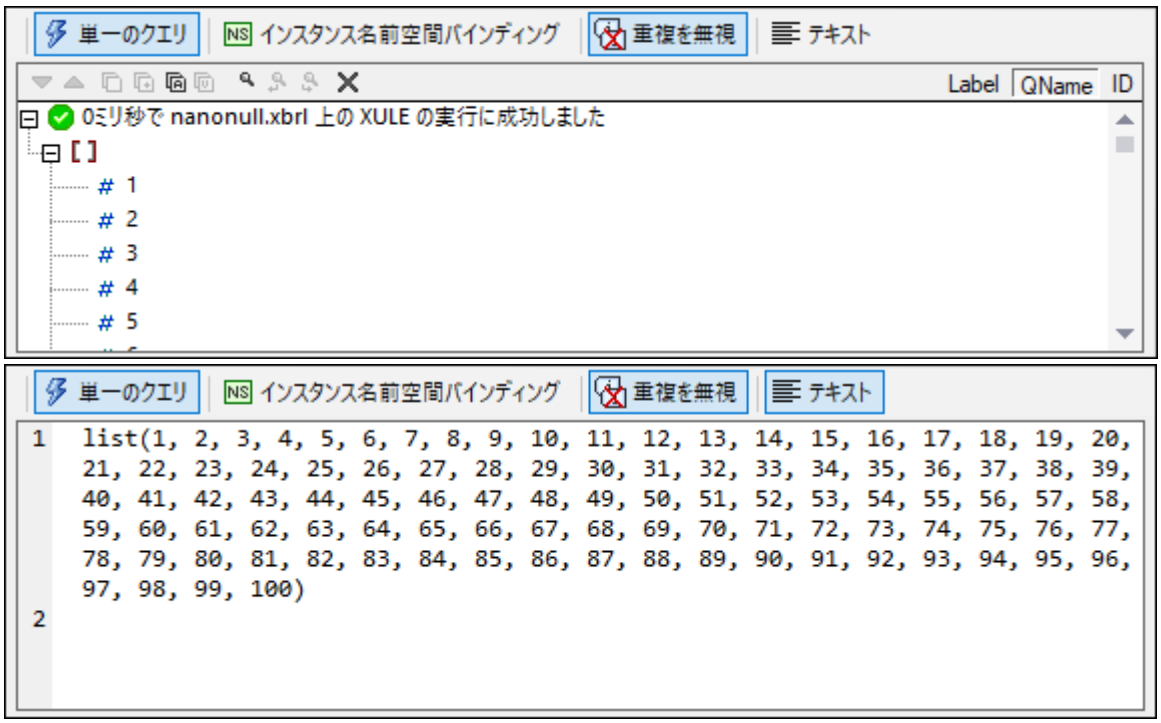
インスタンス名前空間バインディング
「**インスタンス名前空間バインディング**」オプションが選択されているとXULE クエリ内で名前空間を宣言する必要はありません。名前空間プレフィックスはXBRL インスタンス内で宣言されているインスタンス内名前空間にバインドされます。 例えば、上のスクリーンショット内で `us-gaap` 名前空間プレフィックスはインスタンスドキュメント内で名前空間にバウンドされています。他方、「**インスタンス名前空間バインディング**」オプションは選択解除され、 XULE クエリ内で、下のスクリーンショット内で表示されるように`namespaces` キーワードと共に名前空間を宣言しなければなりません。



名前空間宣言を持つXULE クエリをフィルする手間を省けるためにこのオプションは役に立ちます。

重複を無視
HTML コード内で同じファクトがソートされると重複ファクトはインライン XBRL 内で最も一般的に発生します。「**重複を無視**」オプションは重複ファクトが一度のみ出力されることを指定します。

テキスト
テキスト オプションはテキストとツリー出力間の結果ペインの出力を切り替えます。例えば、下のスクリーンショットはツリーフォーム*(左)* とテキストフォーム*(右側)*を表示しています。クエリ: `list(for $i in range(100) $i).`



結果ペイン

ラベル、QNames、IDs と共に結果は表示されます。クエリの実行前に結果ペインのツールバー内でオプションを選択します。下のスクリーンショットはラベル*(左側)* とQNames *(右側)*と共に結果を表示しています。

(C) 2015-2021 Altova GmbH

結果ペインのツールバーにはナビゲート、検索、コピー機能を提供するアイコンが含まれています。これらのアイコンは左から始まり下のテーブルで説明されています。対応するコマンドは結果リストアイテムのコンテキストメニュー内で使用することができます。

アイコン	動作
次へ, *前へ*	結果リスト内にそれぞれ次の、および前のアイテムを選択します。
選択されたメッセージをコピーする	クリップボードに選択された結果アイテムをコピーします。
子を含む選択されたメッセージをコピーする	クリップボードに選択された結果アイテムとその子をコピーします。各アイテムは個別のラインとしてコピーされます。
全てのメッセージをコピー	クリップボードに全ての結果アイテムをコピーします。
選択されたラインの値をクリップボードにコピーする	選択された結果アイテムの値のみをクリップボードにコピーします。
検索する	結果内の文字列を検索するために *検索* ダイアログを開きます。
前を検索する	*検索* ダイアログ内に最後に入力された用語の前の発生を検索します。
次を検索する	*検索* ダイアログ内に最後に入力された用語の次の発生を検索します。
クリア	結果リストをクリアします。

17.6.3 XULE 実行

XBRL インスタンスドキュメント上で XULE ドキュメント[821]を実行する場合、メニューコマンド「XBRL | XULE の実行」[1409] を選択します。コマンドを以下の場合に使用することができます:

- XULE ドキュメントがアクティブなドキュメントの場合、コマンドの選択はXULE ドキュメントが実行されるXBRL インスタンスを選択するようにプロンプトします。
- XBRL インスタンスドキュメントがアクティブなドキュメントの場合、コマンドの選択は使用する XULE ドキュメント、またはXULE ドキュメントセットXULE ドキュメントを選択するようにプロンプトします。
- (i) XULE ドキュメント (**.xule** ファイル)ー または[XULE ドキュメント セット](#) 821 (zip アーカイブ)ー およびXBRL インスタンスドキュメントが[XMLSpy プロジェクト](#) 950 の一部の場合 (ii) XBRL インスタンスファイルが[XMLSpy プロジェクトのプロパティ](#) 1194 内のターゲット XBRL ファイルとして設定されている場合、XMLSpy プロジェクトウィンドウ内のプロジェクト XULE ファイルを右クリックし、「**XULE の実行**」を選択します。XULE ドキュメントはXULE 実行のためのプロジェクトターゲットである XBRL 上で実行されます。

更に、XMLSpy インターフェイスからのXULE 処理コマンドを含む[RaptorXML(+XBRL) Server](#) 957 コマンドを実行するために、XMLSpy を設定することができます。

XULE 実行オプション

次のXULE 実行オプションを使用することができます:

- 以下に送信することができます: (i) メッセージウィンドウに送信 (ii) メモリ内に一時的に保管される新規 XMLSpy ウィンドウ内に表示される新規ドキュメント。このドキュメントは「ファイル | 名前を付けて保存」 1138 を使用して保存することができます。
- 同じファクトへの複数のレファレンスを参照するファクトを重複します。(HTML コードのために同じファクトが複数回ノートされるとファクトの重複はインライン XBRL 内で通常発生します。)重複ファクトは一度のみレポートすることができます。

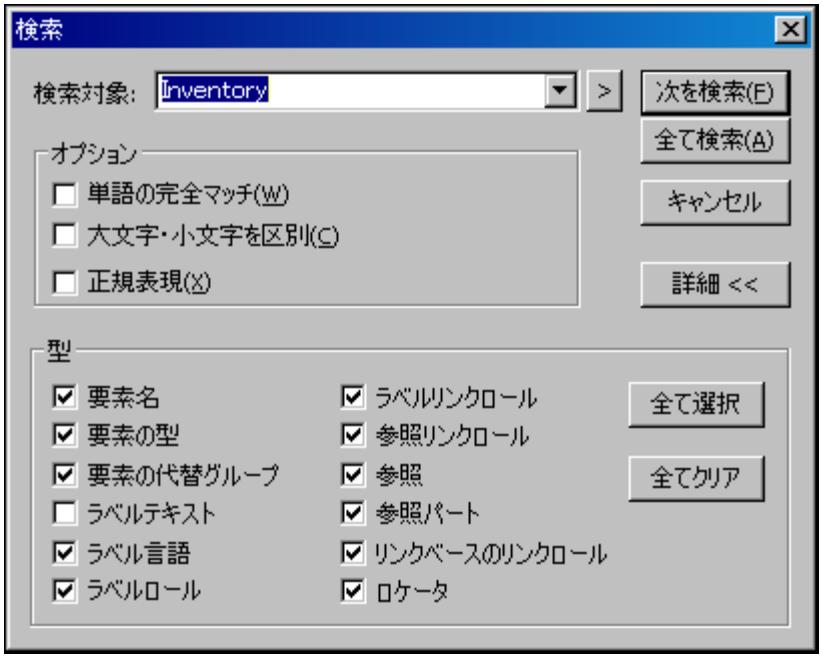
[オプションダイアログのXBRL XULE タブ](#) 1488 (「**ツール | オプション | XBRL | XULE**」)内で設定を選択することができます。

17.7 XBRL 内を検索

XBRL ビューにおいてタクソノミがアクティブになっている場合、XMLSpy のXBRL 内を検索する機能を使って、タクソノミの検索を行うことができます。XBRL 内を検索する機能へアクセスする方法は、以下のようにいくつかあります:

- XBRL ビューにて XBRL タクソノミがアクティブになっているときに、メニューコマンドから「編集 | 検索」を選択。
- XBRL ウィンドウ内の「検索」ボタンを押下。
- 「Ctrl」+ F を押下

上のどの方法を使用しても、(以下のスクリーンショットにある) 検索ダイアログが表示されます。

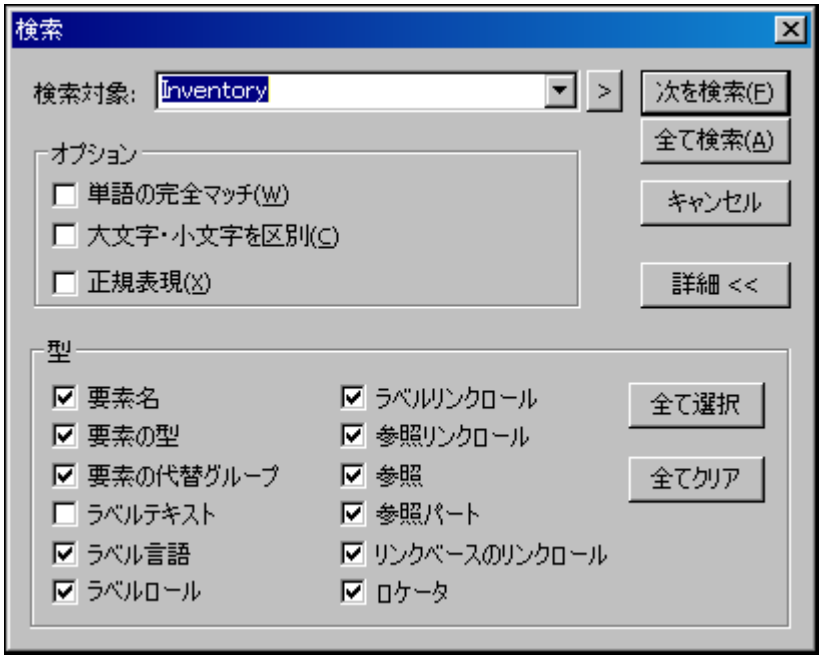


使用方法は以下のとおりです:

- 検索対象のテキストフィールドに検索語の入力[829]を行ない、必要なオプションにチェックを入れます。
- 型ペインにて検索を行うXBR Lコンポーネントを指定[829]します。
- 「次を検索」や「全て検索」ボタンを使ってコマンドを実行[831]します。
- スキーマの中を検索ウィンドウを使用[833]して検索の結果を確認し、コンポーネントの素早いナビゲーションを行ないます。

17.7.1 検索語

オプションペインにある対応するオプションにチェックを入れることで、大文字・小文字を区別した検索や、単語に完全にマッチする検索を行うことができます。正規表現を使った検索を行うには、「次を検索」や「全て検索」ボタンをクリックする前に、オプションペインにある*正規表現*オプションにチェックを入れてください。正規表現の使用に関する詳細は、以下を参照ください。

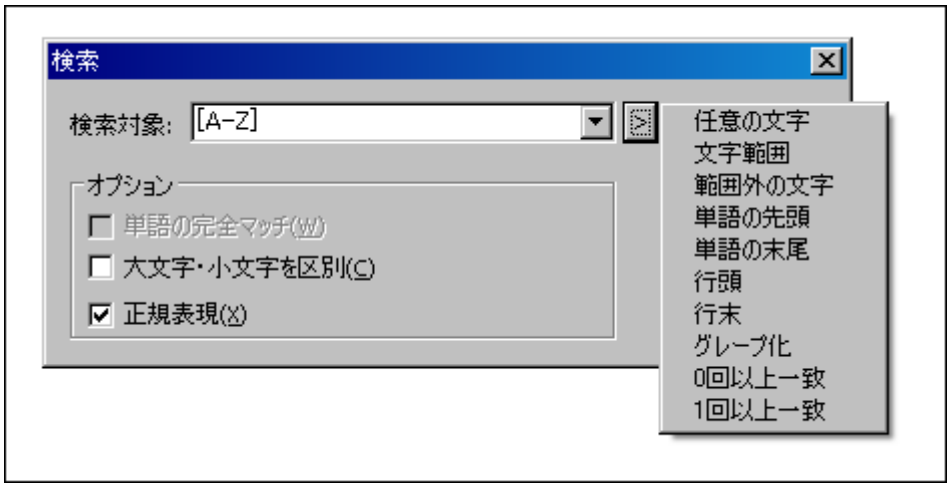


メモ: アルファベットではない(もしくは未知の)文字は、全て区切り文字として検索は行われます。例えば、`asset` という単語で検索が行われたとき、`xbrl:asset` という結果が返されますが、これはコロン文字(:)が区切り文字として見なされるためです。

型ペインにて、検索対象となるコンポーネントを指定することもできます。

正規表現

正規表現を使用することで、検索の条件を更に細かく設定することができます。ポップアップリストを使用することで、正規表現の構築を楽に行うことができます。リストにアクセスするには、検索語の入力フィールドの右にある > ボタンをクリックしてください。



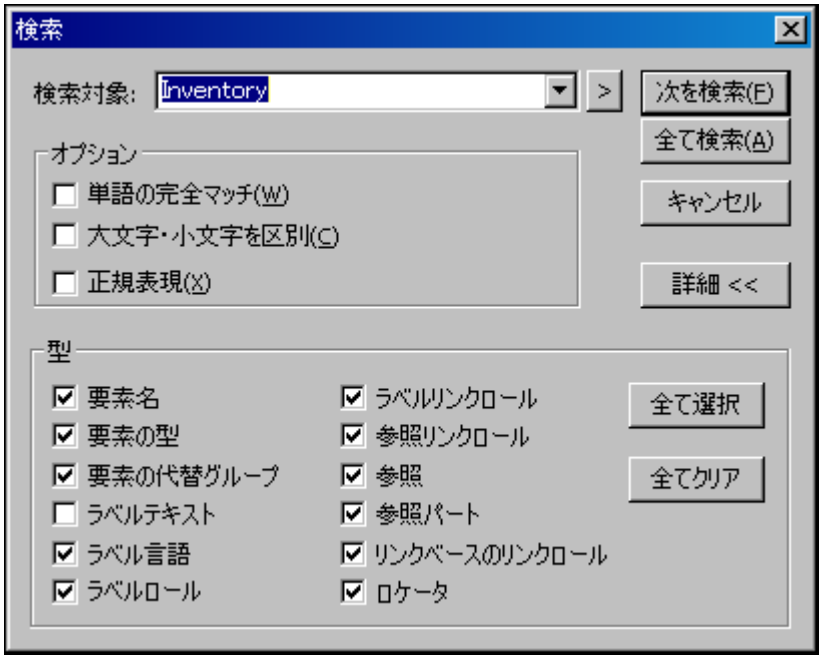
リストから必要なアイテムをクリックすることで、対応する正規表現の文字が入力フィールドへ挿入されます。以下は正規表現の構文文字(メタ文字)です。

.	任意の1文字にマッチします。
\(	タグ付けされた領域の開始を表します。

\)	タグ付けされた領域の終わりを表します。
\<	単語の開始にマッチします。
\>	単語の終わりにマッチします。
\x	通常特別な意味を持つ文字 x の使用を許可します。例えば、¥[は[として解釈され、文字セットの開始とは解釈されません。
[...]	文字の集合を表します。例えば、[abc] により、a、b、c 全てにマッチすることになります。例えば、[a-z] などとすることで、アルファベットの小文字全てを表す範囲を表すこともできます。
[^...]	文字の集合の否定を表します。例えば、[^A-Za-z] は、アルファベット以外の文字を表します。
^	行の開始を表します。
$	行の終わりを表します。例えばA+$ により、行の最後にある1つ以上のA がマッチします。
*	0個以上の文字の繰り返しにマッチします。例えば、Sa*m により Sm、Sam、Saam、Saaam.. といった文字にマッチします。
+	1個以上の文字の繰り返しにマッチします。例えば、Sa+m により Sam、Saam、Saaam... といった文字にマッチします。

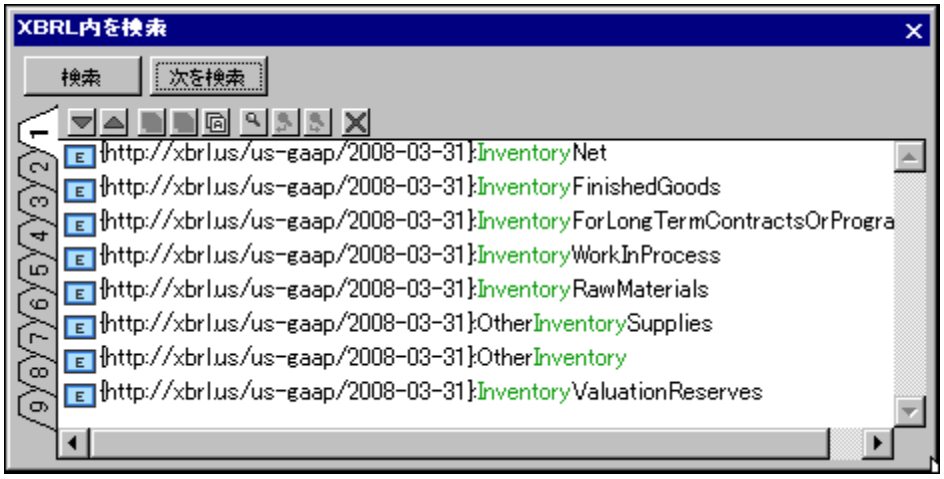
17.7.2 コマンドの実行

検索語が入力され、検索を行うコンポーネントの型に関する検索オプションがセットされた状態で、使用することのできるコマンドには「次を検索」と「すべてを検索」の2 種類あります。これらコマンドは検索ダイアログのボタンにより使用することができます（以下のスクリーンショットを参照）。



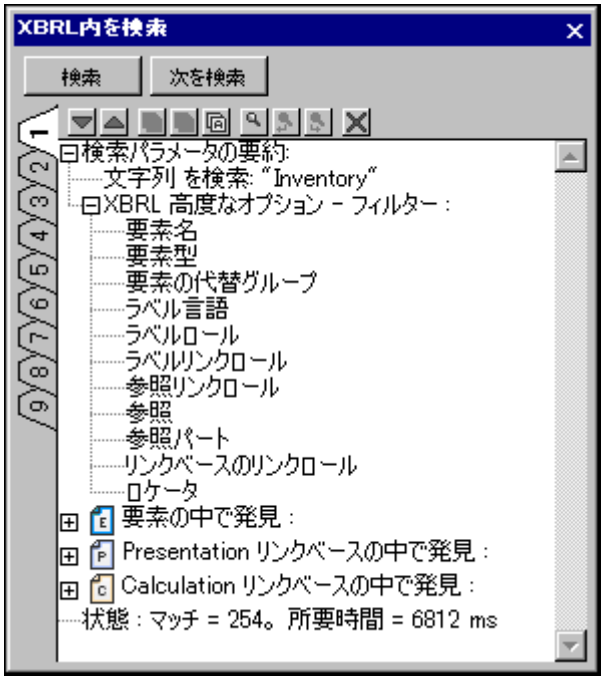
次を検索

次を検索コマンドにより、（スクリーンショットにあるように）検索語にマッチした次の検索結果がXBRL 内を検索ウィンドウに表示されます。次のインスタンスへの検索は、現在アクティブなドキュメントにてカーソルがある位置の次のセルから開始されます。「次を検索」処理は、全てのドキュメントにあるインスタンスが表示されるまで続けられます。



全て検索

全て検索コマンドにより、全ての検索結果が、その概要と共にXBRL 内を検索ウィンドウ内に表示されます（以下のスクリーンショットを参照）。

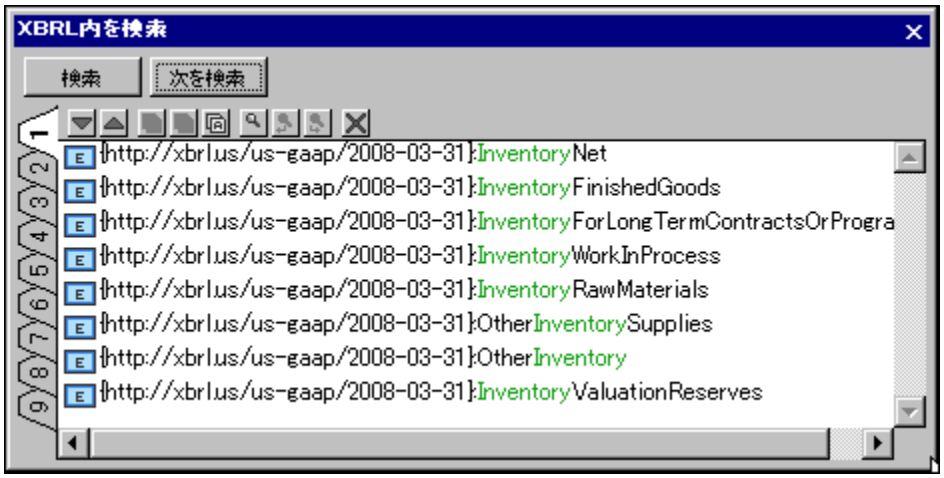


検索結果には、(i) 検索されたXBRL コンポーネント型の概要、(ii) リンクベースごとに表示される検索語に対してマッチしたインスタンスのリスト、そして (iii) マッチしたインスタンスの数と検索に要した時間が表示されます。各リンクベースグループは展開や縮退を行うことができ、そのグループ内のマッチを確認することができます。マッチをクリックすることで、XBRL ドキュメントの対応する要素がメインウィンドウにてハイライトされます。

XBRL 内を検索ウィンドウに関するより詳しい情報は、結果と情報のセクション[833]を参照ください。

17.7.3 結果と情報

「検索」や「次を検索」コマンドが実行されるたびに、その実行結果が(以下のスクリーンショットにある)XBRL 内を検索ウィンドウに表示されます。検索された単語は緑色で表示されます。以下のスクリーンショットでは、`Inventory` という単語が検索された結果が示されます。



XBRL 内を検索ウィンドウの機能

検索の結果は(1 から 9 まで番号が与えられている)9つあるタブに個別に表示させることができます。これらのタブを使うことで、各検索結果を個別のタブに保持しておくことができます。XBRL 内を検索ウィンドウの検索結果をクリックすることで、対応するコンポーネントが XBRL ビューのメインウィンドウに表示、ハイライトされます。これら XBRL 内を検索ウィンドウの機能により、目的のコンポーネントへの検索とナビゲーションを素早く行うことができます。

以下のツールバーコマンドを利用することができます:

- 「前」と「次」アイコンにより、選択されている結果から見て、それぞれ前、そして次の検索結果を選択することができます。
- 「メッセージをコピー」コマンドにより、選択されたメッセージ、選択されたメッセージとその子、そして全てのメッセージをクリップボードにコピーすることができます。
- 「検索」コマンドにより、XBRL 内を検索ウィンドウ内のテキストを検索します。
- 「クリア」コマンドにより、現在アクティブになっているタブに表示されている全てのメッセージを削除します。

18 Office Open XML、ZIP、EPUB

Office Open XML (OOXML) やZIP ファイル、そしてEPUB ファイルは、他のファイルを内部に含むパッケージという意味では、すべて似たようなものです。XMLSpy のArchive View では、これらパッケージの内部構造を閲覧するためのビューが提供され、XMLSpy 内部から構造の修正や、パッケージ内のファイルにアクセス、編集を行うことができます。EPUB ファイルの場合、XMLSpy のArchive View からブラウザービュー内にてEPUB ブックを直接閲覧することができます。

Office Open XML (OOXML)

OOXML はドキュメント、スプレッドシート、そしてプレゼンテーションの記述を行うためのファイル形式です。元々はMicrosoft のOffice パッケージのために開発されましたが、現在はオープンなECMA 仕様になっています。

OOXML ファイルの構造

各 OOXML ドキュメントは、Open Package Convention に従った複数ファイルのパッケージになっています。パッケージはXML やその他のデータファイル(例えばイメージファイル)と、それらファイルの関係が記述されたリレーションシップファイルから構成されています。

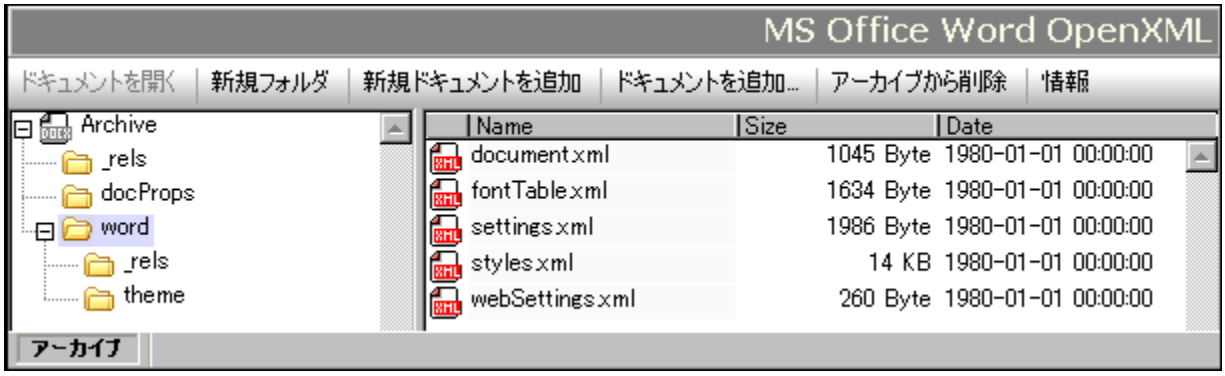
OOXML ファイルの内部構造や、内部フォルダー、ファイル名の構造はドキュメントの種類により変化します。しかし、`[Content_Types].xml` という共通のXML 構造ファイルがルート部にあり、ドキュメントの種類により変化するディレクトリと、`_rels`、そして`docProps` の3つのディレクトリは常に存在します。例えば、`.docx` ドキュメントの場合、ディレクトリの名前は`word` になり、`.xlsx` ドキュメントの場合は、`xl`、そして `.pptx` の場合は`ppt` となります。

```
OOXML ファイル
|-- ファイル:          [Content_Types].xml
|-- フォルダー:    _rels
|-- フォルダー:    docProps
|-- フォルダー:    word/xl/ppt
```

- `_rels` フォルダーには`rels.xml` ファイルが含まれており、パッケージ内の様々なファイルが記述されています。
- `docProps` フォルダーにはドキュメントプロパティの記述を行う`app.xml` と`core.xml` が含まれています。
- `word`、`xl`、そして`ppt` フォルダーには、ドキュメントのコンテンツとなるXML ファイルが含まれています。例えば、`word` フォルダーには、ドキュメントのコアコンテンツとなる`document.xml` というファイルが保存されています。

XMLSpy のアーカイブビューで見るOOXML

(以下のスクリーンショットにある)XMLSpy のアーカイブビューを使うことで、OOXML ファイルのコンテンツの閲覧と編集を行うことができます。



画面の左側にあるフォルダービューには、パッケージ内のフォルダーが表示され、メインウィンドウにはフォルダービューで選択されているフォルダー内にあるファイルが表示されます。アーカイブビューを使えば、アーカイブからファイルやフォルダーの追加や削除を行うことができます。更に、アーカイブビューのファイルをダブルクリックしてXMLSpy にて素早く編集を行うこともできます。

(C) 2015-2021 Altova GmbH

OOXML の内部ファイルに対するインテリジェントな編集

OOXML パッケージ内部にある XML ドキュメントは、標準的なスキーマをベースにしています。XMLSpy は OOXML ドキュメントに対してインテリジェントな編集機能をサポートしており、入力ヘルパーや自動補完、検証などの機能を使用することができます。

ZIP ファイル

ZIP ファイルは複数のファイルを可逆圧縮の形で含むアーカイブファイルです。含まれるファイルの形式には様々なものがあります。XMLSpy のアーカイブビューを使えば、ZIP ファイルの作成、内部構造の修正や、アーカイブされたファイルの編集を行うことができます。これらの機能については、このセクション以下の ZIP ファイル[840] サブセクションにて記述されます。

EPUB ファイル

EPUB ファイルとはデジタル形式の出版物（EPUB books）を配布するためのファイルの集合体のことです。Archive View[257] では EPUB ファイルを開いたり、EPUB ファイルの作成、デジタル EPUB ブックのプレビュー、EPUB アーカイブのコンポーネントファイルを XMLSpy にて直接編集し、EPUB ファイルを検証、コンポーネントを EPUB アーカイブへ保存しなおすことができます。詳細については EPUB ファイル[842] のセクションを参照ください。

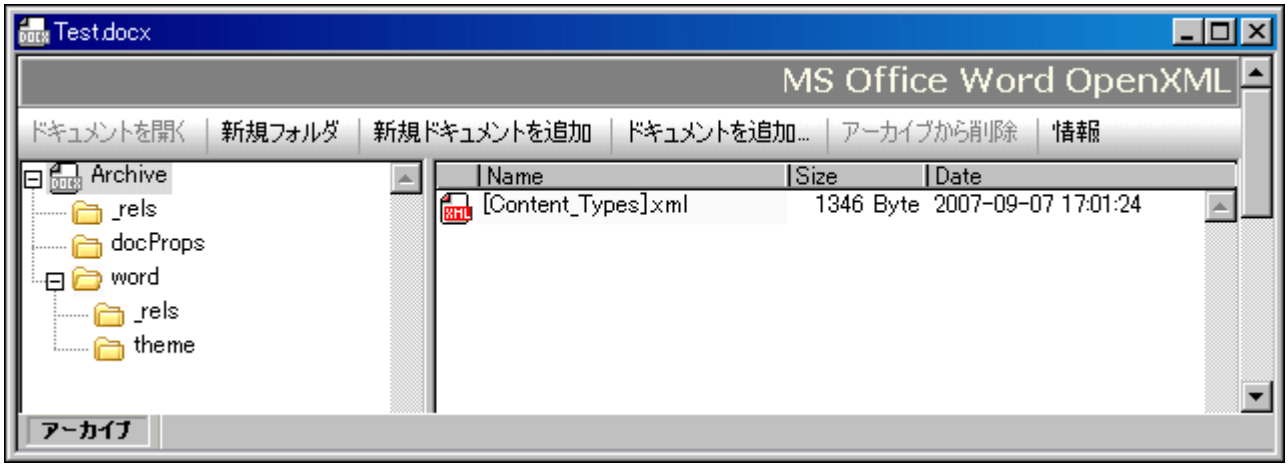
18.1 OOXML ファイルの編集

このセクションでは、Archive View を使ったOOXMLドキュメントの編集方法について説明します。以下の機能について解説します:

- OOXML ファイルを作成、開き、保存する 836
- OOXML ファイルの構造を編集する 836
- OOXML 内部のドキュメントを開き、編集、保存する 837
- OOXML 内部ドキュメントのインテリジェントな編集 837
- OOXML ファイル内部のアドレス 837
- OOXML アーカイブの比較 836

OOXML ファイルを作成、開き、保存する

OOXML ファイルは(**「ファイル | 新規作成」**コマンドから開始できる)新規ドキュメントを作成ダイアログから、目的のファイルの型(`.docx`、`.pptx`、`.xlsx`)を選択して、作成することができます。ファイル名とファイルを保存する場所を指定するよう促されます。指定された場所に新しいファイルが作成され、(以下のスクリーンショットにある)Archive View にて開かれます。OOXMLドキュメントの基本的な内部構造が既に作成されていることに注目してください。



XMLSpy のファイルを開くダイアログ(**「ファイル | 開く」**)により既存のOOXML ファイルを開くこともできます。OOXML の保存を行うには**「ファイル | 上書き保存」**コマンドを使用します。このコマンドによりOOXML ファイルの構造とルーションシップが保存されます。

OOXML ファイルの構造を編集する

Archive View 257 の機能を使用することでOOXML ファイルのコンテンツにフォルダーやドキュメントの追加、削除を行うことができます。これら構造上の修正を終えると、それらの変更を反映させるために**「ファイル | 上書き保存」**コマンドを使ってOOXML ファイルの保存を行う必要があります。以下の点に注意してください:

- 新規フォルダーまたは新規ドキュメントがArchive View のコマンドボタンから 257 追加された場合、作成時(追加時)にその名前を決定しなければなりません。Archive View でフォルダーまたはドキュメントの名前を変更することはできません。
- 新規ドキュメントがアーカイブフォルダーに追加されると、そのドキュメントを保存するか、OOXML ファイルそのものを保存することで、そのドキュメントが実際にアーカイブに保存されます。

OOXML 内部のドキュメントを開き、編集、保存する

Archive View で表示されている OOXML 内部のドキュメントをダブルクリックするか、メインウィンドウで選択し、[ドキュメントを開く](257)コマンドボタンをクリックすることでドキュメントを開くことができます。ドキュメントは XMLSpy のウィンドウで別に開かれます。ドキュメントの編集後は、ドキュメントを保存することで、ドキュメントが OOXML アーカイブファイル内に保存されます。

OOXML 内部ドキュメントのインテリジェントな編集

XMLSpy のインテリジェントな編集機能を OOXML ファイルパッケージ内の OOXML ドキュメントに対して使用することができます。インテリジェントな編集機能により提供される入力ヘルパー、自動補完、そして検証機能を使用することができます。

OOXML ファイル内部のアドレス

OOXML ファイル内のドキュメントのアドレスは、通常のファイルパスとパイプ文字（|）を組み合わせることで表現することができます。例えば、以下のファイルパスは:

C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples¥Office20XX¥ExcelDemo.xlsx|zip¥xl¥テーブル¥テーブル1.xml

XMLSpy サンプルフォルダーの `Examples/Office20XX` フォルダーにある `ExcelDemo.xlsx` OOXML ファイル内に格納されている xl¥テーブル フォルダーの テーブル.xml というファイルを表します。

OOXML アーカイブの比較

OOXML ファイルが Archive View で開かれている場合、「[ツール | ディレクトリを比較](1427)」コマンドを使用することで、他のアーカイブとの比較を行うことができます。

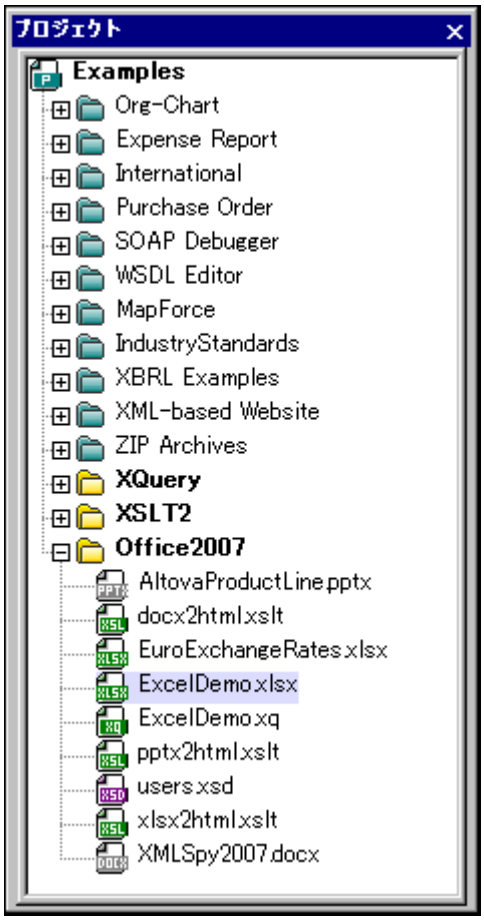
18.2 OOXML サンプルファイル

お使いのXMLSpy アプリケーションフォルダー以下にある Examples\Office フォルダーには、以下のサンプルファイルが配置されています:

- OOXML ファイル: (i) W ord Open XML ファイル(.docx)、(ii) Excel Open XML ファイル(.xlsx)、(iii) PowerPoint Open XML ファイル(.pptx)
- XSLT ファイル: (i) docx2html.xslt (サンプル .docx ファイルを HTML へ変換)、(ii) xslx2html.xslt (サンプル .xslx ファイルを HTML へ変換)、(iii) pptx2html.xslt (サンプル .pptx ファイルを HTML へ変換)
- XQuery ファイル: ExcelDemo.xq (.xslx ファイルからデータを取得する)

XSLT ならびに XQuery ファイルは OOXML ファイル内部のデータにアクセス、変換する際に XSLT や XQuery がどのように使用できるかのデモを行う目的で作成されています。XSLT や XQuery ドキュメントを実行するには、以下のどれかの方法を使用します:

- Archive View にて OOXML ファイルを開きます。フォルダービューにて、Archive を選択し、メニューコマンドから「XSL/XQuery | XSL 変換」を選択するか、「XSL/XQuery | XQuery 実行」を選択します。XSLT または XQuery ファイルを参照し、「OK」をクリックします。
- XMLSpy のプロジェクトウィンドウにて、(以下のスクリーンショットにあるように) Examples プロジェクトの Office2007 フォルダー以下にある .xlsx、.pptx または .docx ファイルを右クリックし、変換コマンドを選択します。変換を行うファイルを選択し、「OK」をクリックします。



- XMLSpy にて XSLT または XQuery ファイルを開き、メニューコマンドから「XSL/XQuery | XSL 変換」または「XSL/XQuery | XQuery 実行」をそれぞれ選択します。変換を行う XML を選択するよう促されるので、.docx、.xlsx、または .pptx ファイルを選択します。
- これらのオペレーションは Altova XSLT ならびに XQuery エンジンを搭載したスタンドアロンパッケージの AltovaXML を使っても行うことができます。

18.3 ZIP ファイル

Archive View[257] ではZIP ファイルを作成し、ファイル内部の構造を修正し、XMLSpy にてZIP パッケージに含まれるファイルを直接編集し、そのファイルをZIP アーカイブへ保存しなおすことができます。

ZIP ファイルの作成と保存

ZIP ファイルは新規ドキュメント作成ダイアログ(「**ファイル | 新規作成**」コマンド)にて .zip を選択することで作成することができます。(スクリーンショットにあるような)空のZIP アーカイブがXMLSpy で作成されます。その後「**ファイル | 上書き保存**」コマンドを使ってZIP ファイルを適切な場所に保存する必要があります。以下に記されるようにファイルやフォルダーを追加して、これらの変更を保存するためにZIP ファイルを保存します。

既存のZIP ファイルはXMLSpy のファイルを開くダイアログ(「**ファイル | 開く**」)から開くことができます。

注意: OOXML ファイルの作成時には、アーカイブがArchive View で開かれる前に保存場所を尋ねられるのに対して、ZIP ファイルの場合、作成時に保存場所を選択するよう促されることはありません。Archive View で開かれている空のアーカイブのZIP ファイルを保存するには、明示的に「**ファイル | 上書き保存**」コマンドを選択する必要があります。

フォルダーやファイルの追加とアーカイブ構造の修正

選択されたアーカイブフォルダーに対してフォルダー(**新規フォルダー**)や、既存のドキュメント(**ドキュメントを追加**)、そして新規ファイル(**新規ドキュメントを追加**)を追加することができます。新規フォルダーや新規ドキュメントを追加したときには、直ちにフォルダーまたはファイルの名前を入力する必要があります。Archive View にてフォルダーやファイルの名前の変更を行うことはできません。

ZIP ファイル内部のアドレス

ZIP ファイル内部のドキュメントは、通常のパスとパイプ文字(|)を組み合わせることで表現することができます。例えば、以下のファイルパスは:

```
C:¥Documents and Settings¥<ユーザー名>¥マイ ドキュメント¥Altova¥XMLSpy2021¥Examples¥Test.zip|zip\TestFolder\MyFile.xml
```

XMLSpy サンプルフォルダーの Examples フォルダーにある Test.zip ZIP ファイル内に格納されている TestFolder フォルダーの MyFile.xml というファイルを表します。

ZIP アーカイブの比較

ZIP ファイルがArchive View で開かれている場合、「ツール | ディレクトリの比較」[1427] コマンドを使用することで、別のアーカイブとの比較を行うことができます。

18.4 EPUB ファイル

EPUB ファイルとはデジタル形式の出版物（EPUB books）を配布するためのファイルの集合体のことです。Archive View[257] では EPUB ファイルを開き、EPUB ファイルの作成、デジタル EPUB ブックのプレビュー、EPUB アーカイブのコンポーネントファイルを XMLSpy にて直接編集し、EPUB ファイルを検証、コンポーネントを EPUB アーカイブへ保存し直すことができます。詳細については EPUB ファイル[842] のセクションを参照ください。

メモ: **(i)** XMLSpy では EPUB 2.0.1 がサポートされ、**(ii)** サンプル EPUB ファイルは `(My) Documents/Altova/XMLSpy2021/Examples` フォルダー内にある `Examples` プロジェクトにてご利用になれます。

用語解説

このセクションでは、以下のような用語が使用されます:

- **EPUB ファイル**とは、`.epub` というファイル拡張子をともなう EPUB ファイルのことです。ファイルそのものは ZIP ファイルで、Archive View にて開くことができる全てのファイルがアーカイブ内に含まれます。
- **アーカイブファイル**とは、EPUB アーカイブを含んでいるファイルのことです。
- **EPUB ブック**という用語は、アーカイブされた EPUB ファイルにより生成されたデジタル形式の出版物のことを指します。

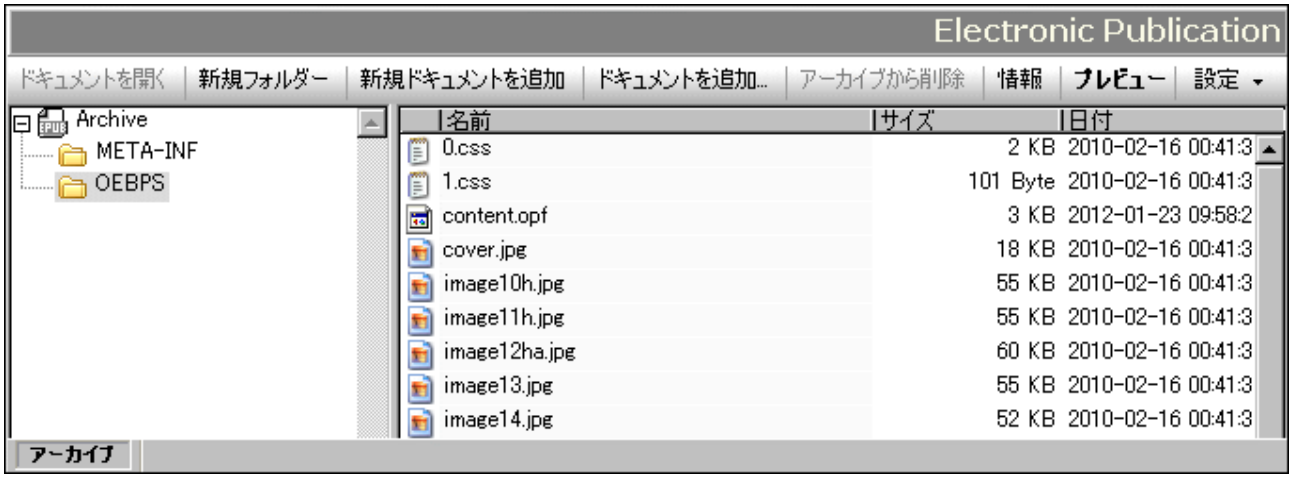
このセクションの構成

このセクションでは、XMLSpy における EPUB の機能が以下のように構成されます:

- Archive View にて EPUB ファイルを開く[842]
- 新規 EPUB ファイルを作成する[843]
- EPUB ブックのプレビュー[843]
- EPUB アーカイブのコンテンツや構造を修正する[844]
- 情報と設定[844]
- XMLSpy にてアーカイブファイルを編集する[844]
- アーカイブファイルの入力ヘルパー[844]
- EPUB ファイルの検証[844]

Archive View にて EPUB ファイルを開く

メニューコマンドから「ファイル | 開く」を選択し、EPUB ファイルを選択します。EPUB ファイルが Archive View にて表示されます（以下のスクリーンショットを参照）。Windows エクスプローラーにて EPUB ファイルを右クリックし、コンテキストメニューからファイルを XMLSpy にて開くこともできます。XMLSpy を EPUB ファイルのデフォルトエディターとすることで[1458]、EPUB ファイルをダブルクリックすることで、ファイルが Archive View にて開かれるようになります。



左側にはアーカイブ内にあるフォルダーが表示され、メインウィンドウには、フォルダービューにて選択されたフォルダー内にあるファイルが表示されます。EPUB アーカイブには以下のような構造とコンポーネントが表示されます。

```
Archive
|-- Mimetype ファイル
|
|-- META-INF フォルダー
|     |-- container.xml
|
|-- ドキュメントフォルダー（上のスクリーンショットでは OEBPS がドキュメントフォルダーとなります）
      |-- HTML、CSS、イメージファイル、そして OPF ならびに NCX ファイルが含まれます
```

新規 EPUB ファイルを作成する

新規にEPUB ファイルを作成するには、メニューコマンドから「**ファイル | 新規作成**」を選択します。新規ドキュメントの作成ダイアログにて、.epub というファイルの型を選択します。名前を付けて保存ダイアログにて、EPUB ドキュメントの名前を入力し、「**保存**」ボタンをクリックします。妥当な EPUB アーカイブを構成するために必要な全てのファイルが含まれるスケルトンの EPUB アーカイブが、Archive View 内の新規ウィンドウにて作成されます。以下に記される方法により、必要に応じてファイルやフォルダーをアーカイブへ追加して、EPUB ファイルを保存します。XMLSpy にて直接アーカイブファイルを編集するには、Archive View にあるファイルをダブルクリックします。新たな XMLSpy ウィンドウにてファイルが開かれます。ファイルの編集を行い、「**ファイル | 保存**」コマンドにより保存してください。

EPUB ブックのプレビュー

EPUB ブックをプレビューするには、Archive View にて EPUB ファイルがアクティブになっている状態で、Archive View のツールバーにてプレビューボタンをクリックします。EPUB ブックが XMLSpy 内にある（Internet Explorer）ブラウザーウィンドウにて開かれます。これらファイルのどれか－コンテンツファイルであれ構造に関連するファイルであれ－が変更され、まだ保存されていない場合、ファイルを保存するように促されます。変更点を保存しない場合、以前保存されたデータがプレビューには使用されます。Archive View のツールバーにある「**設定**」ボタンから、プレビューを行う前に、変更されたファイルを保存するように指定することができます。

以下の点に留意してください:

- EPUB データがプレビューウィンドウで開かれている時にアーカイブビューの「**プレビュー**」ボタンをクリックすると、EPUB データが既に開かれているプレビューウィンドウで再ロードされます。
- プレビューウィンドウ（Internet Explorer の再描画（F5）コマンドにより）を再描画しても、プレビューウィンドウそのものは更新されません。アーカイブビューの「**プレビュー**」ボタンを使用して、プレビューウィンドウ内の EPUB データを更新してください。
- プレビューウィンドウを閉じることで、プレビューを閉じることができます。

メモ: 全てのEPUB マークアップがInternet Explorer によりサポートされているわけではなく、プレビューのレイアウトが崩れることもあります。更に、デジタル出版物のドキュメントがHTML ではなくXML をベースにしたものである場合、プレビューが正常に行われない可能性もあります。最新バージョンのInternet Explorer ではEPUB マークアップの扱いが改善されており、問題が発生した場合は、最新バージョンのInternet Explorer へ更新することをおすすめします。

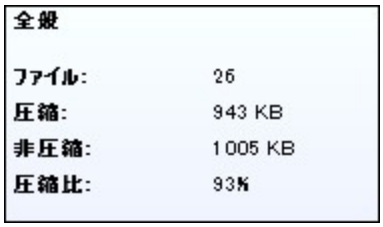
EPUB アーカイブのコンテンツや構造を修正する

選択されたアーカイブフォルダーに対して、フォルダーを追加(**「新規フォルダー」** ボタンをクリック)、新規ファイルを追加(**「新規ドキュメントの追加」** ボタン)、既存のファイルを追加(**「ドキュメントの追加」** ボタン)することができます。新規フォルダーや新規ドキュメントを追加する場合、そのファイルやフォルダーの名前を直ちに入力する必要があり、Archive View にてフォルダー名を変更することはできません。ファイルやフォルダーを選択して、**「アーカイブから削除」** ボタンをクリックすることで、ファイルやフォルダーを削除することができます。

アーカイブを変更したら、それら変更を反映させるために(**「ファイル | 保存」** コマンドにて)EPUB ファイルを保存する必要があります。

情報と設定

「情報」 ボタンをクリックすることで、Archive View の下部にアーカイブ情報の概要が表示されます(以下のスクリーンショットを参照)。再度 **「情報」** ボタンをクリックすることで、概要が非表示となります。概要には、アーカイブ内にある(Mimetype や `container.xml` を含む) ファイルの数や圧縮されたEPUB ファイルの大きさ、圧縮されていない状態のファイルの大きさが示されます。



「設定」 ボタンをクリックすることで、ファイルの自動保存オプションに対応したドロップダウンが表示されます。このコマンドにより、(i) 検証前、そしてEPUB ファイルを(XMLSpy の **「プレビュー」** ボタンにより) プレビューする前にファイルを自動保存するように指定することができます。

XMLSpy にてアーカイブファイルを編集する

アーカイブファイルをXMLSpy にて直接編集するには、Archive View に表示されているファイルをダブルクリックします。または、Archive View にて目的のファイルを選択し、Archive View のツールバーからドキュメントを開くボタンをクリックします。XMLSpy にてファイルが開かれます。ファイルを編集した後に、**「ファイル | 保存」** コマンドにてファイルを保存するようにしてください。

アーカイブファイルの入力ヘルパー

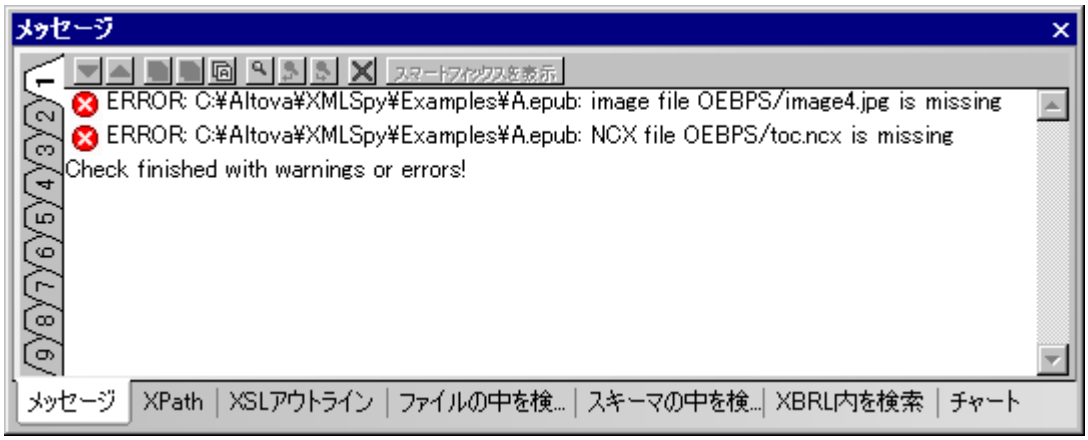
アーカイブファイルがXMLSpy にて開かれると、標準的なアーカイブファイルに対応した入力ヘルパーが表示されます。アーカイブファイルは以下のようなものになります:

- 通常 `content.opf` という名前のOPF ファイルにはEPUB ブックのメタデータが含まれます。Open Packaging Format (OPF) の仕様をベースにしています。
- 通常 `toc.ncx` という名前のNCX ファイル(XML に対するナビゲーションコントロールファイル)には、出版物の目次が含まれます。OPF 仕様のNCX パートをベースにしています。
- `META-INF` という名前のフォルダーには `container.xml` という名前のファイルが含まれており、ブック(OPF ファイル)のコンテンツを定義するファイルを指し示します。`container.xml` ファイルにより Open Container Format (OCF) 仕様のルールに従ったかたちでアーカイブファイルが構成されます。

EPUB ファイルの検証

EPUB ファイルを検証するには、**「XML | XML の検証」** コマンドを選択します。検証結果がメッセージウィンドウに表示されます(以下のスクリーンショットを参照)。アクティブなファイルのどれか(コンテンツファイルであれ構造に関連するファイルであれ)が変更され、まだ保存されていない場合、ファイルを保存するように促されます。EPUB ファイルの検証を行うには、変更されたファイルを保存する必要があります。

Archive View のツールバーにある「設定」ボタンから、プレビューを行う前に、変更されたファイルを保存するように指定することができます。



エラーメッセージには、(i) エラーが発見されたファイルの場所と、可能な場合はエラーが発生した行番号、(ii) エラーの詳細が表示されます。上のスクリーンショットでは、`content.opf` ファイルの21行目にてエラーが発生したことが示されています。メッセージウィンドウ内のエラー行をクリックすることで関連するファイルが開かれ、エラーの箇所がハイライトされます。

メモ: EPUB 検証エンジンは Java のユーティリティーで、検証エンジンを使用するにはお使いのコンピューターに Java がインストールされていなければなりません。

19 データベース

XMLSpy を使えば様々なデータベース（DB）へ接続し、DB に対して問い合わせを行ったり、DB の構造を XML スキーマとしてインポート、DB から XML データファイルを生成し、DB へデータをエクスポートすることができます。DB に関連した各機能は XMLSpy のメニューコマンドから扱うことができ、詳細はユーザーマニュアル[1127]に記述されています。これらのコマンドの完全なリストを、対応する記述へのリンクと共に以下に記します。

このセクションは、以下のように構成されます:

- データベースへの接続方法[847]: XMLSpy の DB に関するコマンドを実行する際に必要なオペレーションを説明します。
- テストにより XMLSpy で既に使用できることが確認されている DB のリストを示します[933]。

注意: 64ビット版の XMLSpy をお使いの場合は、DB への接続に使用されるデータベースドライバーも64ビットである必要があります。

XMLSpy の DB に関する機能

XMLSpy の DB に関する機能は「DB」[1305] ならびに「変換」[1331] メニューコマンドから利用できます。

- *データベースクエリの実行*[1305]: 「DB」メニューに存在します。DB の構造を別のデータベースクエリウィンドウにロードし、DB への問い合わせを行ないます。結果はデータベースクエリウィンドウに表示されます。
- *IBM DB2*[1318]: 「DB」メニューに存在します。DB2 は XML DB で、XMLSpy により XML DB の XML スキーマ管理や、XML DB の編集ならびに検証を行うことができます。
- *SQL Server*[1324]: 「DB」メニューに存在します。XMLSpy は DB の XML スキーマと編集、検証機能の管理を有効にします。
- *Oracle XML DB*[1326]: 「DB」メニューに存在します。XML スキーマ管理、データベースの問い合わせ、そして DB スキーマをベースにした XML ファイルの生成など、Oracle XML DB に対する数多くの機能を提供します。
- *データベースのデータをインポート*[1334]:「変換」メニュー内に存在します。DB データを XML ファイルへインポートします。
- *データベース構造から XML スキーマを作成*[1339]:「変換」メニュー内に存在します。DB の構造をベースにした XML スキーマを生成します。
- *XML スキーマを元にデータベースをインポート*[1343]:「変換」メニュー内に存在します。XMLSpy にて XML スキーマドキュメントがアクティブになっている状態で、DB の接続を確立し、選択された DB テーブルのデータをインポートします。その結果作成される XML ドキュメントは DB 接続を確立したときにアクティブになっている XML スキーマをベースとします。
- *XML スキーマを元にデータベース構造を作成*[1344]:「変換」メニュー内に存在します。既存の XML スキーマの構造をベースにした、データを持たない DB テーブルが作成されます。
- *データベースへエクスポート*[1350]:「変換」メニュー内に存在します。XML ドキュメントのデータを DB へエクスポートします。XML データを使って既存の DB テーブルを更新するか、XML データを含んだ新しいテーブルを作成することができます。

データ型の変換

XML ドキュメントと DB 間のデータ変換を行うにあたり、データの型も対応する形式に変換する必要があります。XMLSpy によるデータ型の変換方法は付録の DB により生成された XML のデータ型[2081] と XML スキーマから生成された DB のデータ型[2087] に収められています。

Altova DatabaseSpy

Altova の DatabaseSpy は複数のデータベースに対応した DB デザインツールで、XMLSpy にはない機能が提供されます。Altova DatabaseSpy に関する詳しい情報については、Altova ウェブサイトをご覧ください。

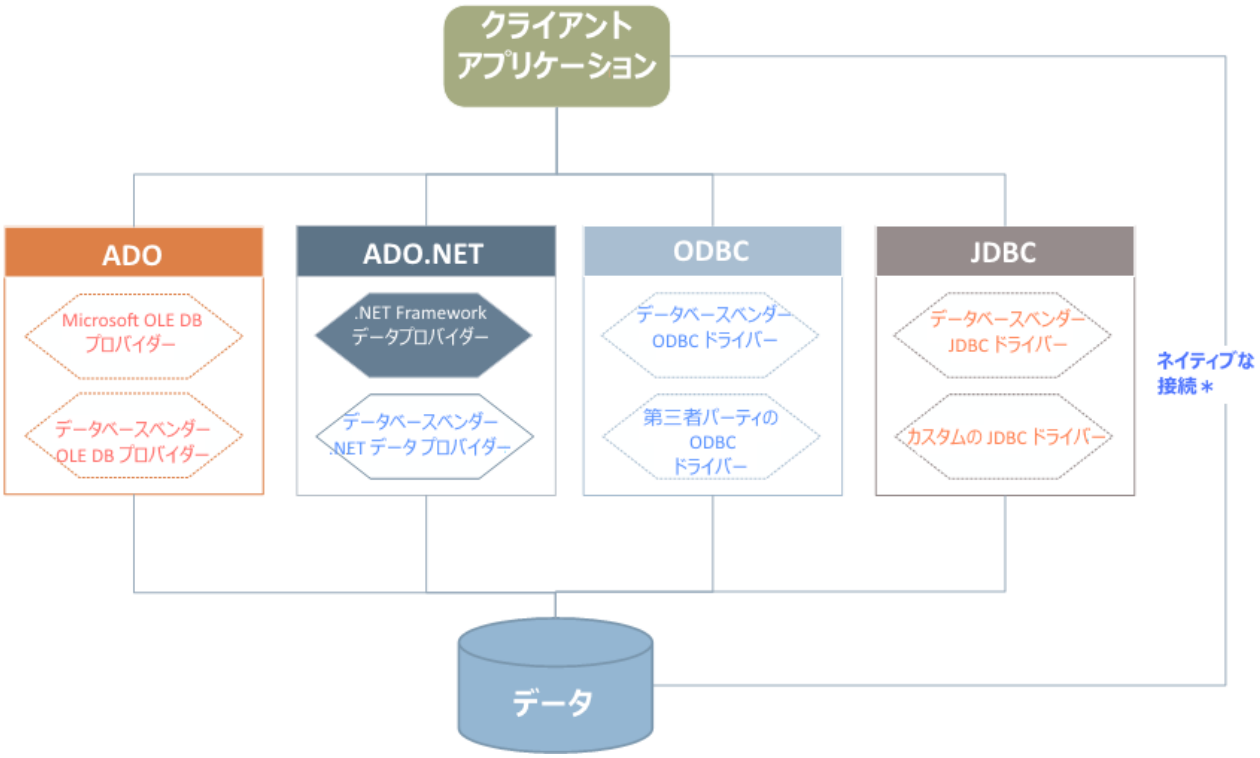
(C) 2015-2021 Altova GmbH

19.1　データソースへの接続

最も簡単なケースとして、データベースは、Microsoft Access、または、SQLite データベースファイルなどのローカルファイルが挙げられます。高度なケースの例としては、データベースは、アプリケーションが接続し、データを消費するオペレーティングシステムと同じオペレーティングシステムを使用するとは限らない、リモート、または、ネットワークデータベースサーバーに存在します。例えば、XMLSpy が、Windows で作動し、データ（例えばMySQL）へアクセスするデータベースがLinux マシーンで作動することは可能です。

多種のデータベース型とリモートおよびローカルで作業するには、XMLSpy は、データ接続インターフェイスとオペレーティングシステム内で使用することのできる、または、定期的にメジャーなデータベースベンダーにより リリースされるデータベースドライバーに従います。常に進化するデータベース技術の景観の中、このアプローチは、クロスプラットフォームに対する柔軟性および相互運用性に対応します。すなわち、XMLSpy により、以下のデータアクセス技術により主要なデータベースにアクセスすることができます

次の図は、最も簡単な方法で、（ジェネリックなクライアントアプリケーションとして表示されている）XMLSpy と（データベースサーバー、または、データベースファイルであることができる）データ保管の間で使用することのできるデータ接続のオプションを示しています。



＊直接なネイティブな接続 は、SQLite とPostgreSQL データベースのためにサポートされています。使用中のシステムにインストールするために追加ドライバーは必要ではありません。

上の図で示されるように、XMLSpy は、主なデータベース型に次のアクセス技術を使用してアクセスすることができます:

- 基となるOLE DB（Object Linking and Embedding, Database）プロバイダーを使用するADO（Microsoft® ActiveX® Data Objects)
- ADO.NET（データとのインタアクションを有効化するMicrosoft .NET Framework 内で使用することのできるライブラリのセット）
- JDBC（Java データベース接続）
- ODBC（Open データベース接続）

ADO.NET の一部はサポートされていない、または、サポートが制限されている場合があります。次を参照してください: ADO.NET サポートメモ[864]。

選択するデータベースへの接続のインターフェイスは、既存のソフトウェアのインフラストラクチャによります。データのアクセス技術と、接続するデータベースと密に統合することのできる、データベースドライバーを通常選択します。例えば、Microsoft Access 2013 データベースに接続するには、Microsoft Office Access データベース Engine OLE DB プロバイダー などのネイティブなプロバイダを使用する ADO 接続文字列を作成します。に接続するには、他方、最新の JDBC、ODBC、または ADO.NET インターフェイスを Oracle W eb サイトからダウンロードします。

W indows 製品 (Microsoft Access または SQL Server など) のためのドライバーは、W indows オペレーティングシステムで使用可能ですが、他のデータベースの種類では使用することができない可能性があります。メジャーなデータベースベンダーは、対応するデータベースに ADO、ADO.NET、ODBC、または JDBC の組み合わせからクロスプラットフォームアクセスをすることができる、一般使用可能なデータベースクライアントソフトウェアとドライバーを定期的にリリースします。更に、上記の技術のためのサードパーティーのドライバーを使用することができます。多くの場合は、1つ以上の方法で必要なデータベースへ使用中のオペレーティングシステムから接続することができます、この結果、XMLSpy からも接続することができます。使用可能な機能、パフォーマンスパラメーター、と既知の問題はデータアクセス技術および使用されるドライバーにより異なります。

19.1.1 接続ウィザードの開始

XMLSpy にはデータソースをセットアップするために必要とするステップを導くウィザードが搭載されています。

ウィザードのステップを開始する前に、データベースの一部の種類は、データベースドライバーまたはデータベースクライアントソフトウェアなどの複数のデータベースの必須条件を個別にインストールまたは構成する必要があることご注意してください。これらは、通常各データベースのベンダーにより提供され、特定の W indows バージョン用に調整されたドキュメントを含みます。データベースの種類別にグループ化されたデータベースドライバーのリストに関しては、をデータベースドライバーの概要[849]参照してください。

データベース接続ウィザードを開始するには次を行ってください:

- **DB** メニューから、「**データベースのクエリ**」をクリックします。

データベース型を選択し「次へ」をクリックすると データベースの種類、技術（ADO, ADO.NET, ODBC, JDBC）と使用されるドライバーにより異なる命令が画面に表示されます。

例えば、各データベースの種類に適用できるサンプルは[データベース接続のサンプル](877) のセクションを参照してください。各データベースへのアクセス技術に関しては以下のトピックを参照してください：

- [ADO 接続の設定](852)
- [ADO.NET 接続の設定](859)
- [ODBC 接続の設定](865)
- [JDBC 接続の設定](868)

19.1.2　データベースドライバー概要

以下のテーブルは、特定のデータベースに特定のデータアクセス技術を使用して接続するために よく使用されるデータベースのドライバーのリストです。このリストは、完全または規範的なものではありません。以下に表示されているドライバに加え他のネイティブまたはサードパーティー代替を使用することができます。

Windows オペレーティングシステムの複数のデータベースドライバーが既に使用されている場合がありますが、代替ドライバーをダウンロードする必要があります。データベースの一部に対して、オペレーティングシステムに付随するドライバーよりもデータベースベンダーより提供される最新のドライバーの パフォーマンスの方が良い場合があります。

データベースのベンダーは、個別にダウンロードすることのできる パッケージ、または、バンドルされたデータベースクライアントソフトウェアを提供します。後者の場合、データベースクライアントソフトウェアは、通常、データベースの管理と接続、およびデータベースクライアントとそのコンポーネントのインストールと構成を簡素化するために管理と構成ユーティリティにより構成されています。

データベースクライアントを正確に構成することは、データベースへの接続を成功させるために必須です。データベースをインストールして、使用する前に、各データベースバージョンおよび各 Windows バージョンにより通常異なるデータベースのインストールおよび構成の手順を確認することが奨励されます。

データベースの種類に関して各データアクセステクノロジーの機能と制限を理解するには、特定のデータベース製品のドキュメントを参照し、特定の環境に対しての接続をテストします。よく発生する接続問題を回避するために、以下に注意してください。:

- ADO.NET の一部のプロバイダーはサポートされない、または、サポートが制限されています。次を参照してください: ADO.NET サポートメモ[864]。
- データベースドライバーをインストールする際、Altova アプリケーション(32-ビット または64-ビット)などの同じプラットフォームが使用されることが奨励されます。例えば、32ビットのAltova アプリケーションが64-ビットオペレーティングシステムで使用される場合、が使用される場合、32-ビットのドライバーをインストールし、32-ビットのドライバーを使用して、データベースの接続をセットアップします。詳細に関しては、使用可能な ODBC ドライバーの表示[867] を参照してください。
- ODBC データソースをセットアップする際、データソース名 (DSN) をユーザー *DSN* の代わりに *システム DSN* として作成することが奨励されます。詳細に関しては、使用可能な ODBC ドライバーの表示[867] を参照してください。
- When setting
- JDBC データソースをセットアップする際、JRE (Java Runtime Environment) 、Java Development Kit (JDK) がインストールされている、またオペレーティングシステムのCLASSPATH 環境変数が構成されていることを確認してください。詳細に関しては、JDBC 接続のセットアップ[868]を参照してください。
- データベースベンダーからインストールするドライバーおよびデータベースクライアントソフトウェアのインストールの手順とサポートの詳細は、インストールパッケージのドキュメントを参照してください。公式なまたはサードパーティデータベースドライバーの場合、特定のオペレーティングシステムでの特定のドライバーに適用することのできる最も包括的な情報と構成手順は、通常ドライバーインストールパッケージの一部です。

データベース	インターフェイス	ドライバー
Firebird	ADO.NET	Firebird ADO.NET Data Provider (https://www.firebirdsql.org/en/additional-downloads/)
	JDBC	Firebird JDBC driver (https://www.firebirdsql.org/en/jdbc-driver/)
	ODBC	Firebird ODBC driver (https://www.firebirdsql.org/en/odbc-driver/)
IBM DB2	ADO	IBM OLE DB Provider for DB2
	ADO.NET	IBM Data Server Provider for .NET
	JDBC	IBM Data Server Driver for JDBC and SQLJ
	ODBC	IBM DB2 ODBC Driver
IBM DB2 for i	ADO	• IBM DB2 for i5/OS IBMDA400 OLE DB Provider • IBM DB2 for i5/OS IBMDARLA OLE DB Provider • IBM DB2 for i5/OS IBMDASQL OLE DB Provider
	ADO.NET	.NET Framework Data Provider for IBM i
	JDBC	IBM Toolbox for Java JDBC Driver
	ODBC	iSeries Access ODBC Driver
IBM Informix	ADO	IBM Informix OLE DB Provider

データベース	インターフェイス	ドライバー
	JDBC	IBM Informix JDBC Driver
	ODBC	IBM Informix ODBC Driver
Microsoft Access	ADO	• Microsoft Jet OLE DB Provider • Microsoft Access Database Engine OLE DB Provider
	ADO.NET	.NET Framework Data Provider for OLE DB
	ODBC	• Microsoft Access Driver
MariaDB	ADO.NET	MariaDB のための専用の.NET コネクタが不足している場合、MySQL のためのコネクタ/NET を使用してください（https://dev.mysql.com/downloads/connector/net/).
	JDBC	MariaDB Connector/J (https://downloads.mariadb.org/)
	ODBC	MariaDB Connector/ODBC (https://downloads.mariadb.org/)
Microsoft SQL Server	ADO	• Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) • Microsoft OLE DB Provider for SQL Server (SQLOLEDB) • SQL Server Native Client (SQLNCLI)
	ADO.NET	• .NET Framework Data Provider for SQL Server • .NET Framework Data Provider for OLE DB
	JDBC	• Microsoft JDBC Driver for SQL Server (https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server)
	ODBC	• ODBC Driver for Microsoft SQL Server (https://docs.microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-sql-server)
MySQL	ADO.NET	• Connector/NET (https://dev.mysql.com/downloads/connector/net/)
	JDBC	Connector/J (https://dev.mysql.com/downloads/connector/j/)
	ODBC	Connector/ODBC (https://dev.mysql.com/downloads/connector/odbc/)
Oracle	ADO	• Oracle Provider for OLE DB • Microsoft OLE DB Provider for Oracle
	ADO.NET	Oracle Data Provider for .NET (http://www.oracle.com/technetwork/topics/dotnet/index-085163.html)
	JDBC	• JDBC Thin Driver • JDBC Oracle Call Interface (OCI) Driver これらのドライバーは、通常、データベースクライアントのインストール中にインストールされます。（Thin ドライバーではない）OCI ドライバーを使用する接続は、Oracle XML DB のコンポーネントを使用しています。
	ODBC	• Microsoft ODBC for Oracle • Oracle ODBC Driver（通常、Oracle データベースクライアントのインストール中にインストールされます）

データベース	インターフェイス	ドライバー
PostgreSQL	JDBC	PostgreSQL JDBC Driver (https://jdbc.postgresql.org/download.html)
	ODBC	psqlODBC (https://odbc.postgresql.org/)
	Native Connection	使用可能。ネイティブな接続を使用する場合、ドライバーをインストールする必要はありません。
Progress OpenEdge	JDBC	JDBC Connector (https://www.progress.com/jdbc/openedge)
	ODBC	ODBC Connector (https://www.progress.com/odbc/openedge)
SQLite	Native Connection	使用可能。ネイティブな接続を使用する場合、ドライバーをインストールする必要はありません。
Sybase	ADO	Sybase ASE OLE DB Provider
	JDBC	jConnect™ for JDBC
	ODBC	Sybase ASE ODBC Driver
Teradata	ADO.NET	.NET Data Provider for Teradata (https://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata)
	JDBC	Teradata JDBC Driver (https://downloads.teradata.com/download/connectivity/jdbc-driver)
	ODBC	Teradata ODBC Driver for Windows (https://downloads.teradata.com/download/connectivity/odbc-driver/windows)

19.1.3 ADO 接続の設定

Microsoft ActiveX Data Objects (ADO) はOLE DB を介して多種のデータソースにデータベースアクセステクノロジーです。OLE DB はODBC またはJDBC への代替インターフェイスです。COM (コンポーネント オブジェクト モデル) 環境内での統一されたアクセスを提供します。ADO は新規のADO.NET 859 前のバージョンであり、他のデータソースのためにも使用可能ですがMicrosoft Access or SQL Server などのMicrosoft ネイティブなデータベースにアクセスを可能にする方法です。

需要な点は、複数のADO プロバイダー間から選択することができ、一部はダウンロードされワークステーションに使用する前にインストールされる必要があります。SQL Server への接続のために以下のADO プロバイダーを使用することができます:

- Microsoft OLE DB *Driver* for SQL Server (MSOLEDBSQL)
- Microsoft OLE DB *Provider* for SQL Server (SQLOLEDB)
- SQL Server Native Client (SQLNCLI)

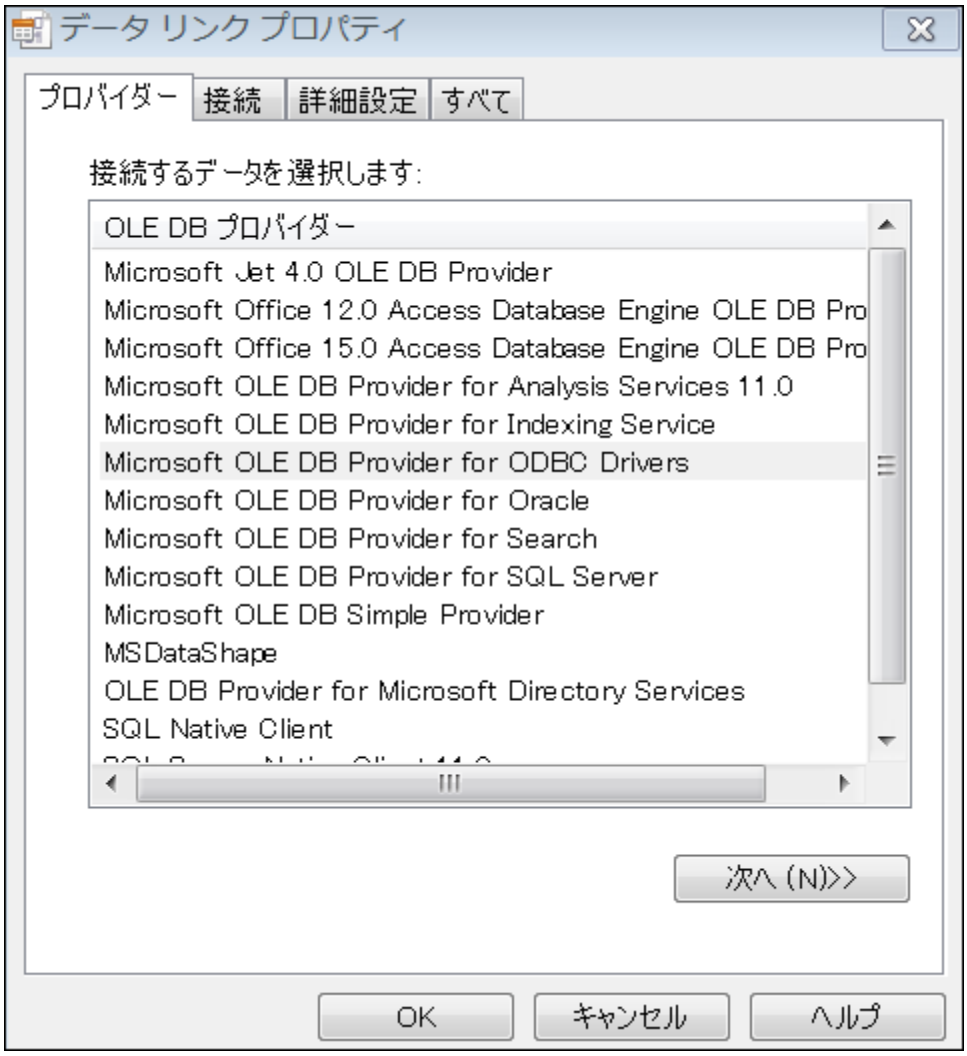
上記でリストされたプロバイダーで奨励されるプロバイダーはMSOLEDBSQL です。https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 からダウンロード可能です。XMLSpy (32-ビット または64-ビット) のプラットフォームに一致する必要があることご注意してください。SQLOLEDB とSQLNCLI プロバイダーは非奨励であると考えられています。

Microsoft OLE DB Provider for SQL Server (SQLOLEDB) は共通テーブル式（CTE）とネストされたSELECTステートメントなどの複雑なクエリのパラメーターバインディングに関する問題があることが知られています。

ADO 接続の設定:

1. データベース接続ウィザードを開始します[848]。
2. 「ADO 接続」をクリックします。

3. 構築] をクリックします。



4. 接続するデータプロバイダーを選択します。下のテーブルは一般的なシナリオをリストしています。

このデータベースに接続する方法...	このプロバイダーを使用する...
Microsoft Access	• **Microsoft Office Access Database Engine OLE DB Provider** (奨励) • **Microsoft Jet OLE DB Provider** **Microsoft Office Access Database Engine OLE DB Provider** がリスト内で使用できない場合 Microsoft Access または Microsoft Access Database Engine Redistributable (https://www.microsoft.com/en-us/download/details.aspx?id=54920) がコンピューターにインストールされていることを確認してください。
SQL Server	• **Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)** が OLE DB プロバイダーのために奨励されています。リストにこのプロバイダーが表示されるためには、プロバイダーが https://docs.microsoft.com/en-

このデータベースに接続する方法...	このプロバイダーを使用する..
	us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 からダウンロードされインストールされる必要があります。 • **Microsoft OLE DB Provider for SQL Server (OLEDBSQL)** • **SQL Server Native Client (SQLNCLI)**
他のデータベース	データベースに適用可能なプロバイダーを選択してください。 視聴中のデータベースへのOLE DB プロバイダーが使用できない場合 データベースベンダーから必要なドライバーをインストールします(データベースドライバー概要[849]を参照してください)。代わりにADO.NET、ODBC、またはJDBC 接続を設定します。 オペレーティングシステムが必要なデータベースのためにODBC ドライバーを必要とする場合 **Microsoft OLE DB Provider for ODBC Drivers** を使用することもできます。またはODBC 接続[865]を使用してください。

5. プロバイダーを選択後 [次へ] をクリックしてウィザードを完了します。

ウィザードの次のステップは選択されたプロバイダーにより異なります。SQL Server には、データベースサーバーのホスト名、認証メソッド、データベース名、およびデータベースユーザー名とパスワードを提供または選択する必要があります。サンプルに関してはMicrosoft SQL Server (ADO) への接続[901] セクションを参照ください。Microsoft Access に関しては、データベースファイルへのパスを参照または提供するように問われます。サンプルに関してはMicrosoft Access (ADO) への接続[898] セクションを参照ください。

初期化プロパティの完全なリスト（接続パラメーター）は接続ダイアログボックスの全てのタブで使用可能です。これらのプロパティは選択されたプロバイダーにより異なり、接続を可能にするために明示的に設定される必要がある場合があります。以下のセクションはMicrosoft Access とSQL Server データベースのための基本的な初期化プロパティを構成する方法について説明しています。

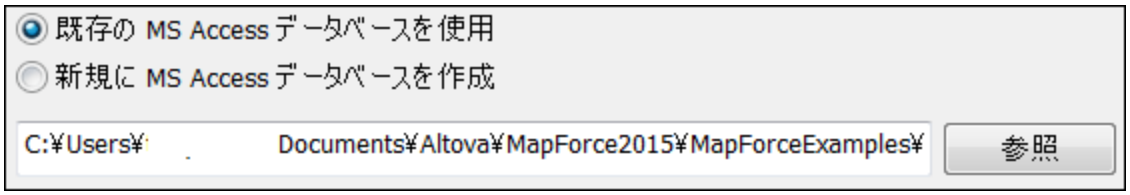
- SQL Server データリンクプロパティの設定[856]
- Microsoft Access データリンクプロパティの設定[857]

19.1.3.1 既存の Microsoft Access データベースへの接続

このアプローチは、パスワードで保護されていないMicrosoft Access データベースに接続する場合に適しています。データベースがパスワードにより保護されている場合、Microsoft Access (ADO) への接続[898] への接続で説明されている通りデータベースパスワードを設定してください。

既存のMicrosoft Access データベースへの接続方法:

1. データベース接続ウィザードを実行します(データベース接続ウィザードの開始[848]を参照)。
2. **Microsoft Access (ADO)** を選択して、「次へ」をクリックします。



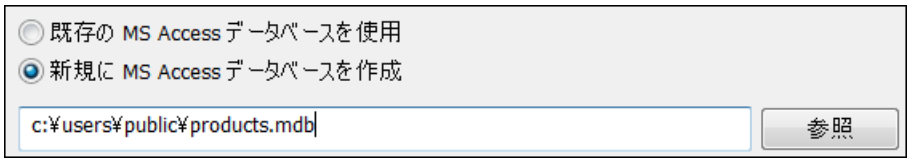
3. [既存の MS Access データベースを使用] を選択します。
4. データベースファイルを参照、または、(相対または絶対)パスを入力します。
5. 「接続」をクリックします。

19.1.3.2 新規の Microsoft Access データベースの作成

既存のデータベースファイルに接続する代替として、Microsoft Access が使用中のマシンにインストールされていない場合でも、新規の Microsoft Access データベースファイル(.accdb, .mdb) を作成して、接続することができます。XMLSpy により作成されるデータベースは空です。Microsoft Access または、DatabaseSpy (http://www.altova.com/ja/databasespy.html)などのツールを使用して、必要とされるデータベース構造を作成し、データを追加します。

新規の Microsoft Access データベースの作成方法:

1. データベース接続ウィザードを実行します(データベース接続ウィザードの開始 848 を参照)。
2. **Microsoft Access (ADO)** を選択して、「次へ」をクリックします。



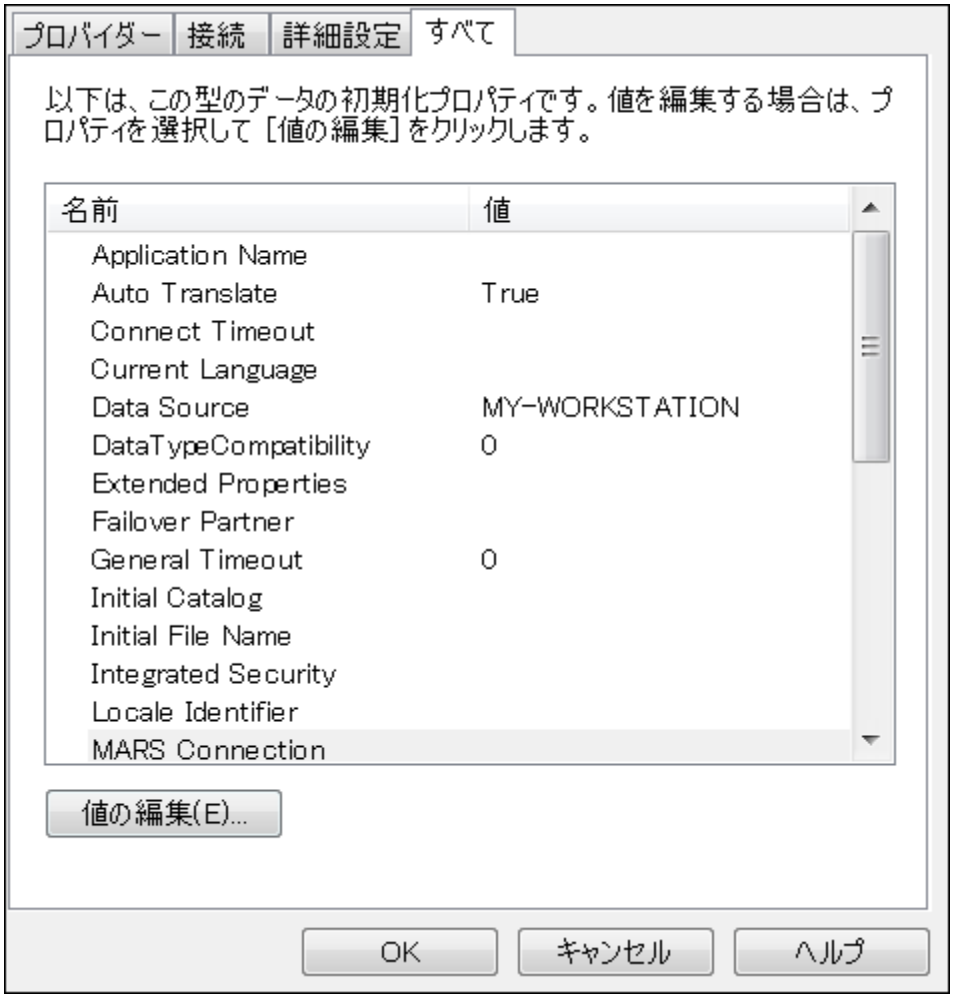
3. [新規 MS Access データベースの作成] を選択して、作成されるデータベースファイルの(相対または絶対)パスを入力します(例えば、**c:\users\public\products.mdb**)。または、[参照] をクリックして、フォルダーを選択して、ファイル名テキストボックスにデータベースファイルの名前(例えば、**products.mdb**)を入力し、[保存] をクリックします。

> **メモ**
> - データベースファイルを作成する場所に対して、書き込みアクセスの許可があることを確認してください。
> - データベースファイル名は **.mdb** or **.accdb** 拡張子を持つ必要があります。

4. 「接続」をクリックします。

19.1.3.3 SQL Server データ リンク プロパティの設定

Microsoft SQL Server データベースに ADO 852 を介して接続する場合、次のデータリンクプロパティダイアログボックスの全ての タブ内の次の接続プロパティを設定する必要があります。

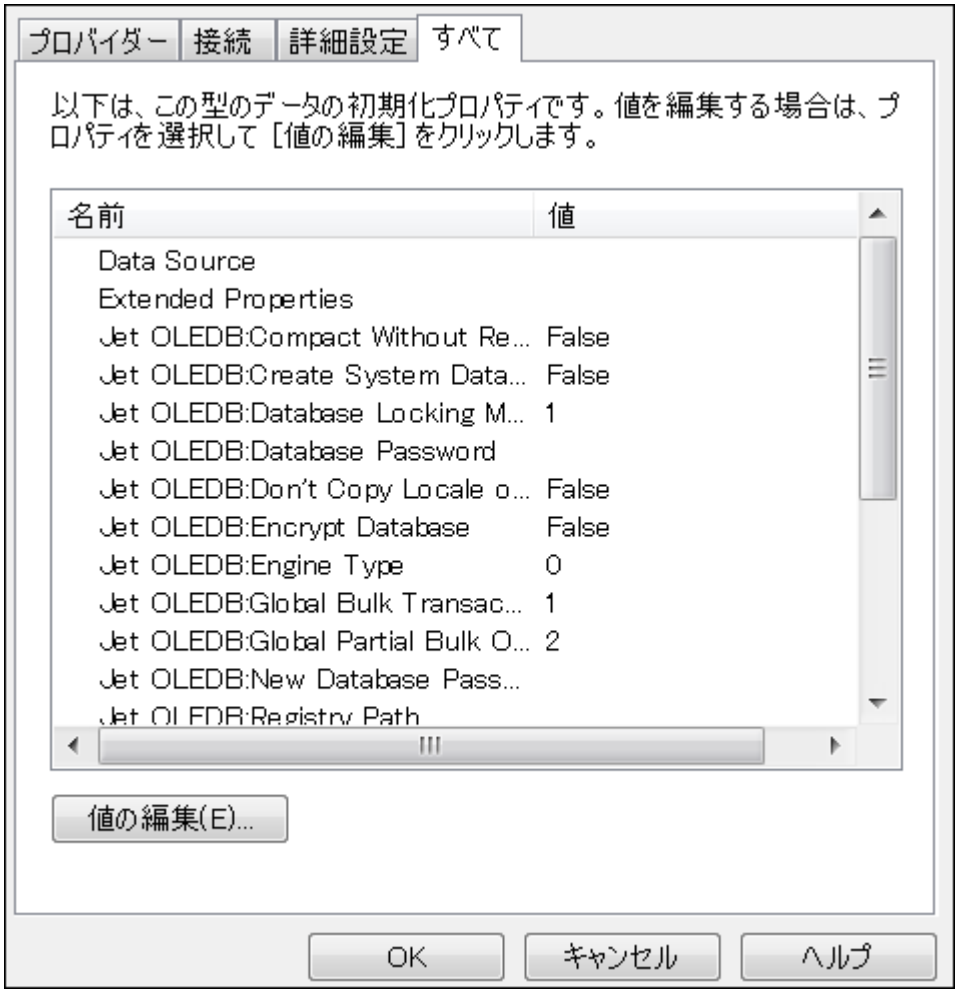


データ リンク プロパティ ダイアログ ボックス

プロパティ	メモ
統合セキュリティ	**プロバイダータブで SQL Server Native Client** がデータプロバイダーに選択された場合は、このプロパティを、空白文字に設定します。
セキュリティ情報を保持	このプロパティを**True** に設定する。

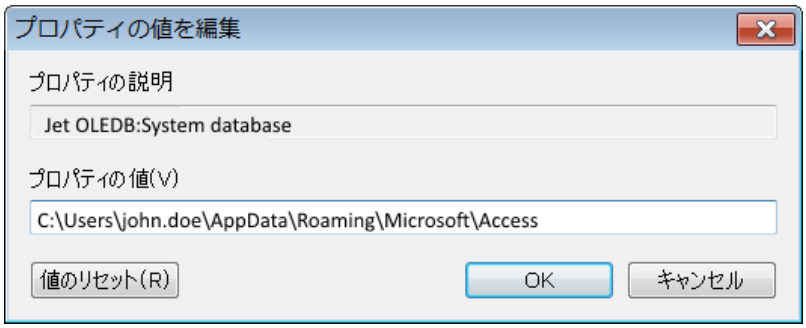
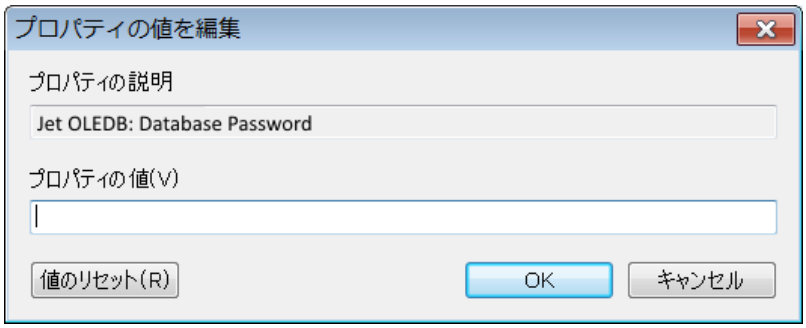
19.1.3.4 Microsoft Access データ リンク プロパティの設定

Microsoft Access データベースにADO[852] を介して接続する場合、次のデータリンク プロパティダイアログボックスの全ての タブ内の次の接続プロパティを設定する必要があります。



データリンクプロパティダイアログボックス

プロパティ	メモ
データソース	このプロパティは、Microsoft Access データベースファイルへのパスを保管します。データベースへの接続問題を回避するために、UNC（Universal Naming Convention 汎用名前付け規則）パスフォーマットの使用が奨励されます。例: `\\anyserver\share$\filepath`
Jet OLEDB: システムデータベース	このプロパティはワークグループ情報ファイルへのパスを保管します。データベースに接続する前に、このプロパティを明示的に設定する必要がある可能性があります。 "ワークグループ情報ファイル" エラーが理由で接続できない場合、ワークグループ情報ファイル（**System.MDW**）をユーザープロファイルに適用できるように設定します（手順は http://support.microsoft.com/kb/305542 を参照してください）、プロパティの値を **System.MDW** ファイルのパスに設定します。

プロパティ	メモ
	
Jet OLEDB: データベース パスワード	データベースがパスワードで保護されている場合は、このプロパティの値をデータベースのパスワードに設定します。 

19.1.4 ADO.NET 接続のセットアップ

ADO.NET は、データベースからのデータを含むデータと作業するためにデザインされた Microsoft .NET フレームワーク ライブラリのセットです。XMLSpy から、データベースに接続するには、ADO.NET、Microsoft .NET フレームワーク 4、または、以降を使用する必要があります。下に表示されているとおり、.NET プロバイダーを選択して、接続文字列を提供し、データベースに ADO.NET を介して接続します。

.NET データプロバイダーは、コマンドを実行する特定の型のデータソースへの接続を可能にするクラスのコネクションです（例えば、SQL Server または Oracle データベースなど）。すなわち、ADO.NET とは、XMLSpy などのアプリケーションがデータベースプロバイダーを介してデータベースと作動することを意味します。各 データプロバイダーは特定の型のデータソースと作動するように最適化されています。.NET プロバイダーには以下の2つの種類があります：

1. デフォルトとして Microsoft .NET フレームワークを使用して提供されます。
2. 主要なデータベースベンダーを使用して .NET フレームワークの拡張として提供されます。このような ADO.NET プロバイダーは、個別にインストールされている必要があり、通常それぞれのデータベースベンダーのウェブサイトからダウンロードすることができます。

メモ　特定の ADO.NET プロバイダーは、サポートされていない、または、サポートが制約されています。ADO.NET サポートに関するメモ[864]を参照してください。

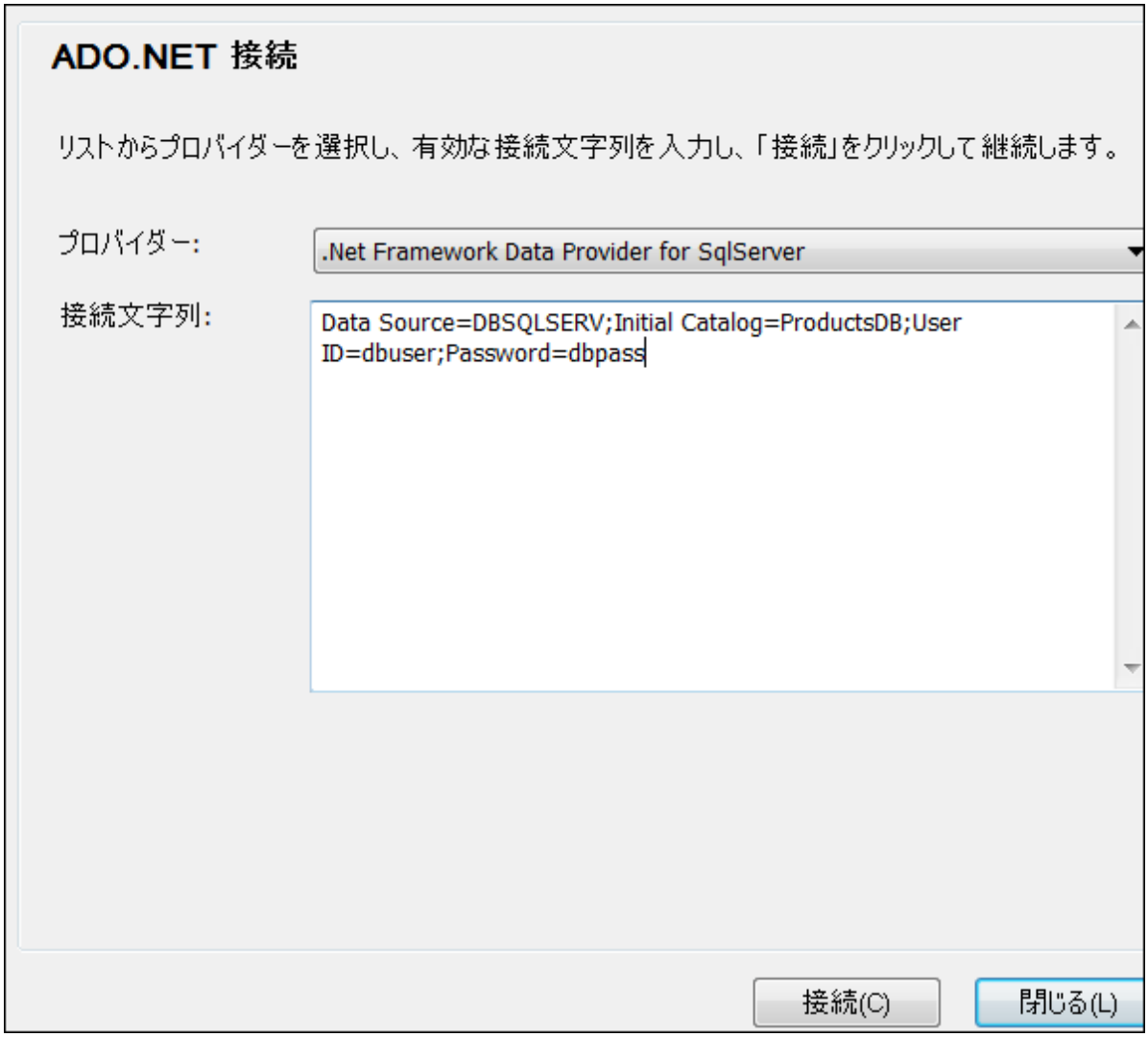
ADO.NET 接続の作成:

1. データベース接続ウィザードを開始します[848]。
2. 「ADO.NET 接続」をクリックします。
3. .NET データ プロバイダーをリストから選択します。

.NET フレームワークと共に使用することのできるプロバイダーのリストはデフォルトでプロバイダーのリストに表示されます。ベンダーに特有の.NET データプロバイダーは、使用中のシステムに既にインストールされている場合のみ使用することができます。.NET データプロバイダーは、データベースベンダーにより提供される、.msi、または、.exe ファイルを実行して、GAC (Global Assembly Cache) 内にインストールされる必要があります。

4. データベース接続文字列を入力します。接続文字列は、データベース接続情報をセミコロンで区切られた接続パラメーターのキー値ペアとして定義します。例えば、`Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;password=dbpass` などの文字列は、サーバー `DBSQLSERV` 上で、ユーザー名 `dbuser` とパスワード `dbpass` を使用して、SQL Server データベース `ProductsDB` に接続します。接続文字列を、キー値ペアを直接「接続文字列」ダイアログボックスに入力することにより作成することができます。他の接続オプションは Visual Studio で作成することができます (次を参照: Visual Studio 内で接続文字列を作成する[861])。

接続文字列の構文は、プロバイダーリストより選択されたプロバイダーにより異なります。例えば、次を参照してください: サンプル ADO.NET 接続文字列[863]。



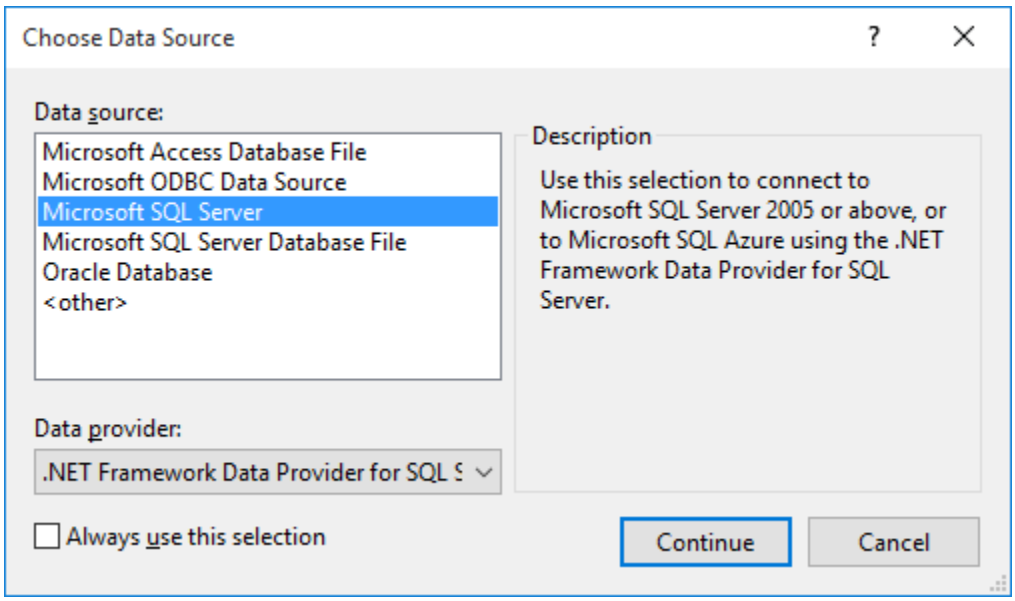
5. 「接続」をクリックします。

19.1.4.1 Visual Studio 内で接続文字列を作成する

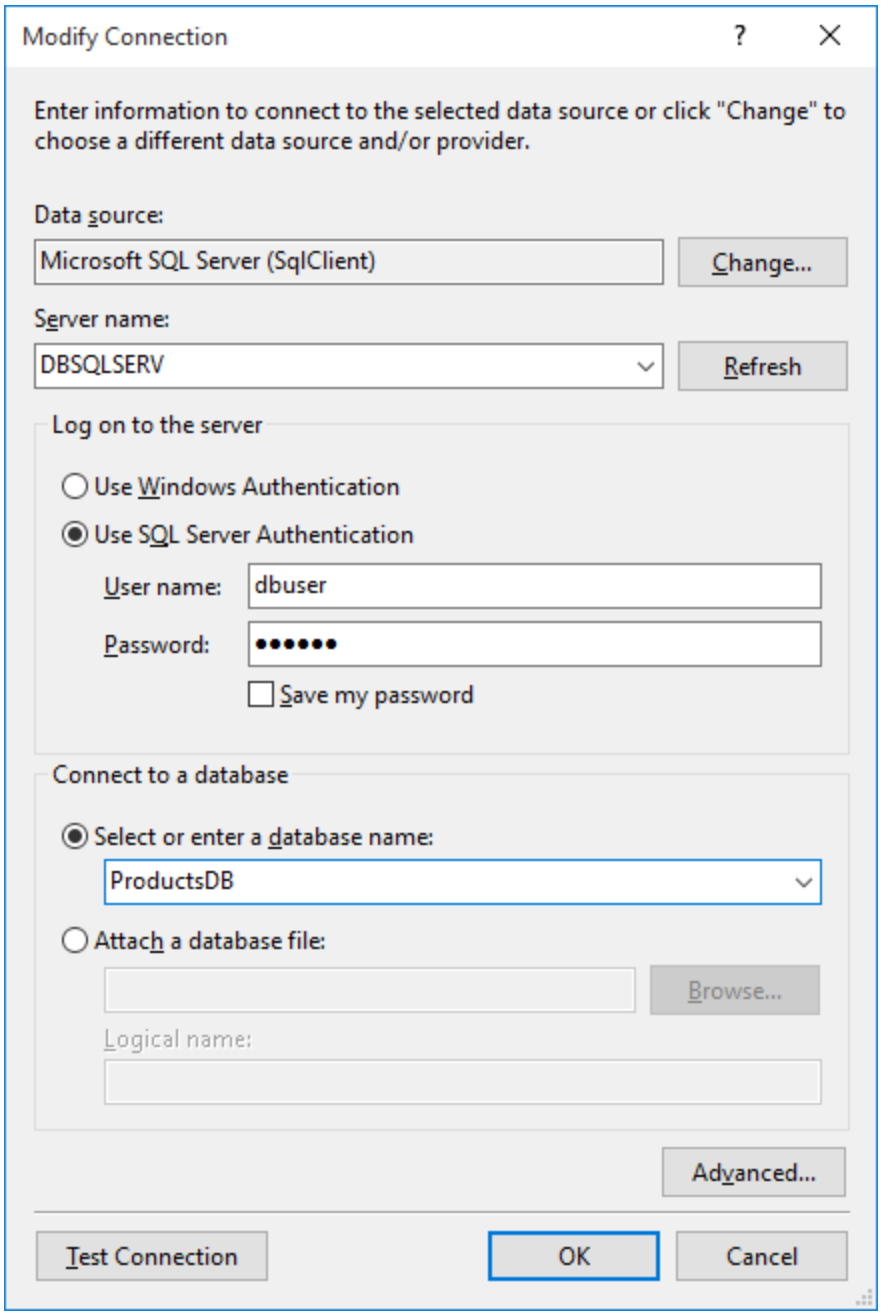
ADO.NET を使用してデータソースに接続するには、有効な接続文字列が必要です。次の手順は、Visual Studio から接続文字列を作成する方法を説明しています。

Visual Studio 内で接続文字列を作成する:

1. 「ツール」メニューから、「データベースへの接続」をクリックします。
2. リストからデータソースを選択します(この例の場合、Microsoft SQL Server)。データプロバイダーは、選択に従い自動的に記入されます。



3. 「接続」をクリックします。



4. データベースへのサーバーホスト名、ユーザー名、およびパスワードを入力します。この例の場合、サーバー `DBSQLSERV` 上のデータベース `ProductsDB` にSQL Server 認証を使用して接続します。

5. 「OK」をクリックします。

データベースへの接続に成功した場合、サーバーエクスプローラーウィンドウに表示されます。サーバーエクスプローラーウィンドウを、メニューコマンド「**表示 | サーバーエクスプローラー**」を使用して表示することができます。データベース接続文字列を取得するには、サーバーエクスプローラーウィンドウ内の接続を右クリックし、「**プロパティ**」を選択します。接続文字列が、Visual Studio プロパティウィンドウに表示されます。XMLSpy の「接続文字列」ボックスに文字列を貼り付ける前に、アスタリスク（*）文字を実際のパスワードと置き換える必要があることにご注意してください。

19.1.4.2 サンプル ADO.NET 接続文字列

ADO.NET 接続をセットアップするには、データベース接続ダイアログボックスからADO.NET プロバイダーを選択し、接続文字列を入力します。(次も参照してください ADO.NET 接続のセットアップ[859])。異なるデータベースのサンプルADO.NET 接続文字列は.NET プロバイダーの下にリストされています。

Teradata のための.NET Data

このプロバイダーは、次のTeradata W eb サイトからダウンロードすることができます (http://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata)。サンプル接続は以下のとおりです:

```
Data Source=ServerAddress;User Id=user;Password=password;
```

IBM i のための.NET Framework データプロバイダー

このプロバイダーは、*IBM i Access Client Solutions - Windows Application Package* の一部としてインストールされます。サンプル接続文字列は以下のとおりです:

```
DataSource=ServerAddress;UserID=user;Password=password;DataCompression=True;
```

詳細に関しては、インストールパッケージ内に含まれる「.NET プロバイダー技術レファレンス ヘルプファイル」を参照してください。

MySQL のための.NET Framework データプロバイダー

このプロバイダーをMySQL ウェブサイトからダウンロードすることができます(https://dev.mysql.com/downloads/connector/net/)。サンプル接続文字列は以下のとおりです:

```
Server=127.0.0.1;Uid=root;Pwd=12345;Database=test;
```

次も参照してください: https://dev.mysql.com/doc/connector-net/en/connector-net-programming-connecting-connection-string.html

SQL Server のための.NET Framework データプロバイダー

サンプル接続文字列は以下のとおりです:

```
Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;Password=dbpass
```

次も参照してください: https://msdn.microsoft.com/en-us/library/ms254500(v=vs.110).aspx

.NET Framework 4.0 のための IBM DB2 データプロバイダー 10.1.2

```
Database=PRODUCTS;UID=user;Password=password;Server=localhost:50000;
```

メモ このプロバイダーは、通常 IBM DB2 Data Server Client パッケージと共にインストールされます。IBM DB2 Data Server Client パッケージをインストール後 ADO.NET プロバイダーのリスト内でプロバイダーが不足している場合、次の技術メモを参照してください: http://www-01.ibm.com/support/docview.wss?uid=swg21429586。

次も参照してください:
http://www.ibm.com/support/knowledgecenter/en/SSEPGG_10.1.0/com.ibm.swg.im.dbclient.adonet.ref.doc/doc/DB2ConnectionClassConnectionStringProperty.html

.NET (ODP.NET) のための Oracle データプロバイダー

ODP.NET プロバイダーを含むインストールパッケージを Oracle ウェブサイトからダウンロードすることができます (次を参照してください: http://www.oracle.com/technetwork/topics/dotnet/downloads/index.html)。サンプル接続文字列は以下のとおりです:

```
Data Source=DSORCL;User Id=user;Password=password;
```

Oracle への接続(ODBC) 912 内で説明されているように **tnsnames.ora** ファイル内で定義された Oracle サービス名をさす DSORCL がデータソース名の場合。

tnsnames.ora ファイル内のサービス名を構成せず接続する場合は、以下のような文字列を使用してください:

```
Data Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=host)(PORT=port)))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=MyOracleSID)));User
Id=user;Password=password;
```

次も参照してください: https://docs.oracle.com/cd/B28359_01/win.111/b28375/featConnecting.htm

19.1.4.3 ADO.NET サポートに関するメモ

下のテーブルは XMLSpy 内で現在サポートされていない、または、制限のあるサポートのある既知の ADO.NET データベースドライバーがリストされています。

データベース	ドライバー	サポートノート
全てのデータベース	ODBC のための Net Framework **Data Provider**	制限つきサポート。Microsoft Access 接続の際に既知の問題があります。ODBC への直接の接続を代わりに使用することが奨励されます。
	OleDb のための .Net Framework **Data Provider**	制限つきサポート。Microsoft Access 接続の際に既知の問題があります。ADO への直接の接続を代わりに使用することが奨励されます。
Firebird	**Firebird ADO.NET Data Provider**	制限つきサポート。ODBC または JDBC を代わりに使

データベース	ドライバー	サポートノート
		用することが奨励されます。
Informix	.NET Framework 4.0 のための IBM Informix Data Provider	サポートされていません。**DB2 Data Server Provider** を代わりに使用してください。
IBM DB2 for i (iSeries)	i5/OS のための .Net Framework Data Provider	サポートされていません。*IBM i Access Client Solutions - Windows Application Package* の一部として提供されている **.Net Framework Data Provider for IBM i** を代わりに使用してください。
Oracle	Oracle のための .Net Framework Data Provider	制限つきサポート。.NET Framework と共にこのドライバは提供されていますが、Microsoft により使用を奨励されていません。
PostgreSQL	-	このベンダーのための ADO.NET ドライバはサポートされていません。ネイティブな接続を代わりに使用してください。
Sybase	-	このベンダーのための ADO.NET ドライバはサポートされていません。

19.1.5 ODBC 接続の設定

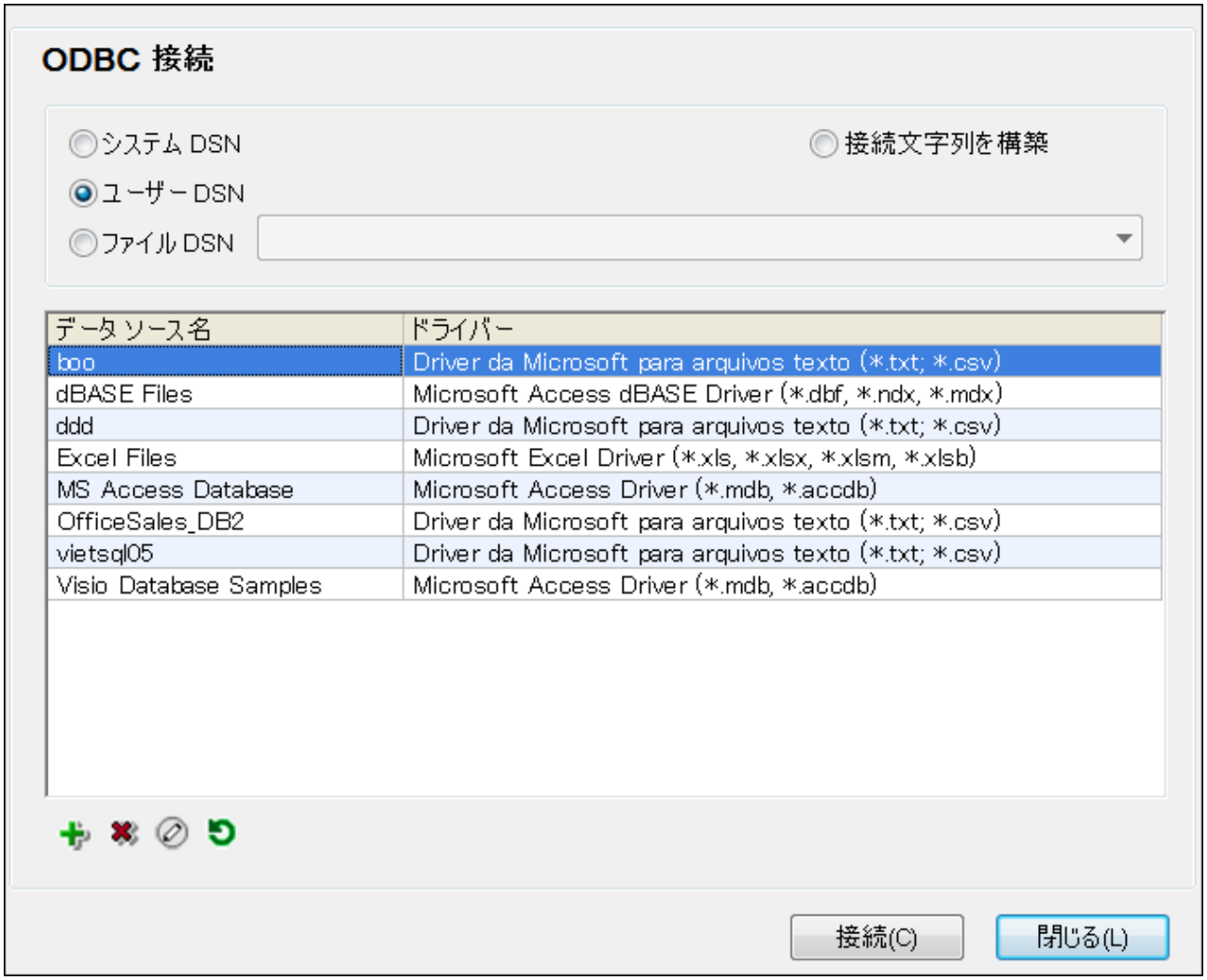
ODBC (Open Database Connectivity) は、一般に使用されているデータアクセス技術で、XMLSpy からデータベースに接続することができます。データベースにアクセスするための主要な手段として、またはネイティブな OLE DB- や JDBC-による接続の代替として使用することができます。

ODBC を使用してデータベースに接続するには、最初に ODBC データソース名(DSN) を作成する必要があります。このステップは、他のユーザーまたはオペレーティングシステムにより DSN が既に作成されている場合必要ありません。DSN は、統一された、XMLSpy を含むオペレーティングシステム上の ODBC に対応したクライアントアプリのデータベース接続方法を表します。DSN には以下の種類があります:

- ファイル DSN
- ユーザー DSN
- ファイル DSN

オペレーティングシステムで特権を持つすべてのユーザーはシステム データ ソースにアクセスすることができます。作成したユーザーのみがユーザー データ ソースにアクセスすることができます。*ファイル DSN* を作成した場合、データソースは .dsn 拡張子を持つファイルとして作成され、データソースにより使用されているドライバーがインストールされていることが前提で、他のユーザーと共有することができます。

ODBC 接続ダイアログボックス内の [**ODBC 接続**] をクリックすると、使用中のコンピューターで既に使用することのできる DSN はデータベース接続ダイアログボックスに表示されています。



ODBC 接続ダイアログボックス

データベースにより必要とされるDSN が無い場合、XMLSpy データベース接続ウィザードが作成の手助けをします。しかし、W indows オペレーティングシステムに直接作成することもできます。いずれにせよ、続行する前に、使用中のデータベースにODBC ドライバーがオペレーティングシステムのODBC ドライバーリストに適用できることを確認してください。(使用可能な ODBC ドライバーの表示[867] を参照)。

新規のDSNを使用しての接続:

1. データベース接続ウィザードを開始します[848] 。
2. データベース接続ダイアログボックスの[ODBC 接続] をクリックします。
3. システムDSN を作成するには、オペレーティングシステムの管理者権限が必要です。

   > また、XMLSpy が管理者として作動している必要があります。

4. 「追加」 をクリックします。

5. ドライバーを選択して、(作成するDSN の種類に応じて) [ユーザーDSN] または[システムDSN] をクリックします。使用中のデータベースに適用することのできるドライバーがリストされていない場合、データベースベンダーからダウンロードして、インストールしてください。(次を参照: データベースドライバー概要)。
6. ポップアップするダイアログボックスに特定の接続情報を入力してセットアップを完了します。

接続を成功させるには、データベースサーバーのホスト名(またはIP アドレス)、とデータベースのユーザー名およびパスワードを提供する必要があります。他の接続パラメーターがある可能性がありますが、これらのパラメーターはデータベースプロバイダーにより異なります。各接続方法に固有のパラメーターについての詳細は、ドライバープロバイダーのドキュメントを参照してください。一度作成されると、DSN はデータソース名のリストで使用することができます。これによりデータベース接続の詳細をデータベースの接続を希望する際いつでも使用することができます。ユーザーDSN はユーザーDSN のリストに追加されますが、システムDSN はシステムDSN のリストに追加されます。

既存のDSNを使用しての接続:

1. データベース接続ウィザードを開始します[848]。
2. [ODBC 接続] をクリックします。
3. 既存のデータソース(ユーザーDSN、システムDSN、ファイルDSN) から種類を選択します
4. 既存 DSN レコードをクリックして、「接続」をクリックします。

既存の.dsn ファイルをベースにして接続文字列を構築する:

1. データベース接続ウィザードを開始します[848]。
2. [ODBC 接続] をクリックします。
3. [接続文字列の構築] を選択して [構築] をクリックします。
4. ファイルDSN を使用して、接続文字列を構築する場合、[ファイルデータソース] タブをクリックします。それ以外の場合は、[マシンデータソース] タブをクリックします。(システムDSN とユーザーDSN は"マシン" データソースとして知られています。)
5. 必要とされる.dsn ファイルを選択して、「OK」 をクリックします。

用意された接続文字列を使用して接続する:

1. データベース接続ウィザードを開始します[848]。
2. [ODBC 接続] をクリックします。
3. [接続文字列の構築] を選択しています。
4. 接続文字列を与えられたボックスに貼り付け、「接続」 をクリックします。

19.1.5.1 使用可能な ODBC ドライバーの表示

ODBC データソース管理で使用中のオペレーティングシステムで使用可能な ODBC ドライバーを確認することができます。W indows コントロールパネルの管理ツールからODBC データソース管理 (**Odbcad32.exe**) にアクセスすることができます。64-ビット版オペレーティングシステムには、2 種類のバージョンの実行可能ファイルがあります:

- **C:\Windows\SysWoW64** ディレクトリにある、**Odbcad32.exe** ファイルの32-ビット版バージョン(**C:** が使用中のシステムドライブとして前提されています)。
- **C:\Windows\System32** ディレクトリにある、**Odbcad32.exe** ファイルの64-ビット版バージョン

インストールされている32-ビット版データベースドライバーは、ODBC データソース管理の32-ビット版バージョンで、64-ビット版バージョン内の64-ビット版ドライバーで確認することができます。ですから、対応するバージョンのODBC データソース管理でデータベースドライバーを確認してください。

ODBC データソース管理者

ターゲットデータベースのドライバーがリストに存在しない場合、または、代替ドライバーを追加する場合、データベースのベンダーからダウンロードしてください(データベースドライバー概要[849] を参照)。ODBC ドライバーがシステムで使用可能になると、ODBC 接続を作成することができます(ODBC 接続のセットアップ[865] を参照)。

19.1.6 JDBC 接続の設定

JDBC (Java Database Connectivity) は、Oracle の Java ソフトウェアプラットフォームの一部であるデータベースアクセスインターフェイスです。JDBC 接続は ODBC 接続に比べ、通常より多くリソースを集中的に使用する接続ですが、ODBC で使用することのできない機能を使用することができます。ODBC コネクタにより使用できないデータベース機能を使用する場合 JDBC 接続を使用することが奨励されています。

前提条件

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- データベースベンダーからの JDBC ドライバーがインストールされている必要があります。これは使用可能でありデータベースによりサポートされている場合、データベースクライアントインストールの一部として、または、JDBC ライブラリ (.jar ファイル) として個別にダウンロードされている JDBC ドライバーの可能性があります。データベース接続のサンプル[877] も参照してください。
- Oracle データベースに接続する場合は、一部の Oracle ドライバーは特定の JRE バージョンに固有のため、追加コンポーネントと構成が必要な点に注意してください。Oracle 製品のドキュメント (例えば、「JDBC 開発者ガイドおよびリファレンス」) には、各 JDBC ドライバーの構成の手続きの詳細が説明されています。

- CLASSPATH 環境変数は、(.jar ファイルの1つまたは複数の) 使用中のWindows オペレーティングシステムのJDBC ドライバーのパスを含む必要があります。データベースクライアントをインストールする際に、インストーラーはこの変数を自動的に構成する可能性があります (CLASSPATH の構成 870 を参照)。

Windows 資格情報を使用してJDBC を介してSQL Server に接続する方法

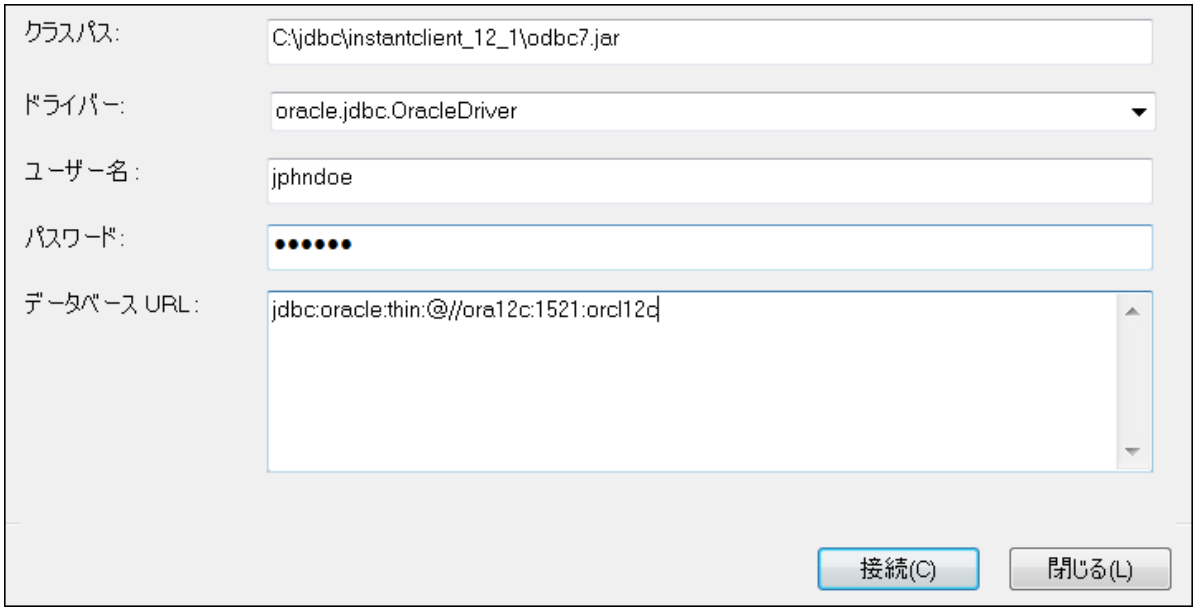
Windows 資格情報 (統合されたセキュリティ) を使用してJDBC を介してSQL Serverに接続する場合、以下に注意してください:

- JDBC ドライバーパッケージに含まれている **sqljdbc_auth.dll** ファイルはシステムPATH 環境変数上のディレクトリにコピーされる必要があります。このようなファイルは2個存在します。1つはx86 プラットフォームのために、もう1つはx64 プラットフォームのために存在します。JDK プラットフォームに対応するPATH が追加されていることを確認してください。
- JDBC 接続文字列はプロパティ **integratedSecurity=true** を含む必要があります。このプロパティを異なる場所から追加することができます:
  - データベース接続ウィザードから追加することができます。以下を参照してください。
  - データベースコンポーネント設定から追加することができます。
  - 適応可能な場合、生成済みのJava コード内のデータベース接続文字列の編集をすることにより追加することができます。

詳細に関しては *SQL Server のための Microsoft JDBC ドライバー* ドキュメントを参照してください。
https://docs.microsoft.com/en-us/sql/connect/jdbc/building-the-connection-url。

JDBC 接続のセットアップ方法

1. データベース接続ウィザードを開始します。 848
2. 「JDBC 接続」をクリックします。
3. 任意で、セミコロンで区切られた.jar ファイルパスのリストを「クラスパス」テキストボックスに入力することもできます。ここで入力される.jar ライブラリは、CLASSPATH 環境変数内で既に定義されているものに加え、環境にロードされます。「クラスパス」テキストボックスの編集が完了すると、ソース.jar ライブラリ内で検索されるJDBC ドライバーは、「ドライバー」リストに自動的に追加されます (次のステップを参照してください)。



4. 「ドライバー」の横でJDBC ドライバーをリストから選択、または、Java クラス名を入力します。CLASSPATH 環境変数により構成されたJDBC ドライバーおよび、「クラスパス」テキストボックス内で検出されたドライバーが含まれます (次を参照: CLASSPATH の構成 870)。

CLASSPATH 変数内で定義されているJDBC ドライバーのパスを使用することができます。また、データベース接続ダイアログボックスに直接入力された .jar パスはJava Virtual Machine (JVM) によりサポートされています。JVM は接続を構築するためのドライバーを決定します。JVM にロードされるJava クラスを確認し、JDBC ドライバーの競合、および、データベースに接続する際に発生する予期されないエラーを回避するようにしてください。

5. 対応するボックスに、ユーザー名とパスワードを入力します。
6. データベースURL テキストボックス内に、JDBC 接続 URL (文字列) をデータベースの型のとおり入力します。次のテーブルは一般のデータベースの型のためのJDBC 接続 URL (文字列) の構文について説明しています。

データベース	JDBC 接続 URL
Firebird	jdbc:firebirdsql://**<host>**[:**<port>**]/**<database path or alias>**
IBM DB2	jdbc:db2://**hostName**:**port**/**databaseName**
IBM DB2 for i	jdbc:as400://**[host]**
IBM Informix	jdbc:informix-sqli://**hostName**:**port**/**databaseName**:INFORMIXSERVER=**myserver**
MariaDB	jdbc:mariadb://**hostName**:**port**/**databaseName**
Microsoft SQL Server	jdbc:sqlserver://**hostName**:**port**;**databaseName**=name
MySQL	jdbc:mysql://**hostName**:**port**/**databaseName**
Oracle	jdbc:oracle:thin:@**hostName**:**port**:**SID** jdbc:oracle:thin:@//**hostName**:**port**/**service**
Oracle XML DB	jdbc:oracle:oci:@//**hostName**:**port**:**service**
PostgreSQL	jdbc:postgresql://**hostName**:**port**/**databaseName**
Progress OpenEdge	jdbc:datadirect:openedge://**host**:**port**;databaseName=**db_name**
Sybase	jdbc:sybase:Tds:**hostName**:**port**/**databaseName**
Teradata	jdbc:teradata://**databaseServerName**

メモ 上記のフォーマットに対する構文のバリエーションも使用することができます (例えば、データベースURL はポートを除外、またはデータベースのユーザー名とパスワードを含む可能性があります)。詳細はデータベースのベンダーのドキュメントを参照してください。

7. 「接続」をクリックします。

19.1.6.1 CLASSPATH の構成

CLASSPATH 環境変数はクラスと、オペレーティングシステム上の他のリソースファイルを検索するためJava Runtime Environment (JRE) またはJava Development Kit (JDK) により使用されます。JDBC を介してデータベースに接続すると、この変数はオペレー

ティングシステム上のJDBC ドライバーへのパス、または一部の場合、使用中のデータベースの種類に固有の追加ライブラリファイルへのパスを含むように構成されなければなりません。

下のテーブルは、CLASSPATH 変数に通常含まれる必要のあるサンプルファイル パスをリストしています。重要なことは、システム上の情報をシステム上のJDBC ドライバーのロケーション、JDBC ドライバー名、およびオペレーティングシステム上に存在するJRE/JDK バージョンをベースにしたこの情報を調整する必要があります。接続問題を避けるために、使用中のオペレーティングシステムにインストールされているJDBC ドライバに適用されるインストールの指示、インストール前とインストール後の構成ステップを確認してください。

データベース	サンプルCLASSPATH エントリ
Firebird	C:\Program Files\Firebird\Jaybird-2.2.8-JDK_1.8\jaybird-full-2.2.8.jar
IBM DB2	C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc.jar;C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc_license_cu.jar;
IBM DB2 for i	C:\jt400\jt400.jar;
IBM Informix	C:\Informix_JDBC_Driver\lib\ifxjdbc.jar;
Microsoft SQL Server	C:\Program Files\Microsoft JDBC Driver 4.0 for SQL Server\sqljdbc_4.0\enu\sqljdbc.jar
MariaDB	**<installation directory>**\mariadb-java-client-2.2.0.jar
MySQL	**<installation directory>**\mysql-connector-java-*version*-bin.jar;
Oracle	**ORACLE_HOME**\jdbc\lib\ojdbc6.jar;
Oracle (with XML DB)	**ORACLE_HOME**\jdbc\lib\ojdbc6.jar;**ORACLE_HOME**\LIB\xmlparserv2.jar;**ORACLE_HOME**\RDBMS\jlib\xdb.jar;
PostgreSQL	**<installation directory>**\postgresql.jar
Progress OpenEdge	%DLC%\java\openedge.jar;%DLC%\java\pool.jar; メモProgress OpenEdge SDK がマシンにインストールされていると想定します。%DLC% は OpenEdge がインストールされているディレクトリです。
Sybase	C:\sybase\jConnect-7_0\classes\jconn4.jar
Teradata	**<installation directory>**\tdgssconfig.jar;**<installation directory>**\terajdbc4.jar

- CLASSPATH 変数の変更は、使用中のコンピュータのJava アプリケーションの振る舞いに影響する可能性があります。続行する前に生じえる影響を把握するために、Java ドキュメントを参照してください。
- 環境変数は、ユーザーまたはシステムであることができます。システム環境変数を変更するには、オペレーティングシステムに対する管理者権限が必要です。
- 環境変数を変更後、設定の変更を有効にするには、プログラムを再起動してください。または、オペレーティングシステムをログオフまたは再起動してください。

Windows 7 でCLASSPATH を構成する:

1. [スタート] メニューを開始して、[コンピューター] を右クリックします。

2. [プロパティ] をクリックします。
3. [システムの詳細設定] をクリックします。
4. [詳細設定] タブから [環境変数] をクリックします。
5. CLASSPATH 変数を、ユーザーまたはシステム環境変数からロケートして、編集をクリックします。CLASSPATH 変数が存在しない場合は、[新規] をクリックして作成します。
6. オペレーティングシステム内で JDBC ドライバーがある場所にパスを含むように変数の値を編集します。JDBC ドライバーパスを他の既に CLASSPATH 変数内にあるパスから区別するためにセミコロン (;) で区切ります。

Windows 10 で CLASSPATH を構成する

1. Windows キーを押して、環境編集を入力します。
2. [システム環境変数の編集] の提案をクリックします。
3. [環境変数] をクリックします。
4. CLASSPATH 変数を、ユーザーまたはシステム環境変数からロケートして、編集をクリックします。CLASSPATH 変数が存在しない場合は、[新規] をクリックして作成します。
5. オペレーティングシステム内で JDBC ドライバーがある場所にパスを含むように変数の値を編集します。JDBC ドライバーパスを他の既に CLASSPATH 変数内にあるパスから区別するためにセミコロン (;) で区切ります。

19.1.7 PostgreSQL 接続のセットアップ

PostgreSQL データベースへの接続は、ネイティブな接続として、または、ODBC、JDBC、および他のドライバーを使用して確立することができます。ネイティブな接続の利点は、使用中のシステムにインストールする場合、ドライバーを必要としないことです。

ネイティブではないドライバーを使用しての接続を作成する場合は、以下のトピックを参照してください:

- JDBC 接続の設定 868
- PostgreSQL (ODBC) への接続 918

それ以外の場合、PostgreSQL へのネイティブの接続をセットアップする場合、以下の手順に従ってください。継続するには、以下の前提条件が必要です: ホスト名、ポート、データベース名、ユーザー名、およびパスワード。

ネイティブ PostgreSQL 接続のセットアップ:

1. データベース接続ウィザードを開始します 848.
2. 「PostgreSQL 接続」をクリックします。
3. 対応するボックスに、ホスト (PostgreSQL が同じマシンで作動している場合、*ローカルホスト*)、ポート (通常、5432 ですが、これは任意です)、データベース名、ユーザー名とパスワードを入力します。

4. 「**接続**」をクリックします。

PostgreSQL データベースサーバーが異なるマシンに存在する場合、次の点に注意してください：

- PostgreSQL データベースサーバーは、クライアントからの接続を受け入れるように構成されなければなりません。具体的には、**pg_hba.conf** ファイルがローカルではない接続を許可するように構成される必要があります。第2に、**postgresql.conf** ファイルは、特定されたIP アドレスとポートをリッスンするように構成される必要があります。詳細に関しては、PostgreSQL ドキュメントを参照してください（https://www.postgresql.org/docs/9.5/static/client-authentication-problems.html）。
- サーバーマシンがファイアウォールを越えてデザインされたポート（通常、5432）上で接続を受け入れるように構成されている必要があります。例えば、「**コントロール パネル**」>「**Windows ファイアウォール**」>「**詳細設定**」>「**受信の規則**」で設定すると、Windows 上で作動するデータベースサーバーでは、ファイアーウォールを越えてポート 5432 に接続を許可するルールを作成する必要がない場合があります。

19.1.8　SQLite 接続の設定

SQLite (http://www.sqlite.org) は、ポータブル性とインストールの簡便性が重要な場合理想的な構成のファイルベースの、内蔵のデータ型です。Since SQLite データベースは、XMLSpy によりネイティブにサポートされるため、接続のためにドライバーをインストールする必要はありません。

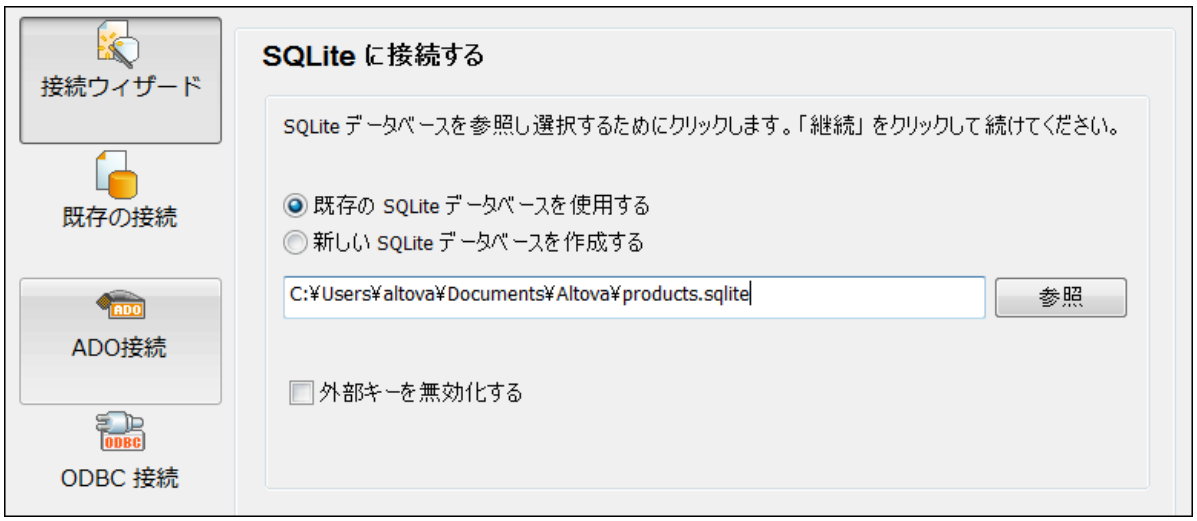
19.1.8.1　既存の SQLite データベースへの接続

接続ウィザードまたは **次を使用して開く...** コマンドを使用して直接ウィンドウエクスプローラーから既存の SQLite データベースに接続することができます。

接続ウィザードの使用

既存のSQLite データベースに接続する方法:

1. データベース接続ウィザードを実行します(データベース接続ウィザードの開始[848]を参照)。
2. **SQLite** を選択して、「次へ」をクリックします。



3. [既存のSQLite データベースを使用] を選択して、SQLite データベースファイルを参照、またはデータベースへの(相対または絶対)パスを入力します。「接続」ボタンは、データベースファイルへのパスを入力すると有効化されます。
4. 「外部キーの無効化」チェックボックスを任意で選択します。部キーの制約[875]も参照してください。
5. 「接続」をクリックします。

ウィンドウエクスプローラーから開く

SQLite データベースをウィンドウエクスプローラーから以下のように直接開くことができます:

1. ウィンドウエクスプローラー内の既存のデータベースファイルを右クリックし、「開く」をコンテキストメニューから選択します。
2. **DatabaseSpy** を提案されたリストから選択します。この選択肢が使用できない場合 **他のアプリを選択** を選択し、DatabaseSpy 実行可能ファイルをインストールディレクトリから参照します。
3. データベース接続ウィザードが表示されると[閉じる] をクリックして無視します。

後者が既に開かれている場合データベースファイルを DatabaseSpy にドラッグアンドドロップすることができます。

メモ

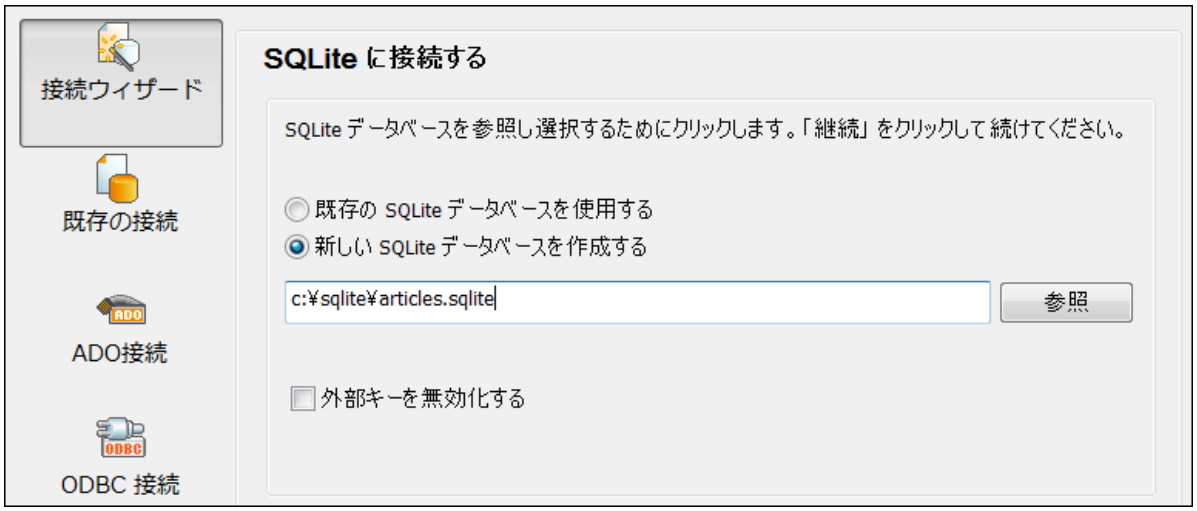
- アプリケーションが起動される都度データベース接続ウィザードが表示されるように[ツール | オプション] から DatabaseSpy をオプションで構成することができます。これを行うには設定の一般 グループのチェックボックス **データベース接続ダイアログの作成を表示** をクリアします。
- DatabaseSpy が開かれており同じデータベースファイルへのデータソースが既存の場合再利用されます。すなわち、同じデータベースへの複製データソースは作成されません。**開く** を使用して、またはドラッグアンドドロップを使用してデータベースファイルを開く際に適用することができます。

19.1.8.2 新規の SQLite データベースの作成

既存のデータベースファイルに接続する代替として、新規のSQLite データベースファイルを作成して、接続することができます。XMLSpy により作成されるデータベースは、空です。クエリまたはスクリプトを使用して、必要とされるデータベース構造を作成し、データを追加します。

新規のSQLite データベースの作成方法:

1. データベース接続ウィザードを実行します（ データベース接続ウィザードの開始[848]を参照）。
2. **SQLite** を選択して、「次へ」をクリックします。



3. [新規 SQLite データベースの作成] を選択して、作成されるデータベースファイルの（相対または絶対）パスを入力します（例えば、**c:\users\public\products.sqlite**）。または、[参照] をクリックして、フォルダーを選択して、ファイル名テキストボックスにデータベースファイルの名前（例えば、**products.sqlite**）を入力し、[保存] をクリックします。

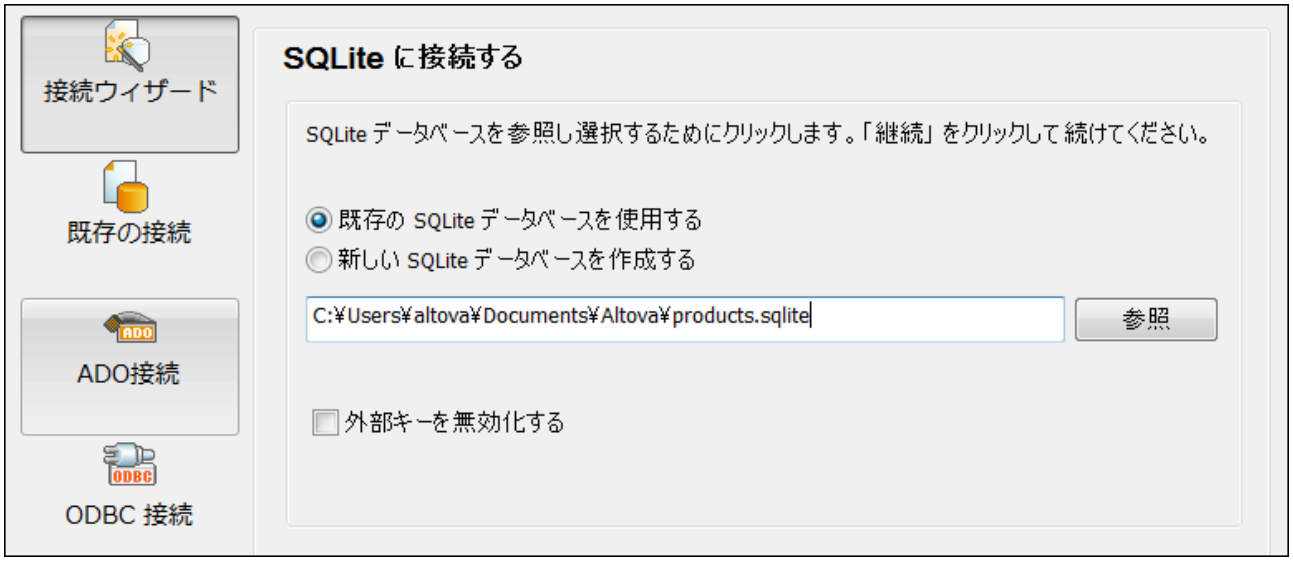
> データベースファイルを作成する場所に対して、書き込みアクセスの許可があることを確認してください。

4. 「**外部キーの無効化**」チェックボックスを任意で選択します。次も参照してください：外部キーの制約[875]。
5. 「**接続**」をクリックします。

19.1.8.3 外部キーの制約

XMLSpy, から既存のSQLite データベースに接続する場合、または新規を作成する場合、デフォルトで、外部キーの制約 が有効化されます。外部キーの制約はデータベース内のデータの整合性を保持する手助けを行います。例えば、外部キーが有効化されていると、他のテーブル内に依存関係がある場合、テーブルからレコードを削除することができません。

特定の場合、一時的にこの振る舞いをオーバーライドし、外部キーを無効化する必要がある場合があります。例えば、データ検証エラー無しにデータの列を更新、または、挿入する場合など。SQLite データベースに接続する前に、明示的に外部キーを無効化するには、データベース接続ウィザードの「**外部キーの無効化**」オプションを選択します。



「SQLite への接続」ウィザードページ

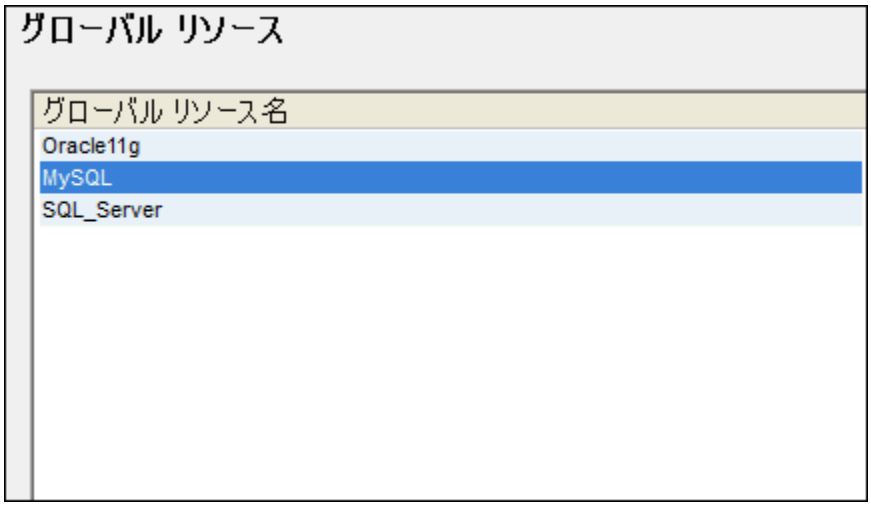
外部キーが無効化されていると、検証チェックのためチェックすることのできなかったデータに対して操作をおこなうことができます。同時に、データベース内に不正確なデータを入れてしまう危険性、または、「無所属の」行を作成する可能性あります。(「無所属の」行の例は、「address」テーブルにリンクされていない「person」テーブル内のアドレスなどが挙げられます。これは、個人名が削除されているにもかかわらず、この個人に関連した住所が削除されていない場合などにこのような行が発生します。)

19.1.9 グローバルリソースからの接続の使用

Altova グローバルリソースはファイル、フォルダー、およびデータベースへのポータブルなレファレンスです。以前にデータベースをグローバルリソースとして既に構成している場合、接続をいつでも再使用することができます(異なる Altova アプリケーション間でも使用可能)。

グローバルリソースからのデータベース接続の使用:

1. データベース接続ウィザードを開始します。 848
2. [グローバルリソース] をクリックします。グローバルリソースとして使用可能なデータベース接続がリストされます。



3. データベース接続履歴から「接続」をクリックします。

ヒント: 各グローバルリソースの追加情報を取得するために、マウスカーソルをグローバルリソースにポイントします。

19.1.10 データベース接続のサンプル

ADO、ODBC、またはJDBC を使用したXMLSpy からのデータベースへの接続のためのサンプルがこのセクションには含まれています。ADO.NET 接続サンプルは個別にリストされています サンプルADO.NET 接続文字列[863] を参照してください。For instructions about establishing a native connection to PostgreSQL とSQLite へのネイティブな接続の作成のための手順は、それぞれ PostgreSQL 接続のセットアップ[872] とSQLite 接続のセットアップ[873] を参照してください。

以下の点に注意してください:

- Windows 構成、ネットワーク環境、およびデータベースクライアント、または、サーバーソフトウェアが各サンプルと異なる場合手順は異なります。
- ほぼすべてのデータベースの型では、1つ以上のデータアクセステクノロジー(ADO、ADO.NET、ODBC、JDBC) またはドライバーを使用して接続することができます。データベース接続パフォーマンス、およびドライバーの機能と制限は、XMLSpy 外で構成された選択されたドライバー、(該当する場合)データベースクライアントソフトウェア、および追加接続パラメータなどによって異なります。

19.1.10.1 Firebird (JDBC) への接続

このサンプルではFirebird データベースサーバーへの接続のサンプルの命令について説明されています。

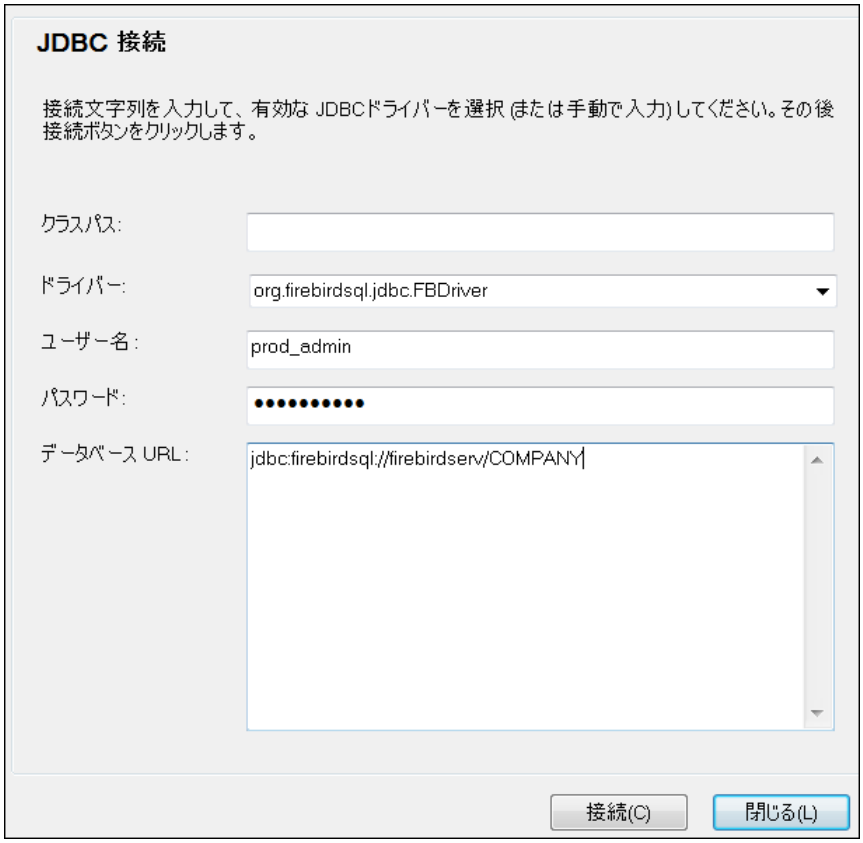
必要条件:

- JRE (Java Runtime Environment) またはJava Development Kit (JDK) がインストールされている必要があります。これはOracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションからJava Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタムJVM パス b) Windows レジストリ内で見つかったJVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームがJRE/JDK のプラットフォームに一致することを確認してください。

- Firebird JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります（データベースへの接続を可能にする .jar ファイルのフォームをとります）。ドライバーは Firebird Web サイトからダウンロードすることができます（ http://www.firebirdsql.org/ ）。このサンプルは、*Jaybird 2.2.8* を使用します。
- データベース接続の詳細があります: ホスト、データベース パス、またはエイリアス、ユーザー名、および パスワード。

JDBC を使用して Firebird に接続する

1. データベース接続ウィザードを開始します[848].
2. 「JDBC 接続」をクリックします。
3. 「クラス パス」の横にデータベースへの接続を与える .jar ファイルへの パスを入力します。必要であれば、セミコロンで区切られた .jar ファイル パスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次の パスを使用して見つけることができます: **C:\jdbc\firebird\jaybird-full-2.2.8.jar**。.jar ファイル パスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラス パス」テキスト ボックスを空白のままにすることができます。（次も参照: CLASSPATH の構成[870]）。
4. 「ドライバー」ボックスから、**org.firebirdsql.jdbc.FBDriver** を選択します。このエントリは、有効な .jar ファイル パスが、「クラス パス」テキスト ボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます（前のステップを参照してください）。



5. 対応するテキスト ボックスに、データベースへのユーザー名と パスワードを入力します。
6. URL テキスト ボックスに、データベース サーバーに適用することのできる ハイライトされた値を置き換える、データベース サーバーの接続文字列を入力します。

```
jdbc:firebirdsql://<host>[:<port>]/<database path or alias>
```

7. 「接続」をクリックします。

19.1.10.2 Firebird (ODBC) への接続

このサンプルでは、Linux サーバー上で実行される、Firebird 2.5.4 データベースへの接続について説明されています。

必要条件:

- Firebird データベースサーバーは、クライアントからのTCP/IP 接続を受け入れるように構成されています。
- Firebird ODBC ドライバーを使用中のオペレーティングシステムにインストールする必要があります。このサンプルは、を使用します。 Firebird ODBC ドライバー バージョン 2.0.3.154 は、Firebird ウェブサイト (http://www.firebirdsql.org/) からダウンロードすることができます。
- Firebird クライアントを、使用中のオペレーティングシステムにインストールする必要があります。 Firebird 2.5.4 クライアントのためのスタンドアロンのインストーラーが存在しない点に注意してください。また、クライアント は、サーバーのインストール パックの一部です。Firebird サーバー インストール パックを Firebird ウェブサイト(http://www.firebirdsql.org/) からダウンロードすることができます。ダウンロードの際、[スーパークラシック/クラシック または スーパーサーバーのための、W indows 実行ファイル] インストーラーを選択してください。クライアント ファイルのみをインストールする場合、[最小クライアント インストール- サーバーツール無し] を選択します。

  重要点:

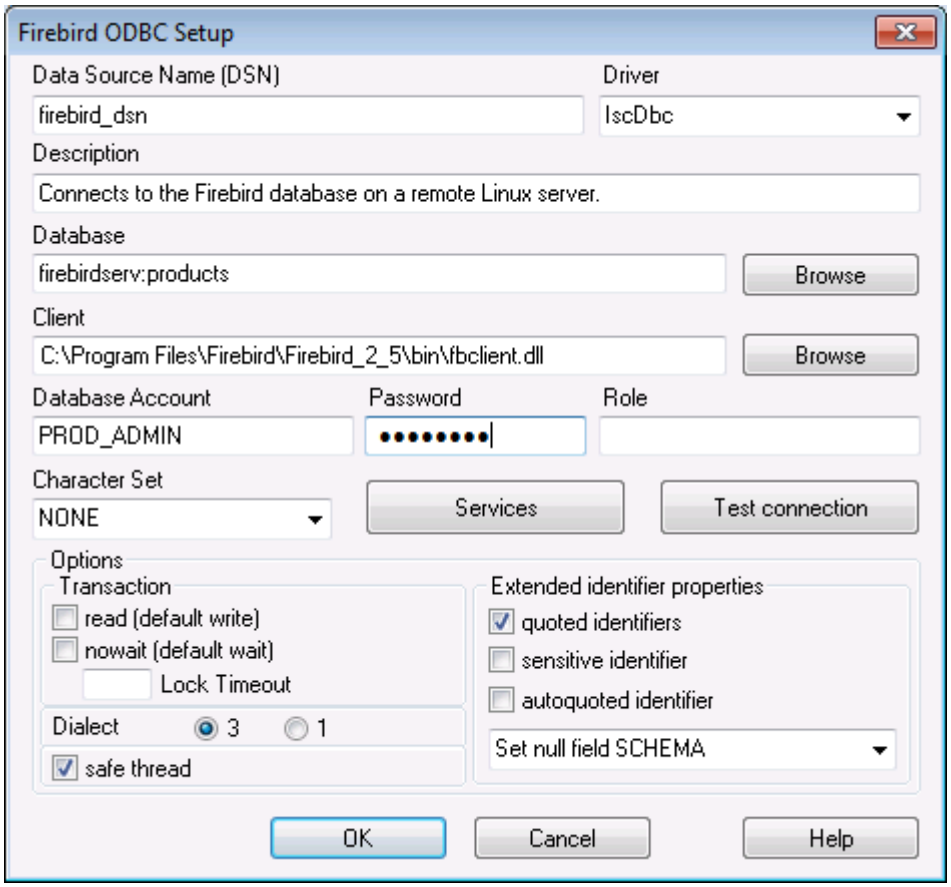
  - Firebird ODBC ドライバーおよびクライアント (32 ビット または 64 ビット) のプラットフォームは、XMLSpy のプラットフォームに適応する必要があります。
  - Firebird クライアントのバージョンは、接続されている Firebird サーバーのバージョンに適応する必要があります。

- データベース接続の詳細 サーバー上のサーバー ホスト名 または IP アドレス、データベース パス (または エイリアス)、ユーザー名、および パスワード。

ODBC を使用して Firebird に接続する:

1. データベース接続ウィザードを開始します[848]。
2. [ODBC 接続] をクリックする。
3. [ユーザー DSN] を選択して、(または、管理者の特権がある場合、[システム DSN] を選択して、)[追加] をクリックします。

4. Firebird ドライバーを選択して、をクリックします。そして[ユーザーDSN](前のステップでの選択により、または、[システムDSN])。Firebird ドライバーがリスト内に存在しない場合、使用中のオペレーティングシステムにインストールされていることを確認してください(使用可能な ODBC ドライバーの表示 867 を参照)。



5. データベース接続の詳細を入力します:

データソース名 (DSN)	作成する詳細名とデータソースを入力します。

データベース	サーバーホスト名 または IP アドレスをコロンとデータベースエイリアス(またはパス)の後に入力します。このサンプルでは、ホスト名は、`firebirdserv`、データベースエイリアスは、`products` です:
	
	`firebirdserv:products`
	
	データベースエイリアスを使用することにより、サーバー側上では、データベース管理者は、エイリアス製品を構成し、サーバー上の実際の Firebird (.fdb) データベースファイルをポイントします(詳細は、Firebird のドキュメントを参照してください)。
	
	ホスト名の代わりに サーバー IP アドレス、エイリアスの代わりに パス を使用することもできます。ですから、以下のサンプル接続の文字列は有効です:
	
	`firebirdserver:/var/Firebird/databases/butterflies.fdb`
	`127.0.0.1:D:\Misc\Lenders.fdb`
	データベースが W indows マシン上に存在する場合、[参照] をクリックして、Firebird (.fdb) データベースファイルを直接選択します。

クライアント	**fbclient.dll** ファイルへのパスを入力します。デフォルトでは、これはFirebird インストールディレクトリのサブディレクトリです。
データベース アカウント	データベース管理者（このサンプルの場合 PROD_ADMIN）により与えられたデータベースユーザー名を入力します。
パスワード	データベース管理者により与えられた データベース パスワード を入力します。

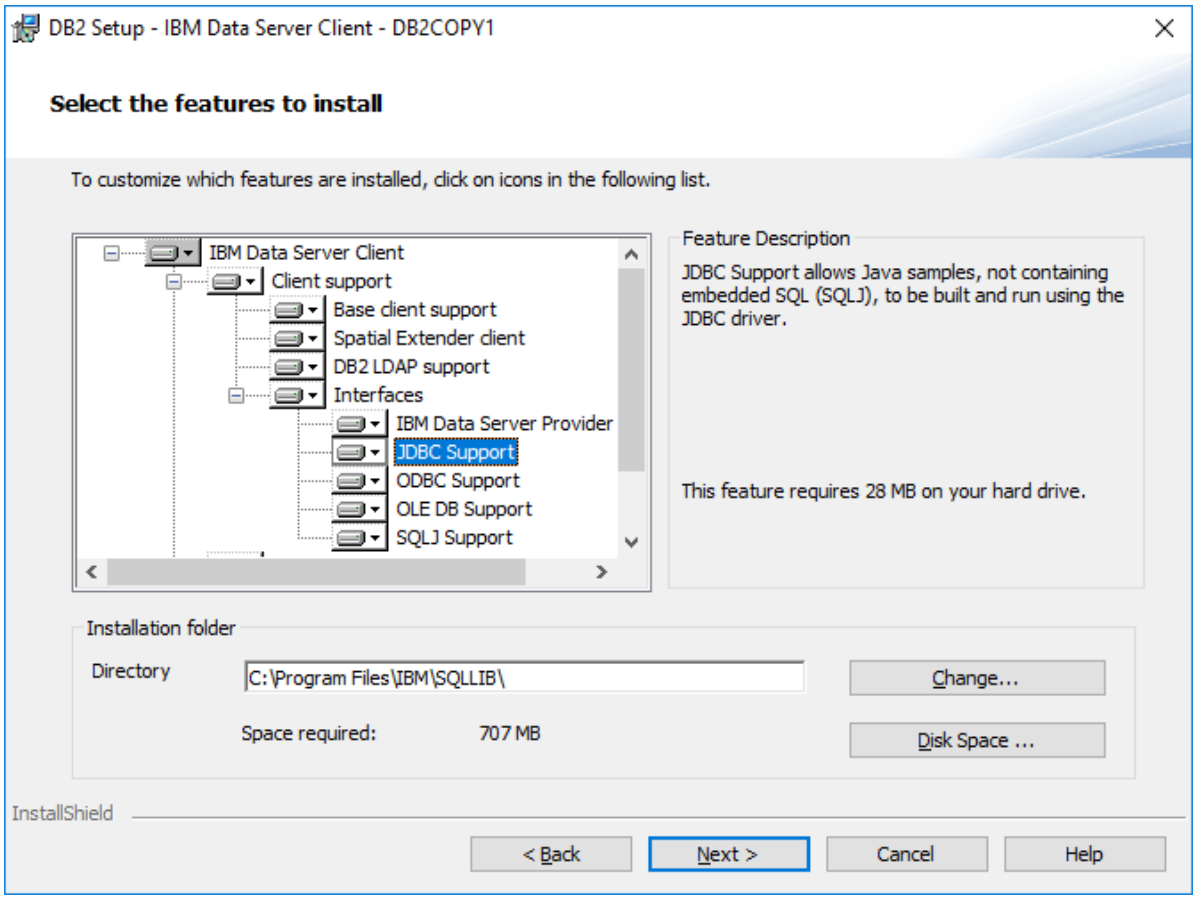
6. 「OK」をクリックします。

19.1.10.3 IBM DB2 (JDBC) への接続

このサンプルでは、JDBC を使用して **IBM DB2** データベースサーバーに接続する手順のサンプルについて説明されています。

必要条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) JAVA_HOME 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。このサンプルは Oracle の OpenJDK 11.0 64-bit を使用し、その結果 XMLSpy の 64 ビット バージョンを使用します。
- (データベースへの接続を与える 1 つ、または、複数の .jar ファイル) JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります。**IBM Data Server Client** バージョン 10.1 (64-ビット) をインストール後使用することのできる JDBC ドライバーをこのサンプルは使用しています。JDBC ドライバーをインストールするためには、通常 インストールを選択してください。または、インストールウィザードでこのオプションを明示的に選択してください。

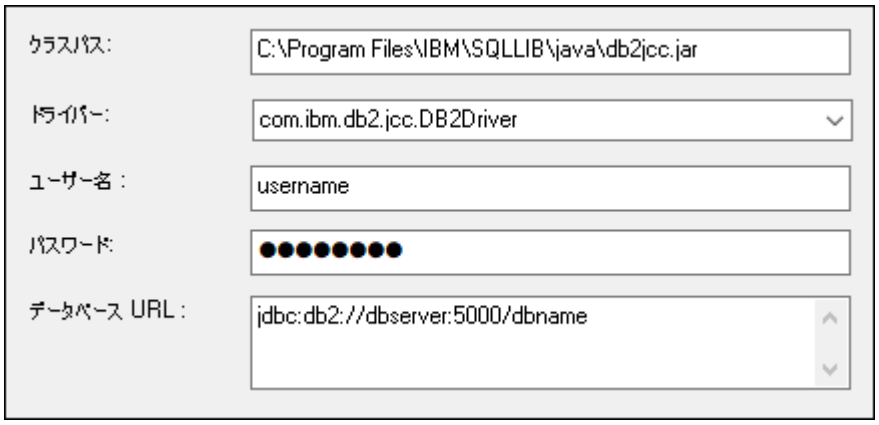


デフォルトのインストールパスを変更していない場合、インストール後必要とされる .jar ファイルは**C:\Program Files\IBM\SQLLIB\java** ディレクトリ内で見つけることができます。

- 以下のデータベース詳細が必要になります: ホスト、データベース、ポート、ユーザー名、および、パスワード。

JDBC を使用して IBM DB2 に接続する:

1. [データベース接続ウィザードを開始します](#) 848.
2. 「JDBC 接続」をクリックします。
3. 「クラスパス」の横に、データベースへの接続を与える .jar ファイルへのパスを入力します。このサンプルは**C:\Program Files\IBM\SQLLIB\java\db2jcc.jar** を参照します。データベースサーバーバージョンにより**db2jcc4.jar** ドライバーを参照する必要がある場合があります。ドライバーの互換性に関してはIBM ドキュメントを参照してください（http://www-01.ibm.com/support/docview.wss?uid=swg21363866）。.jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加している場合クラスパステキストボックスを空にしておくことができます（CLASSPATH の構成 870 を参照してください）。
4. 「ドライバー」ボックスから、**com.ibm.db2.jcc.DB2Driver** を選択します。このエントリは、有効な .jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます（前のステップを参照してください）。



5. データベースユーザーのユーザー名とパスワードを対応するテキストボックスに入力します。
6. JDBC 接続文字列をデータベース URL テキストボックスに入力します。使用中のデータベースサーバーに適用することのできるものと接続の詳細が置き換えられていることを確認してください。

```
jdbc:db2://hostName:port/databaseName
```

7. 「接続」をクリックします。

19.1.10.4 IBM DB2 (ODBC) への接続

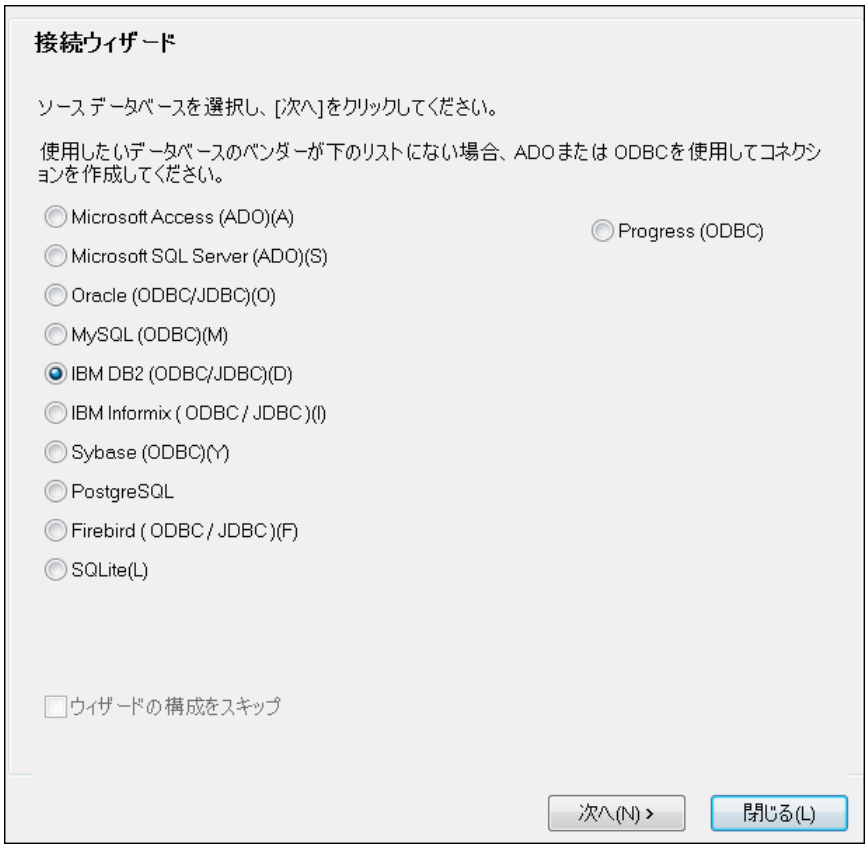
このトピックではODBC を使用した、IBM DB2 への接続のサンプル手順について説明されています。

前提条件:

- IBM Data Server Client は、使用中のオペレーティングシステムにインストールおよび構成される必要があります(このサンプルの場合はIBM Data Server Client 9.7 を使用します)。インストールの手順に関しては、IBM DB2 ソフトウェアにより提供されたドキュメントを確認してください。 IBM Data Server Client インストール後、使用中のコンピューターで ODBC ドライバーが使用可能か確認してください(使用可能な ODBC ドライバーの表示[867]を参照)。
- データベースエイリアスを作成します。複数の作成方法があります:
  - IBM DB2 構成アシスタントから
  - IBM DB2 コマンドラインプロセッサーから
  - ODBC データソースウィザードから(この場合は、下に表示される手順を参照してください)
- データベース接続の詳細: ホスト、データベース、ユーザー名およびパスワード

IBM DB2 への接続:

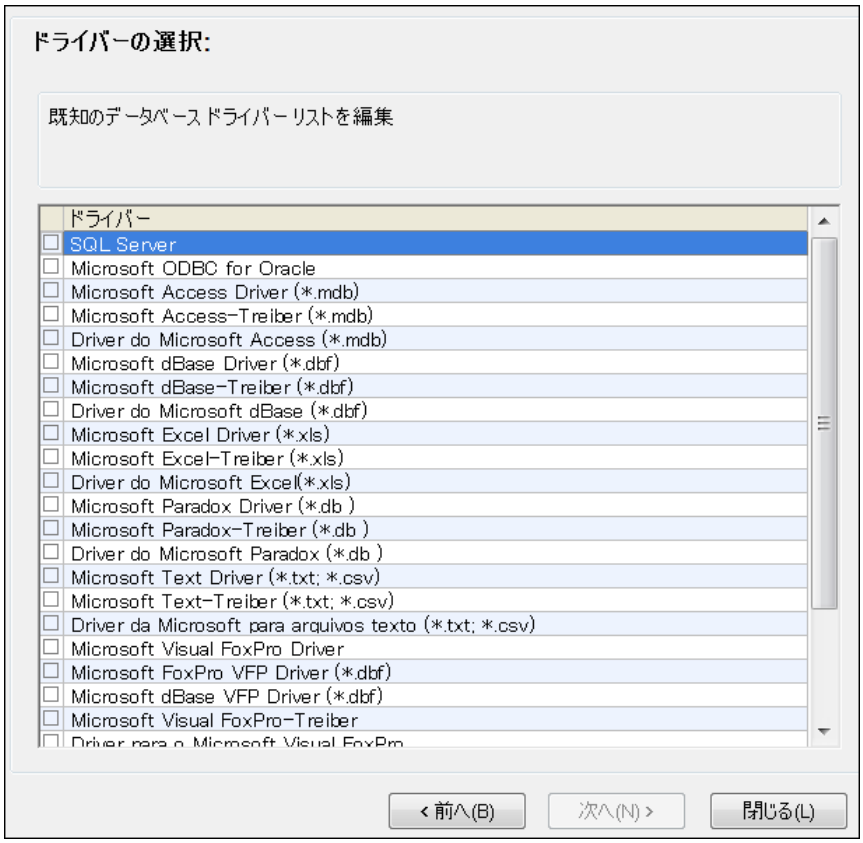
1. データベース接続ウィザードを開始します[848]。**IBM DB2 (ODBC/JDBC)** を選択します。



2. 「次へ」をクリックします。

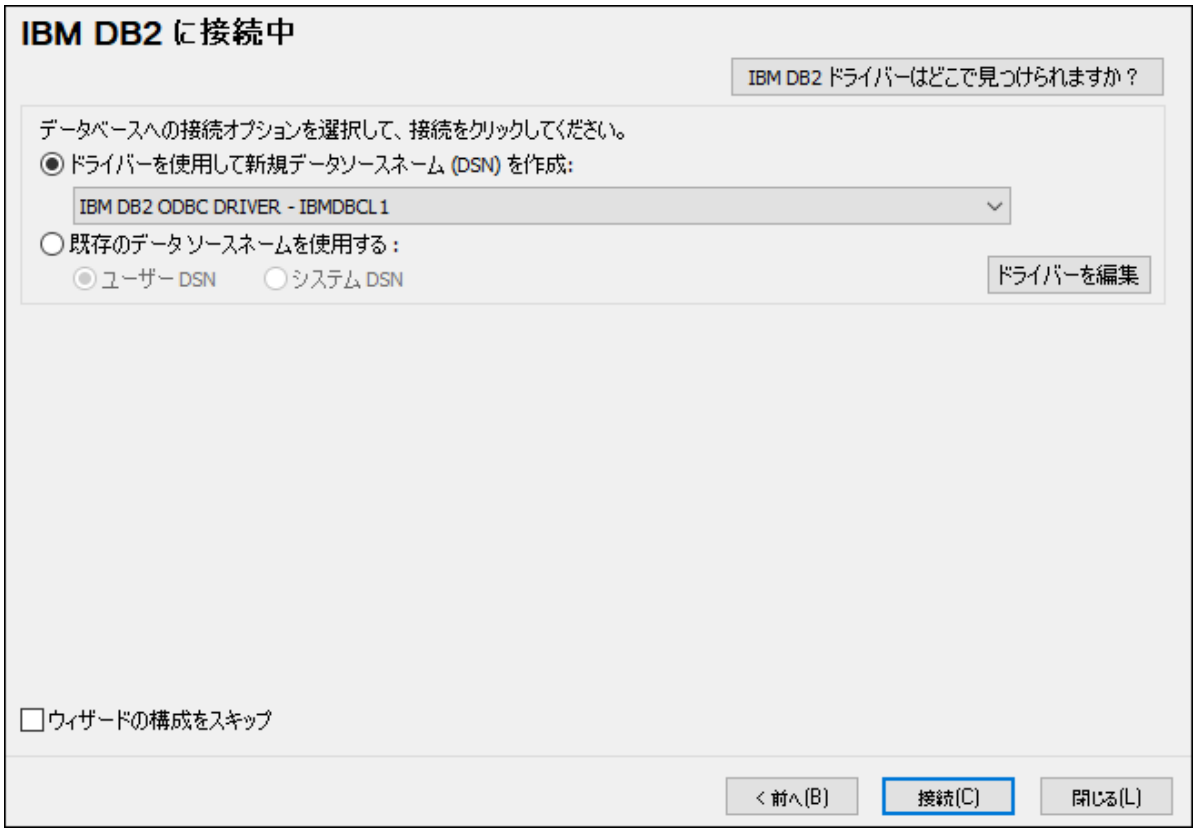
(C) 2015-2021 Altova GmbH

3. **ODBC** を選択して「次へ」をクリックします。データベースのための既知のリストを編集するようにプロンプトされると、IBM DB2 に適用することのできるドライバーを選択して(前提条件[883]を参照)、「次へ」をクリックします。

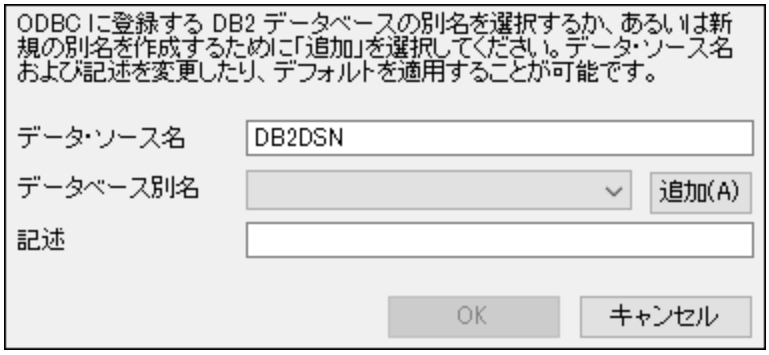


4. リストからIBM DB2 を選択して、**「接続」**をクリックします。（ドライバーで使用可能なリストを編集するには、**[ドライバーの編集]** をクリックして、使用可能なドライバーのリストを編集します。クリックして追加または削除するIBM DB2 ドライバーをチェックまたはチェックの解除をします。ドライバーを選択または選択の解除をします。）

(C) 2015-2021 Altova GmbH

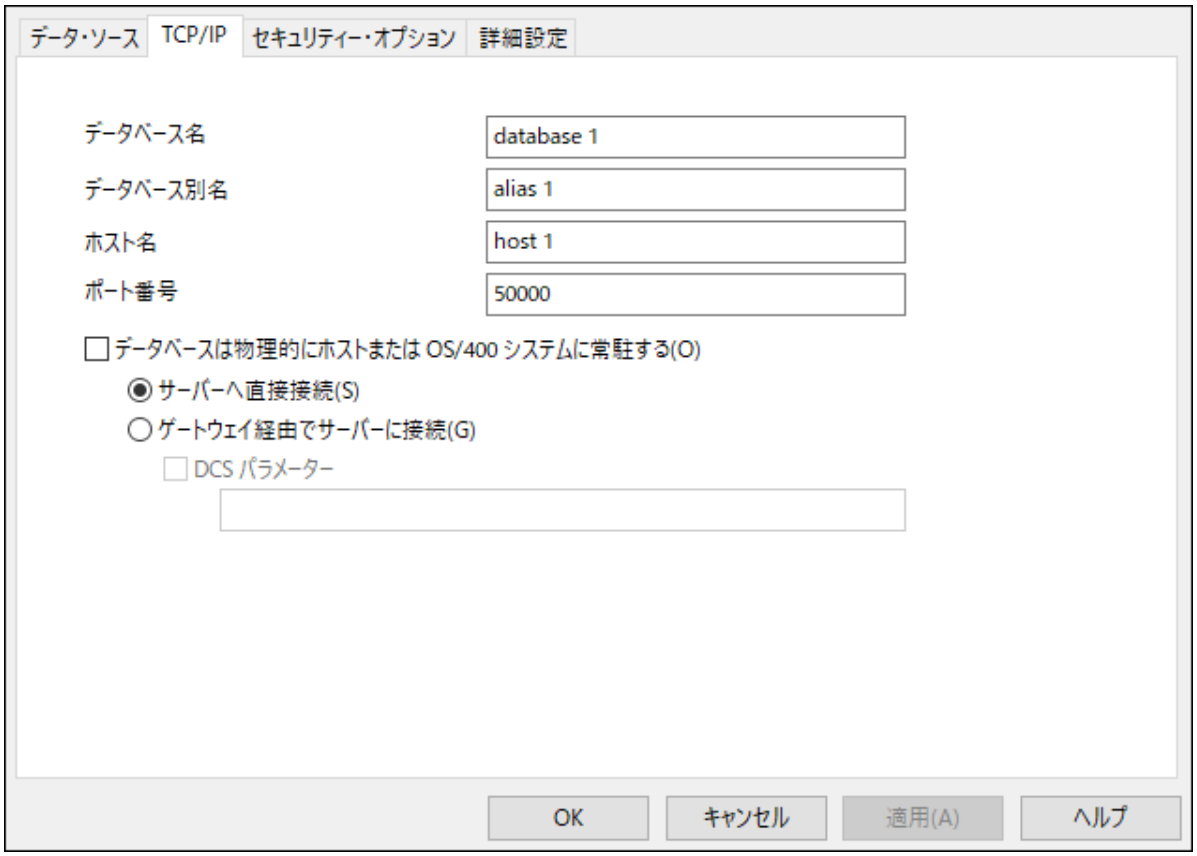


5. ソース名を入力します（このサンプルでは**DB2DSN**）「追加」をクリックします。

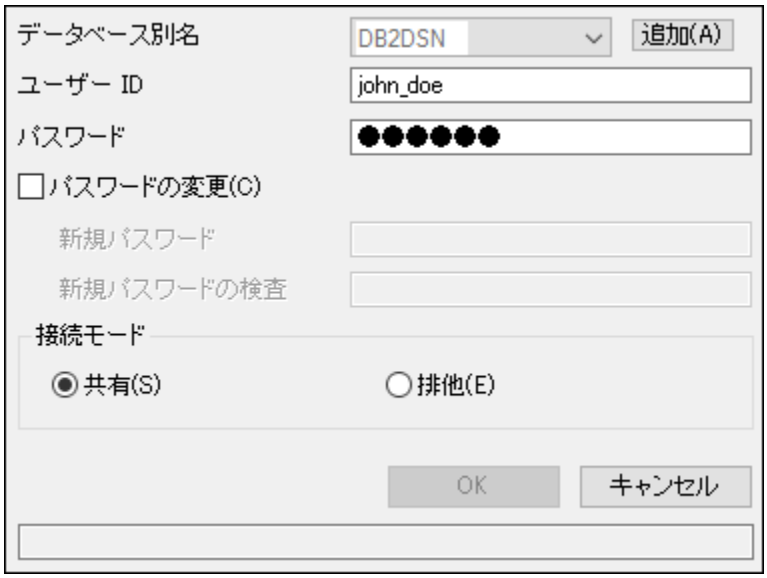


6. [**データソース**] タブで、データベースのユーザー名とパスワードを入力します。

7. **[TCP/IP]** タブで、データベース名、エイリアスの名前、ホスト名とポート名を入力して「OK」をクリックします。



8. ユーザ名とパスワードを再入力して「OK」をクリックします。



19.1.10.5 IBM DB2 for i (JDBC) への接続

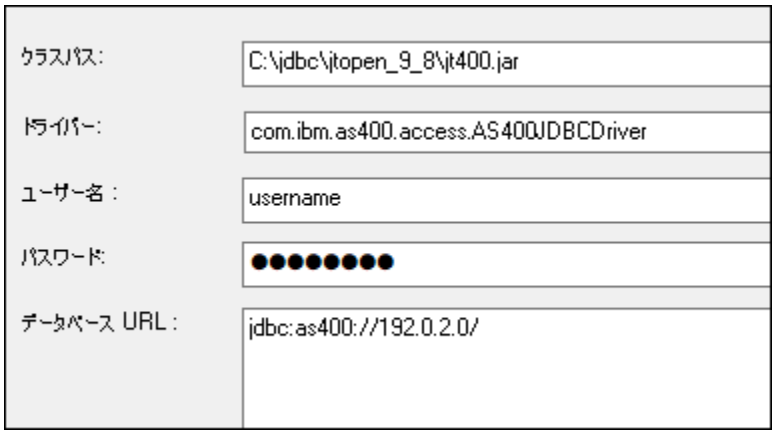
このサンプルではJDBC を介して**IBM DB2 for i** データベースサーバーに接続する方法について説明されています。

必要条件

- JRE (Java Runtime Environment) またはJava Development Kit (JDK) がインストールされている必要があります。これはOracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションからJava Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタムJVM パス b) W indows レジストリ内で見つかったJVM パス。c) JAVA_HOME 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームがJRE/JDK のプラットフォームに一致することを確認してください。このサンプルはOracle のOpenJDK 11.0 64-bit を使用し、その結果 XMLSpy の64 ビットバージョンを使用します。
- JDBC ドライバー (データベースへの接続を提供する1 つ、または、複数の.jar ファイル) が使用中のオペレーティングシステム上で使用可能である必要があります。このサンプルは以下のオープンソースを使用します: **Toolbox for Java/JTOpen** バージョン9.8 (http://jt400.sourceforge.net/)。パッケージをダウンロードし、ローカルディレクトリにアンパックすると、必要な.jar ファイルが**lib** サブディレクトリで使用できるようになります。
- 以下のデータベース接続の詳細が必要です: ホスト、ユーザー名、および パスワード

JDBC を使用してIBM DB2 for i に接続する方法:

1. データベース接続ウィザードを開始します[848]。
2. 「JDBC 接続」をクリックします。
3. 「クラスパス」の横にデータベースへの接続を与える.jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた.jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる.jar ファイルは次のパスを使用して見つけることができます: **C:\Informix_JDBC_Driver\lib\ifxjdbc.jar**。.jar ファイルパスをオペレーティングシステムのCLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成[870])。
4. 「ドライバー」ボックスから、**com.ibm.as400.access.AS400JDBCDriver** を選択します。このエントリは、有効な.jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムのCLASSPATH 環境変数内で検出された場合のみ使用することができます (前のステップを参照してください)。



5. データベースユーザーのユーザー名とパスワードを対応するテキストボックスに入力します。
6. JDBC 接続文字列をデータベースURL テキストボックスに入力します。`host` をデータベースサーバーのホスト名、または、IP アドレスと置き換えてください。

```
jdbc:as400://host
```

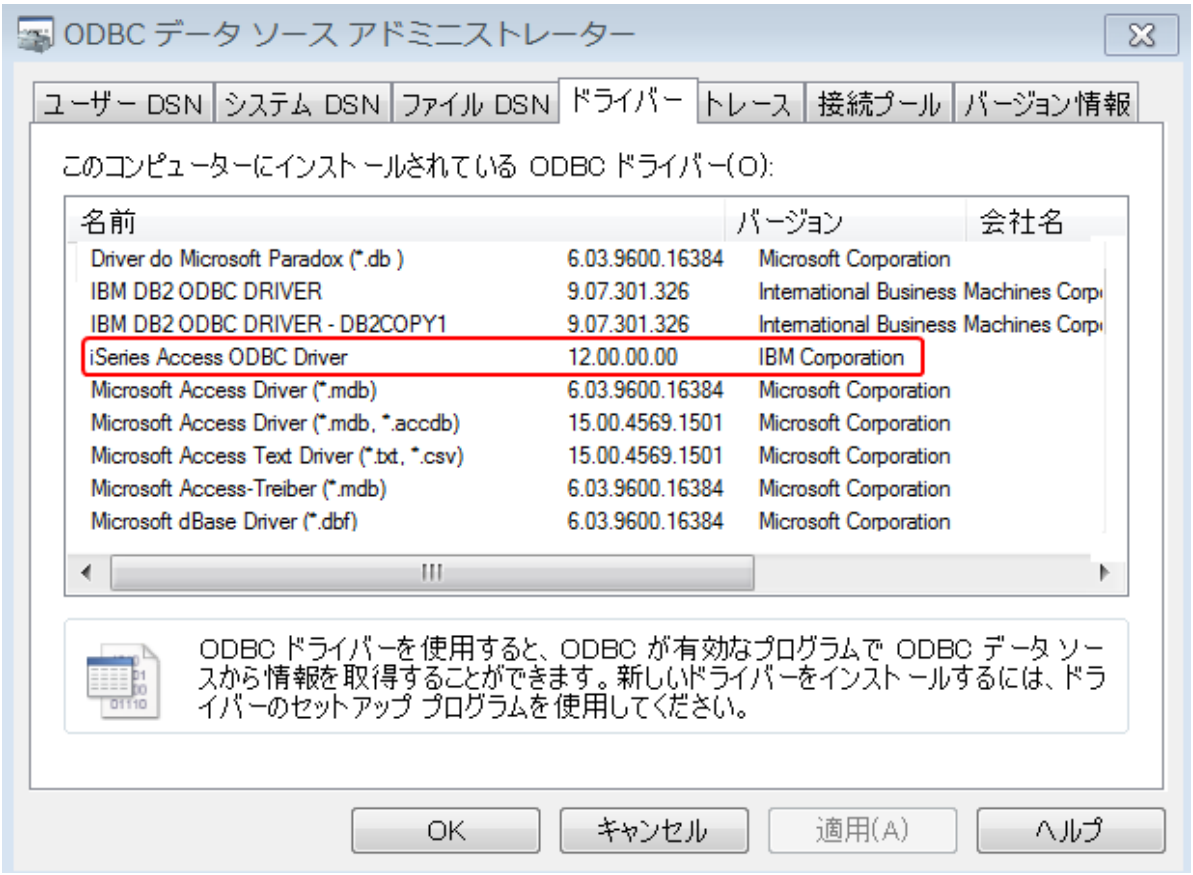
7. 「接続」をクリックします。

19.1.10.6 IBM DB2 for i (ODBC) への接続

このサンプルでは、ODBC を使用して *IBM DB2 for i* データベースへの接続について説明されています。

前提条件:

- *IBM System i Access for Windows* は使用中のオペレーティングシステム（このサンプルでは *IBM System i Access for Windows V6R1M0* を使用しています）にインストールされる必要があります。インストールの手順は、*IBM DB2 for i* ソフトウェアにより提供されているドキュメントを参照してください。インストール後、ODBC ドライバーが使用中のコンピューターで使用可能か確認してください（使用可能なODBC ドライバーの表示 867 を参照）。

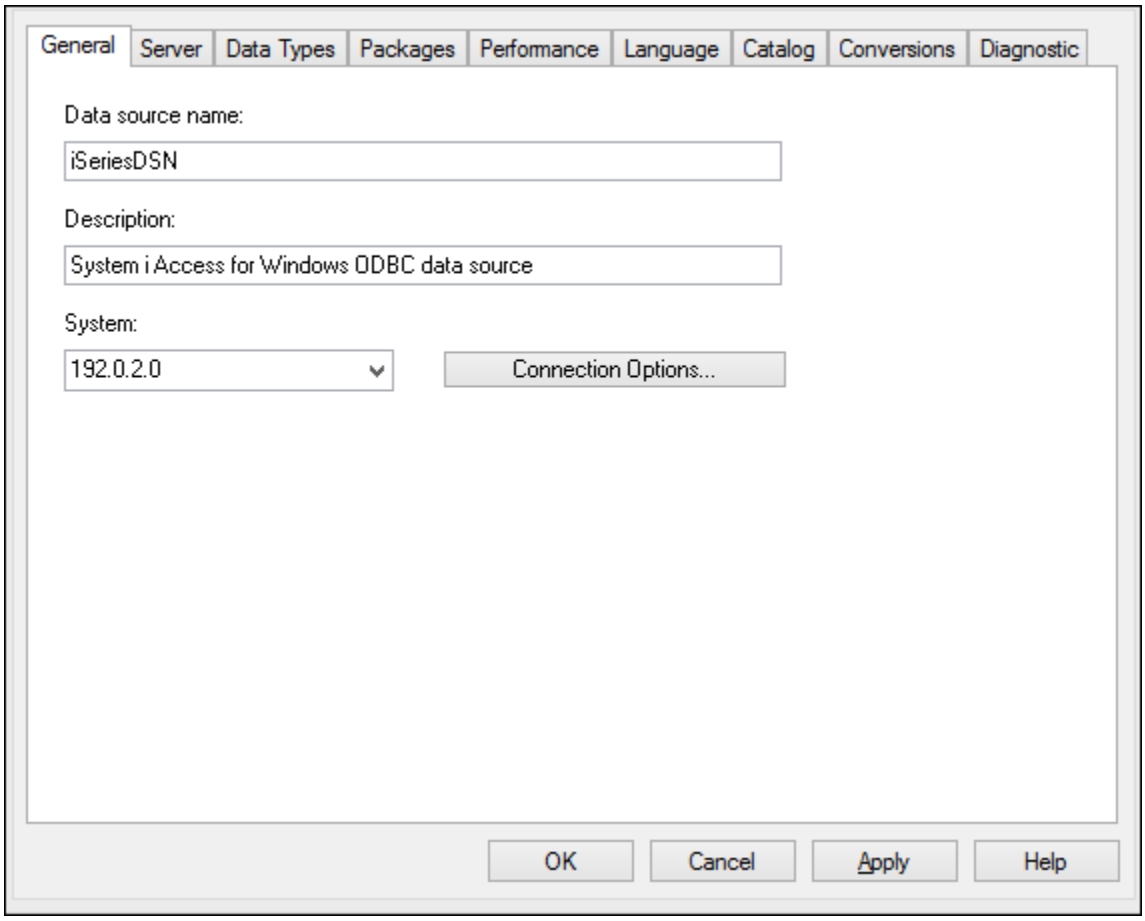


- データベース接続の詳細 データベースサーバーの I.P. アドレス、データベースユーザー名、および パスワード
- *i Navigator* を実行して、新規接続を作成するウィザードの手順を踏んで新規接続を作成してください。システムの指定をプロンプトされると、データベースサーバーのI.P. アドレスを入力してください。接続を作成した後、検証する（接続をクリックして、ファイル > 診断 > 接続の検証 を選択） ことが奨励されます。接続エラーが発生した場合、データベースサーバー管理者に連絡してください。

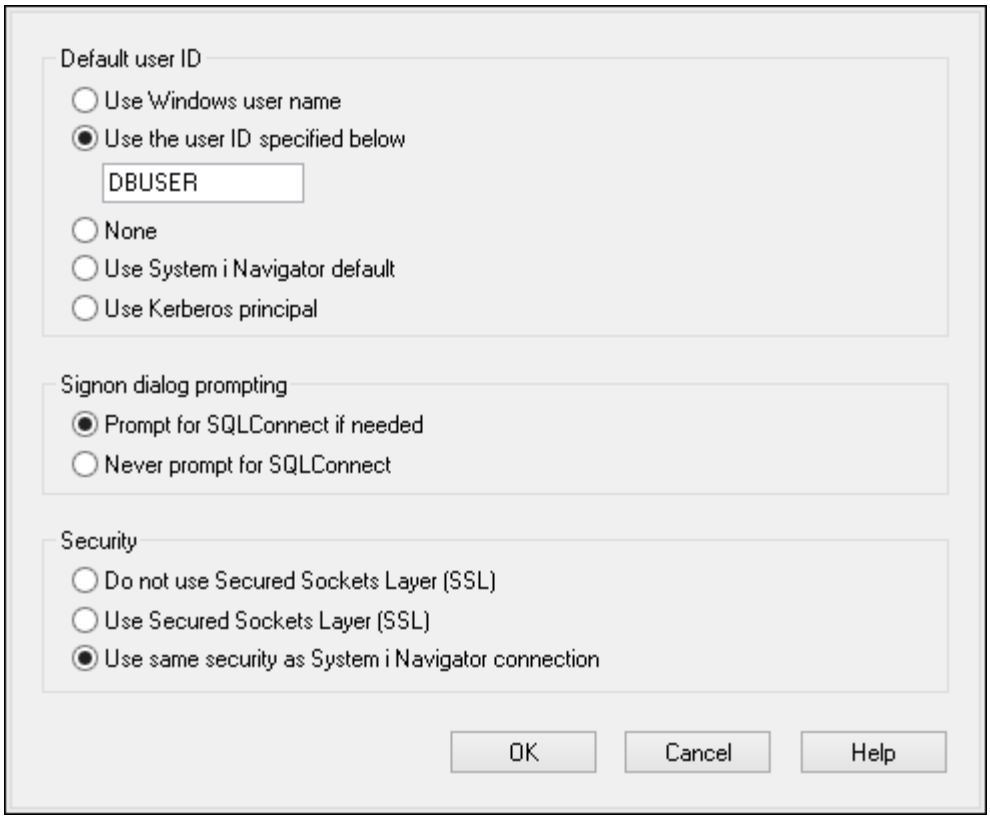
IBM DB2 for i への接続:

1. データベース接続ウィザードを開始します[848].
2. [ODBC 接続] をクリックします。
3. [ユーザー DSN] をクリックします。(または、[システム DSN] または、後に続く手順が同じ [ファイル DSN] をクリックします)。
4. 「追加」 をクリックします。
5. **[iSeries Access ODBC Driver]** をリストから選択します。[ユーザー DSN] (または、該当する場合 [システム DSN]) をクリックします。

6. データソース名を入力して、システムコンボボックスから接続を選択します。このサンプルでは、データソース名は **iSeriesDSN** でシステムは **192.0.2.0** です。



7. [接続オプション] をクリックします。[指定されたユーザー ID を使用] を選択して、データベースユーザー（このサンプル内では **DBUSER**）の名前を入力します。



8. 「OK」をクリックします。DSN 内のリストで新しいデータソースを使用することができます。
9. 「接続」をクリックします。
10. データベースのユーザー名とパスワードをプロンプトされると入力して「OK」をクリックします。

19.1.10.7 IBM Informix (JDBC) への接続

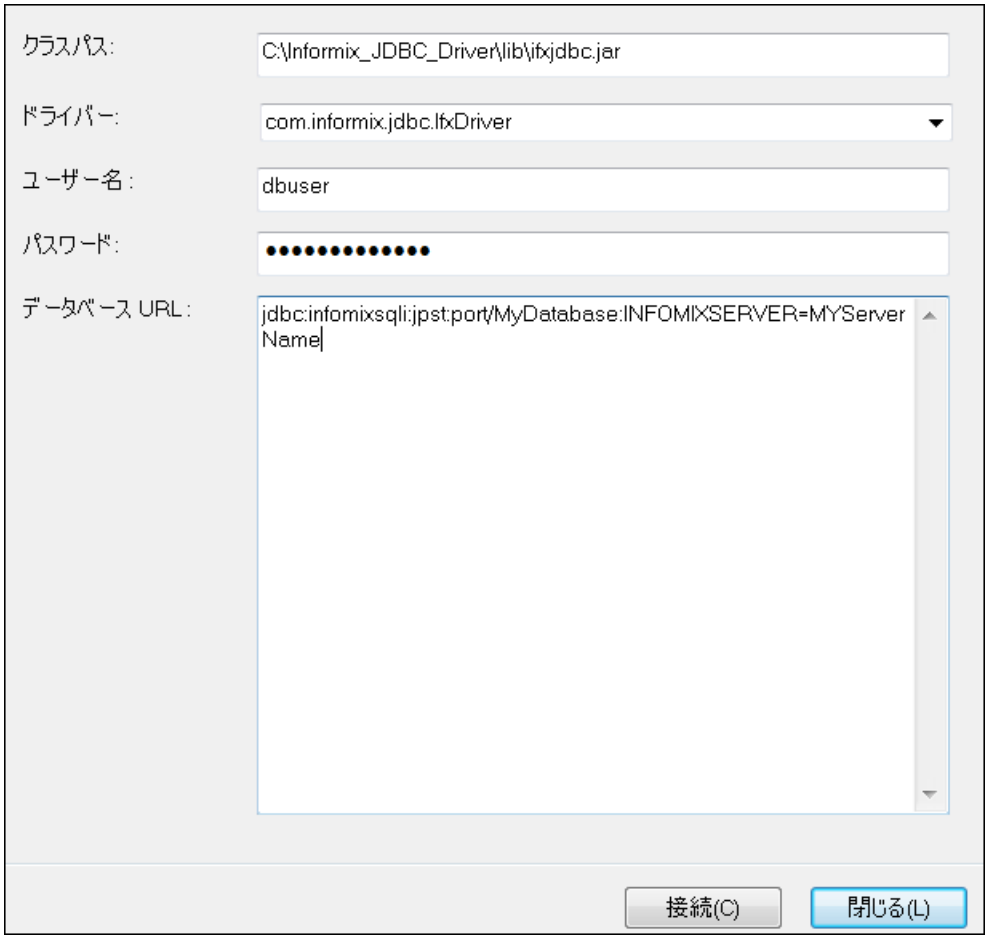
このサンプルでは、JDBC を使用して IBM Informix データベースサーバーに接続する手順のサンプルについて説明されています。

前提条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- (データベースへの接続を与える1つ、または、複数の .jar ファイル) JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります。この例では、IBM Informix JDBC ドライバー バージョン 3.70 が使用されています。ドライバーのインストールの説明に関しては、ドライバーに付随するドキュメント、または、「IBM Informix JDBC ドライバ プログラマーズガイド」を参照してください)。
- データベース接続の詳細: ホスト、サーバーの名前、データベース、ポート ユーザー名、および パスワード

JDBC を使用して IBM Informix に接続する

1. データベース接続ウィザードを開始します[848].
2. 「JDBC 接続」をクリックします。
3. 「クラスパス」の横にデータベースへの接続を与える .jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた .jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次のパスを使用して見つけることができます: **C:\Informix_JDBC_Driver\lib\ifxjdbc.jar**。.jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成[870])。
4. 「ドライバー」ボックスから、**com.informix.jdbc.IfxDriver** を選択します。このエントリは、有効な .jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます (前のステップを参照してください)。



5. データベースのユーザー名とパスワードを対応するテキストボックスに入力します。
6. ハイライトされた値を、使用するデータベースサーバーに適用する値と置き換えて、データベース URL テキストボックス内にデータベースサーバーへの接続文字列を入力します。

```
jdbc:informix-sqli://hostName:port/databaseName:INFORMIXSERVER=myserver;
```

7. 「接続」をクリックします。

19.1.10.8 MariaDB (ODBC) への接続

このサンプルでは、ODBC を使用したMariaDB データベースへの接続方法について説明されています。

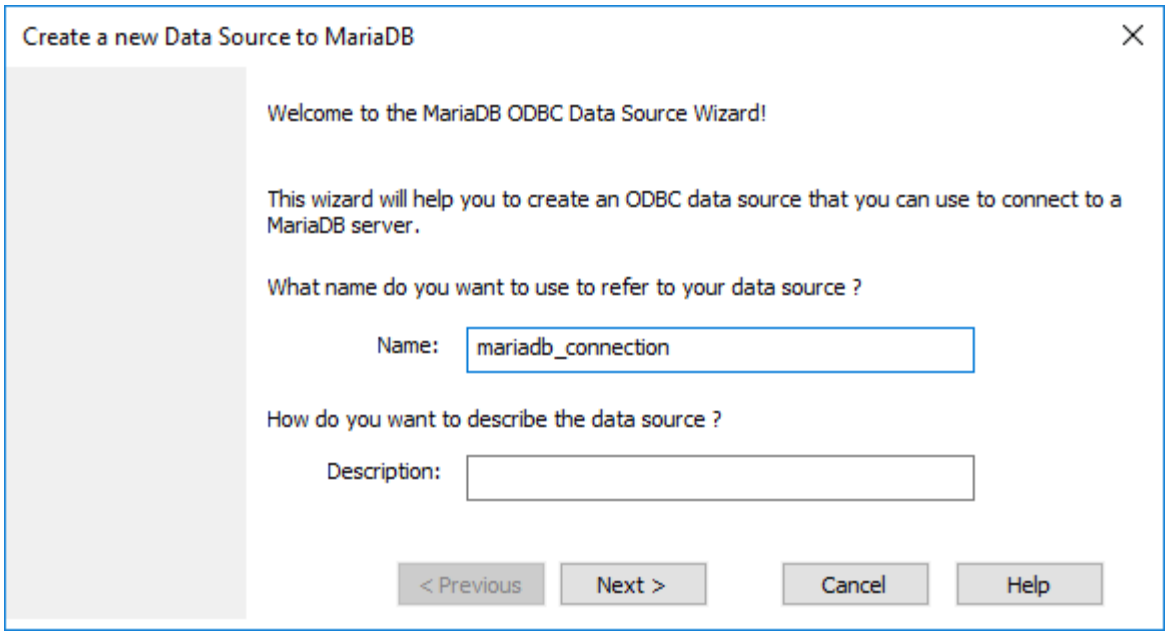
前提条件:

- MariaDB コネクタ/ODBC (https://downloads.mariadb.org/connector-odbc/) がインストールされている必要があります。
- 次のデータベース接続詳細が存在します: ホスト、データベース、ポート、ユーザー名、および、パスワード。

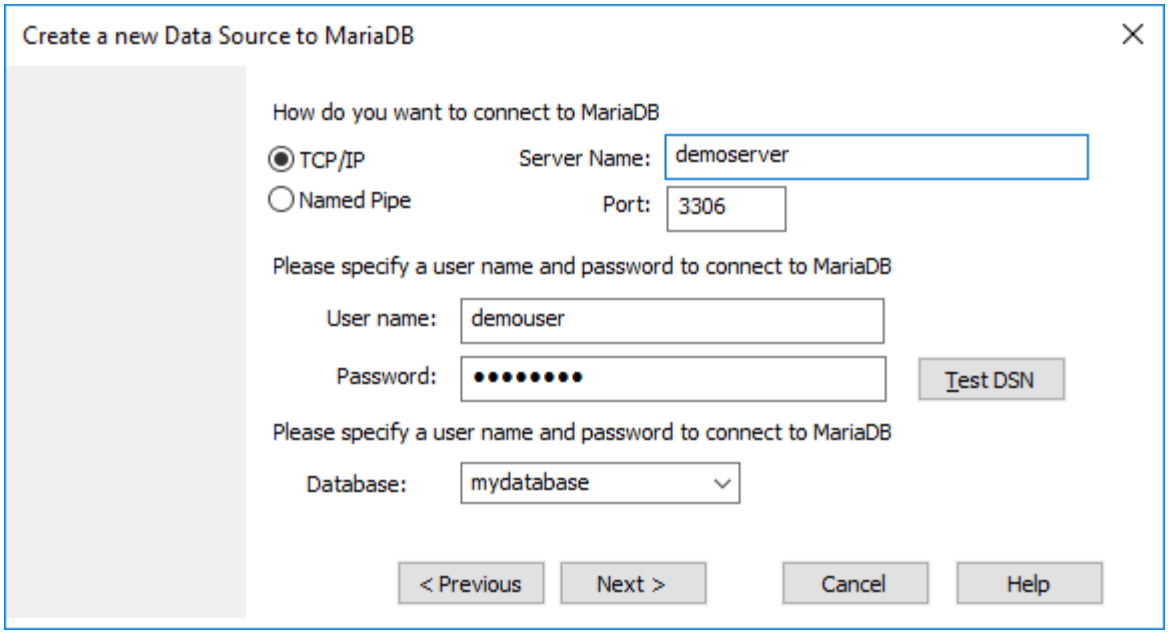
ODBC を使用したMariaDB への接続の方法:

1. データベース接続ウィザードを開始します[848]。
2. 「MariaDB (ODBC)」を選択して、「次へ」をクリックします。

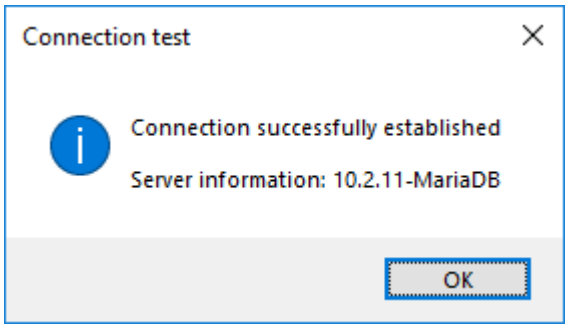
3. 「ドライバーを使用して新規のデータソース名 (DSN) を作成する」を選択し、「MariaDB ODBC 3.0 Driver」を選択します。このようなドライバーがリスト内で使用できない場合、「ドライバーの編集」をクリックし、使用することのできるMariaDB ドライバーを選択します (リストには使用中のオペレーティングシステム上にインストールされているすべてのODBC ドライバーが含まれています)。
4. 「接続」をクリックします。



5. 名前を入力し、任意で、このODBC データソースを将来識別できる詳細を与えます。



6. データベース接続資格情報(TCP/IP サーバー、ユーザー、パスワード)を入力し、データベースを選択して、「Test DSN」をクリックします。接続に成功すると、メッセージが表示されます:



7. 「次へ」をクリックしてウィザードを完了します。他のパラメータが必要になる場合がありますが（例えば、安全な接続を使用して、MariaDB に接続する場合は、SSL 証明書が必要になります）。

メモ　データベースサーバーがリモートの場合、サーバー管理者が遠隔の接続を使用中のマシンの IP アドレスから受け入れるように構成する必要があります。

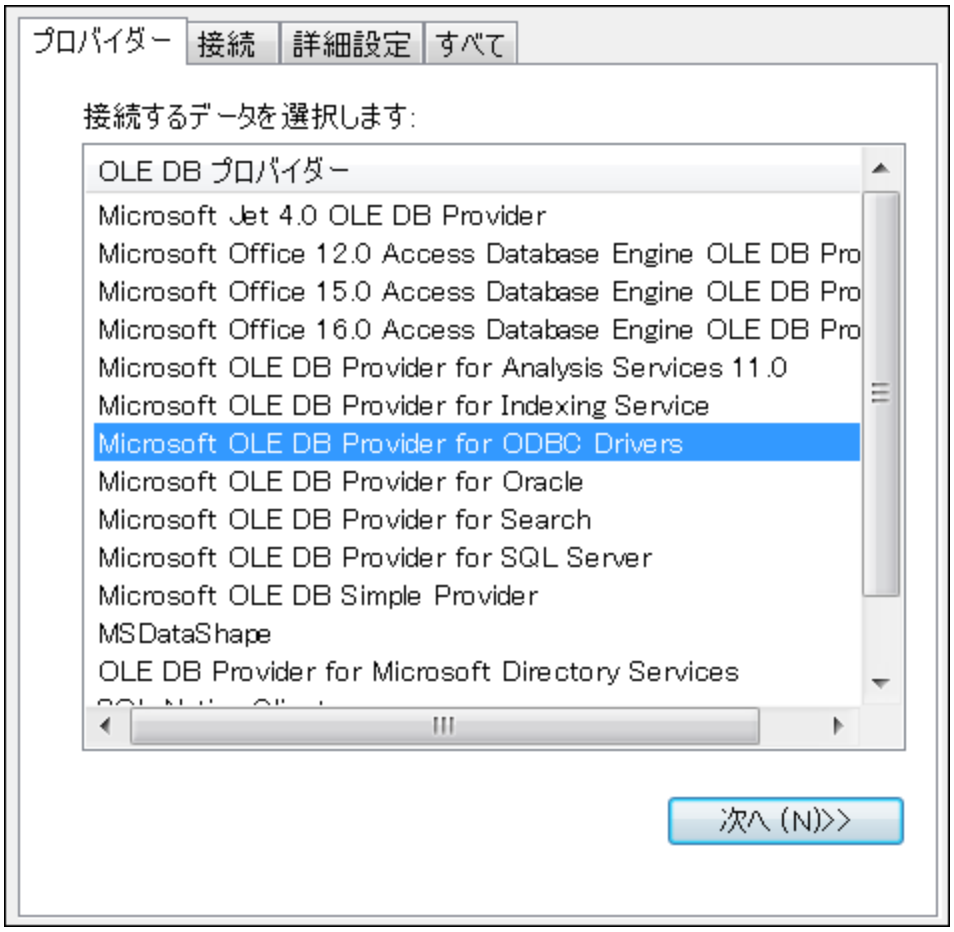
19.1.10.9 Microsoft Access (ADO) への接続

Microsoft Access データベースに接続するには、既存の Microsoft Access データベースに接続する[855]で説明されているとおり、データベースファイルのためのウィザードとブラウザーの手順を踏みます。別の方法としては、このトピックで説明されているとおり、明確的に ADO 接続をセットアップすることです。このアプローチは、データベースがパスワードで保護されている場合、役に立ちます。

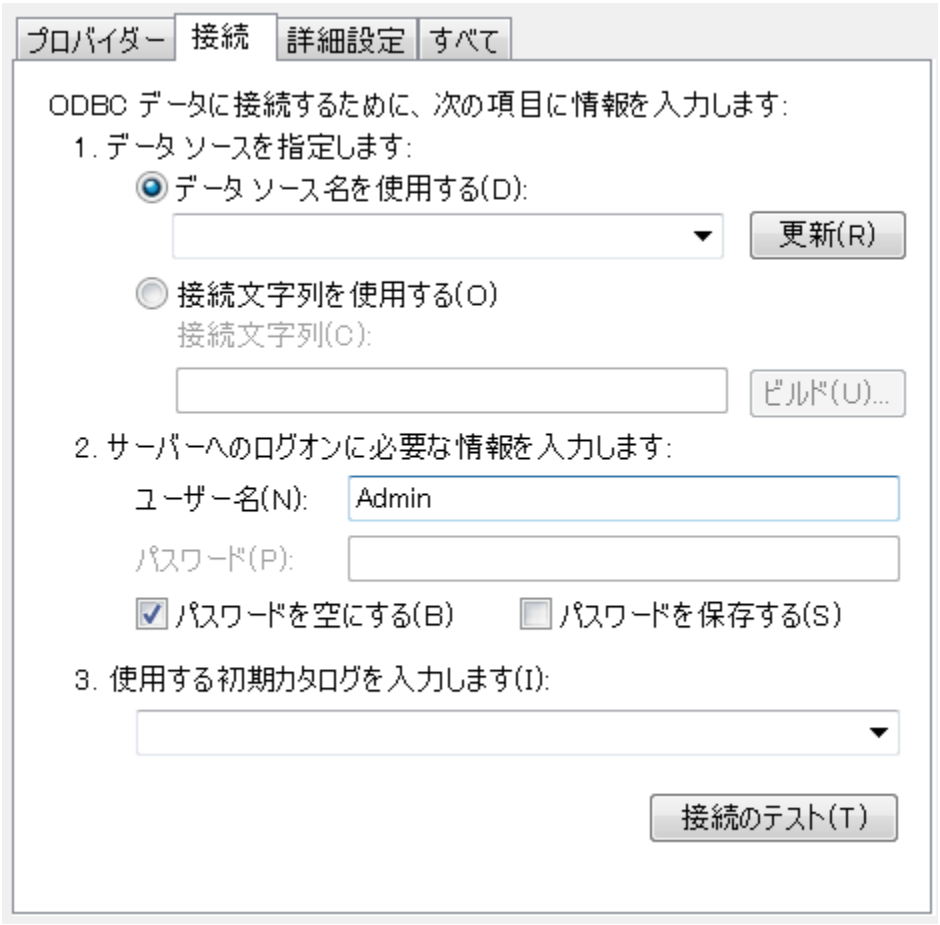
ODBC 接続を使用して、Microsoft Access に接続することもできますが、このシナリオには限度があるため、奨励されません。

パスワードで保護されている Microsoft Access データベースに接続する:

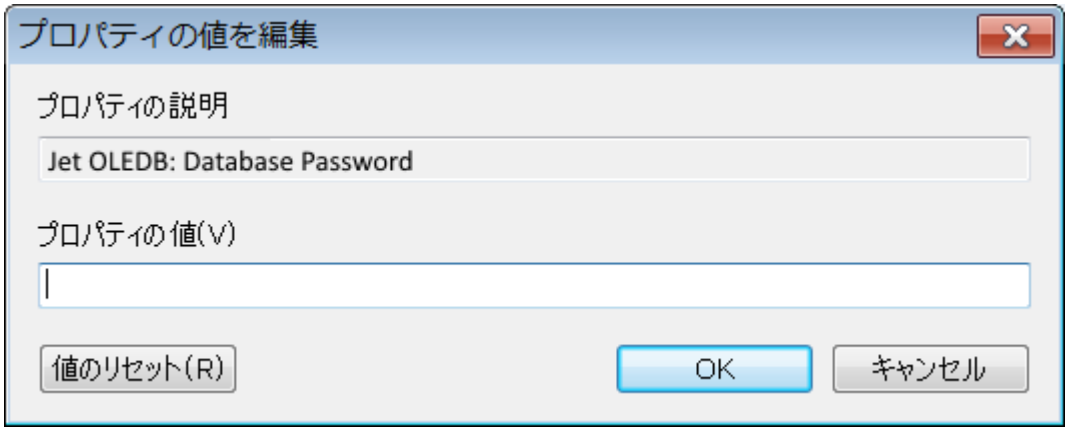
1. データベース接続ウィザードを開始します[848].
2. [ADO 接続] をクリックします。
3. [構築] をクリックします。



4. **[Microsoft Office 15.0 Access Database Engine OLE DB Provider]** を選択して、「次へ」をクリックします。



5. データソースボックス内でUNC フォーマットのMicrosoft Access ファイルへのパスを入力します。例えば、**myserver** がサーバーの名前であり **mynetworkshare** が共有するネットワークの名前である箇所での **\\myserver\\mynetworkshare\Reports\Revenue.accdb**。
6. 全てタブの **Jet OLEDB:Database Password** パスワード プロパティをダブルクリックし、データベースパスワードをプロパティの値として入力します。



メモ また接続できない場合、ワーキンググループ情報ファイル(**System.MDW**) を検索して、ユーザープロフィールに適用できるようにします。**Jet OLEDB: System database** プロパティの値を **System.MDW** ファイルのパスに設定します。

(C) 2015-2021 Altova GmbH

19.1.10.10 Microsoft SQL Server (ADO) への接続

このサンプルでは、ADO を使用したSQL Server データベースへの接続方法について説明されています。 Https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 でダウンロード可能な奨励される **Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)**を使用する場合に適用される説明です。

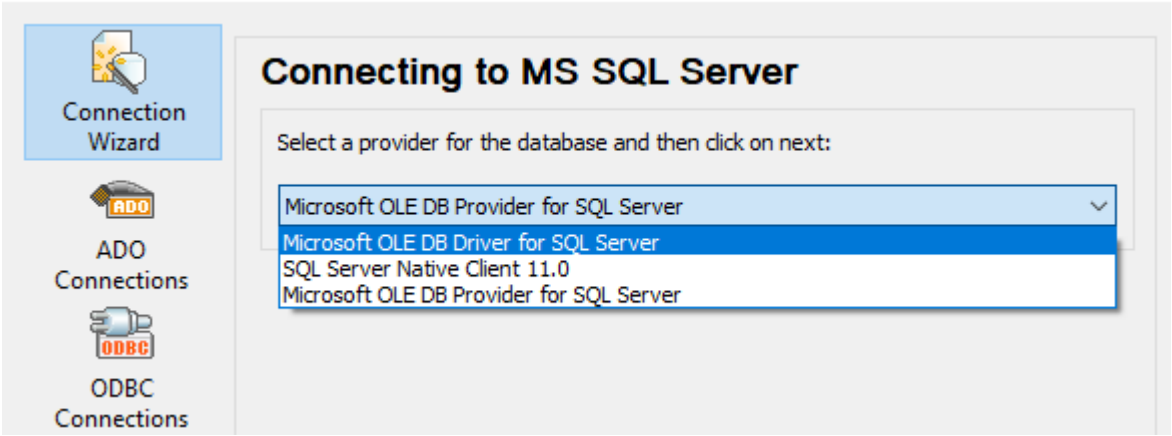
これらの説明を読む前に、ワークステーションに上記のプロバイダーがダウンロードされインストールされていることを確認してください。ADO プロバイダーはXMLSpy (32-ビット または64-ビット)のプラットフォームに一致する必要があることご注意してください。

SQL Server Native Client (SQLNCLI) または**Microsoft OLE DB Provider for SQL Server (SQLOLEDB)** などの他のADO プロバイダーを使用する場合、手順は同様ですが、これらのプロバイダーは奨励されていません。奨励されていないプロバイダーへの接続を成功させるには、SQL Server データリンクプロパティのセットアップ[856]で説明されている追加接続プロパティを設定する必要があります。

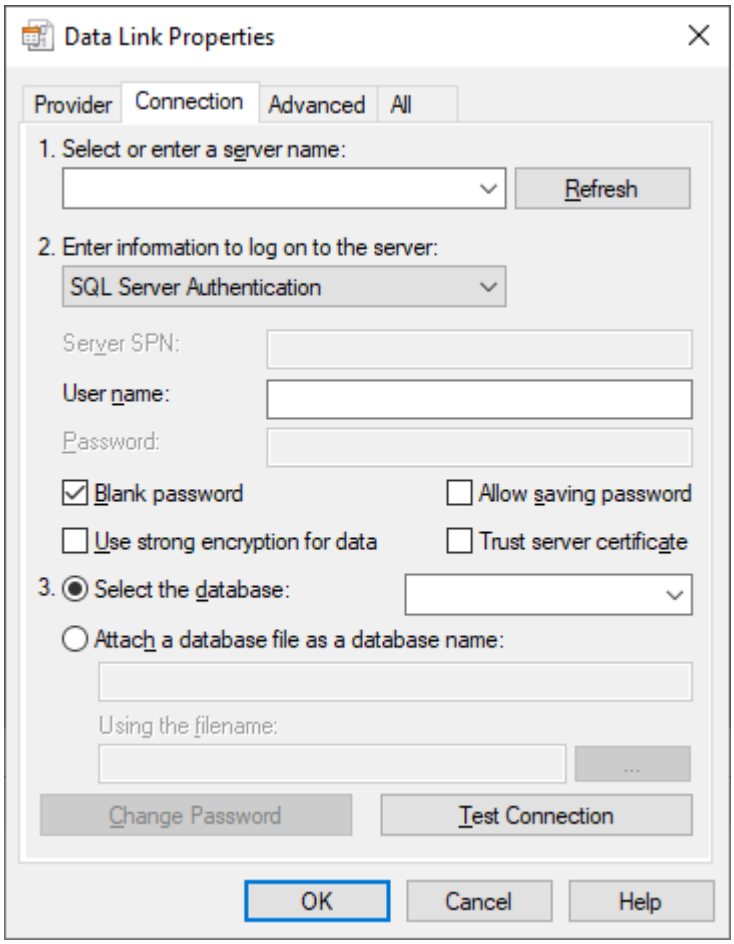
Microsoft OLE DB Provider for SQL Server (SQLOLEDB) は共通テーブル式 (CTE) とネストされたSELECT ステートメントなどの複雑なクエリのパラメーターバインディングに関する問題があることが知られています。

SQL サーバーに接続する方法:

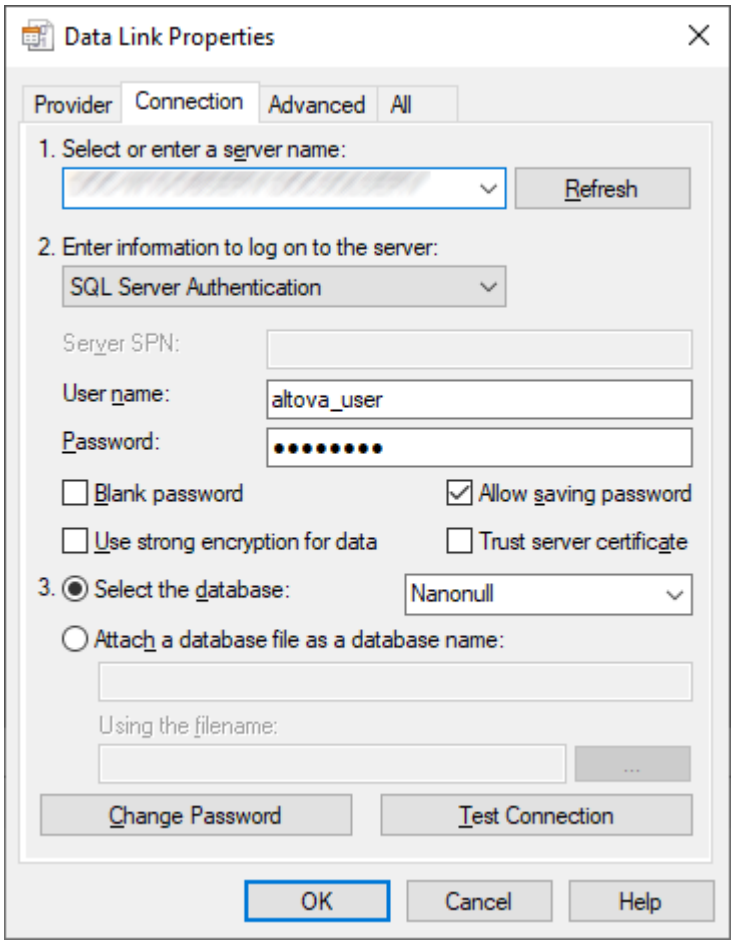
1. データベース接続ウィザードを開始します[848]。
2. **[Microsoft SQL Server (ADO)]** を選択して、[次へ] をクリックします。使用することのできるADO プロバイダーのリストが表示されます。このサンプルでは**Microsoft OLE DB Driver for SQL Server** が使用されています。リストに存在しない場合、上記の通りコンピューターに接続されていることを確認してください。



3. 「次へ」をクリックします。データリンクプロパティダイアログボックスが表示されます。



4. データベースサーバーの名前を選択、または入力します。例えば、**SQLSERV01**。名前を持つSQL Server *インスタンス*に接続する場合、サーバー名は**SQLSERV01\SOMEINSTANCE** になります。
5. データベースサーバーがW indows ドメイン上の認証されているユーザーの接続を許可している場合、「**W indows 認証**」を選択してください。それ以外の場合は、[**SQL Server 認証**]を選択し、[**空のパスワード**] チェックボックスをクリアし、対応するボックスにデータベース証明書を入力します。
6. **パスワードの保存の許可** チェックボックスを選択し、接続するデータベースを選択します（このサンプル内では、Nanonull です）。



7. この時点で接続をテストする場合は、[**接続のテスト**]をクリックします。これは任意の奨励されるステップです。
8. **[OK]**をクリックします。

19.1.10.11 Microsoft SQL Server (ODBC) への接続

このサンプルでは、ODBC を使用したSQL Server データベースへの接続方法について説明されています。

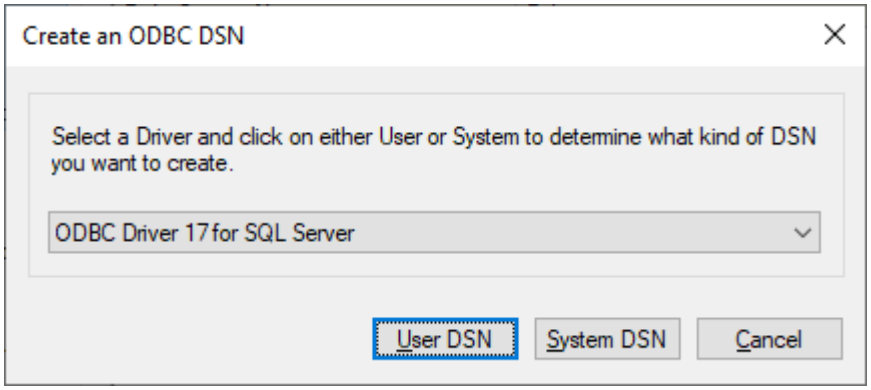
必要条件:

- **Microsoft ODBC Driver for SQL Server** をMicrosoft W eb サイトからダウンロードしてインストールします。https://docs.microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-sql-server を参照してください。このサンプルは**Microsoft ODBC Driver 17 for SQL Server** を**SQL Server 2016** データベースに接続するために使用しています。接続するSQL Server のバージョンにより異なるODBC ドライバーバージョンをダウンロードする必要がある場合があります。使用中のSQL Server によりサポートされるODBC ドライバーバージョンに関する情報は、システムの必要条件を参照してください。

ODBC を使用してのSQL Server への接続:

1. データベース接続ウィザードを開始します[848]。

2. [ODBC 接続] をクリックします。
3. [ユーザーDSN] を選択して、(または、管理者の特権がある場合、 [システムDSN] を選択して、)[追加] をクリックします。
4. ドライバーをリストから選択します。インストール後にのみドライバーはリスト内に表示されることにご注意してください。

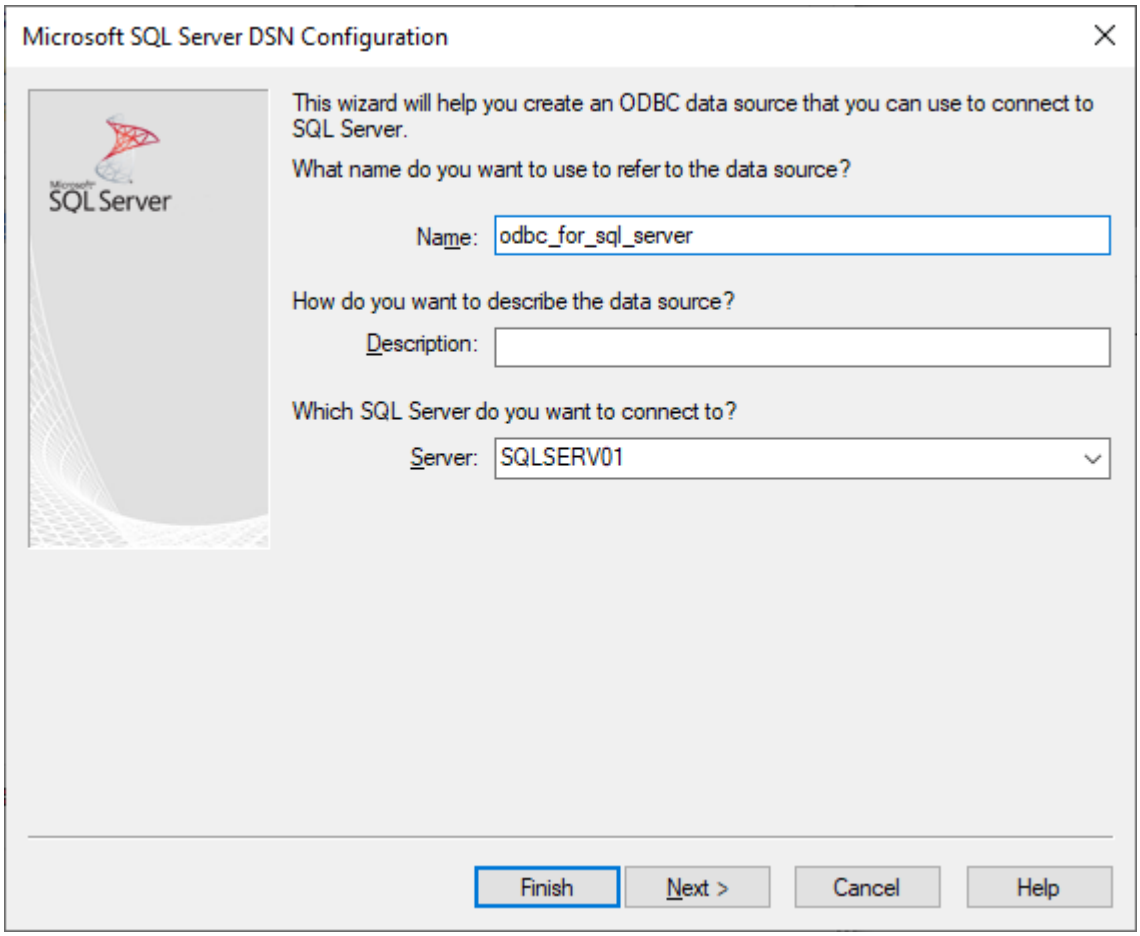


5. [ユーザーDSN] (またはシステムDSN を作成する場合) [システムDSN]をクリックします。

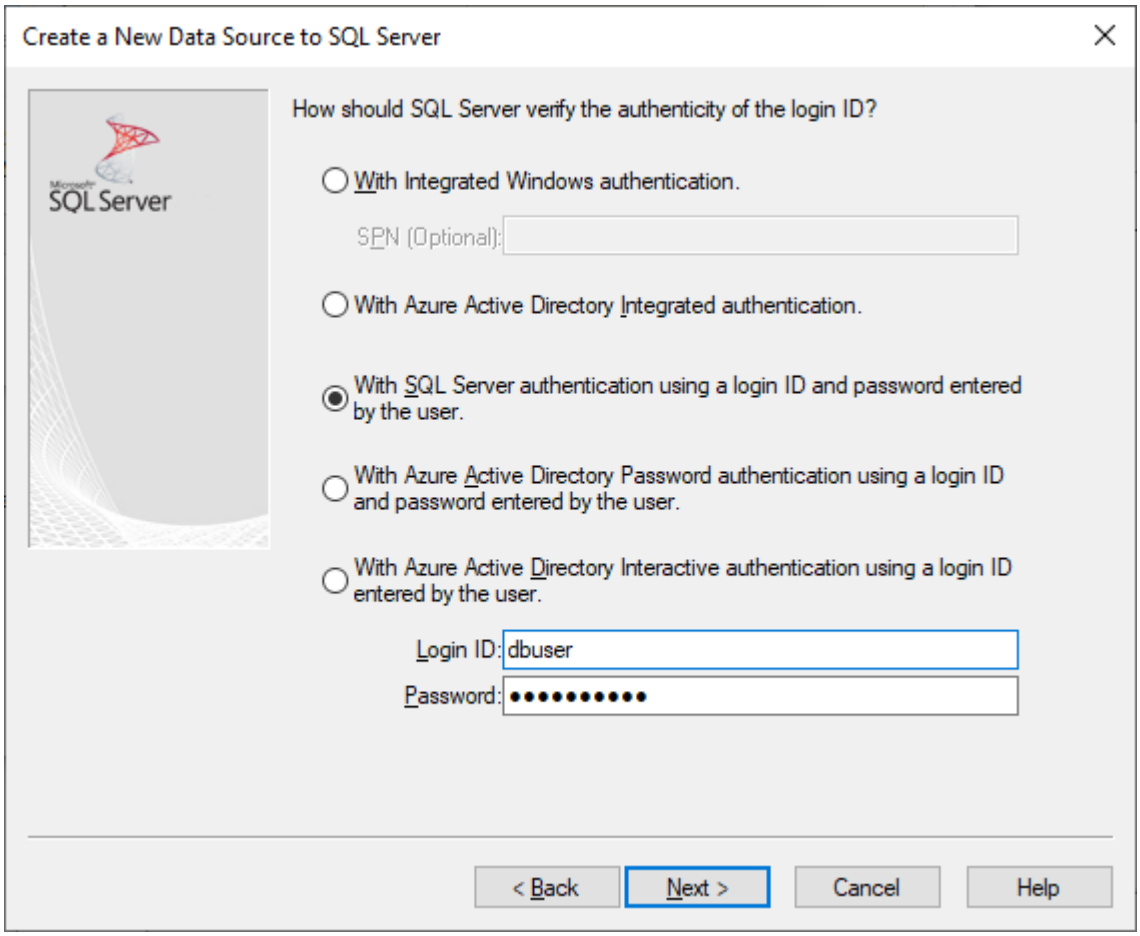
> XMLSpy が管理者として作動することを必要とするシステムDSN の作成方法このため**System DSN** を作成するには、ウィザードをキャンセルし、XMLSpy が管理者として作動していることを確認し、上記のステップを再度実行します。

6. 名前を入力して、任意でこの接続のIDの説明を入力し、接続するSQL Server (この場合 **SQLSERV01**)をリストから選択します。.

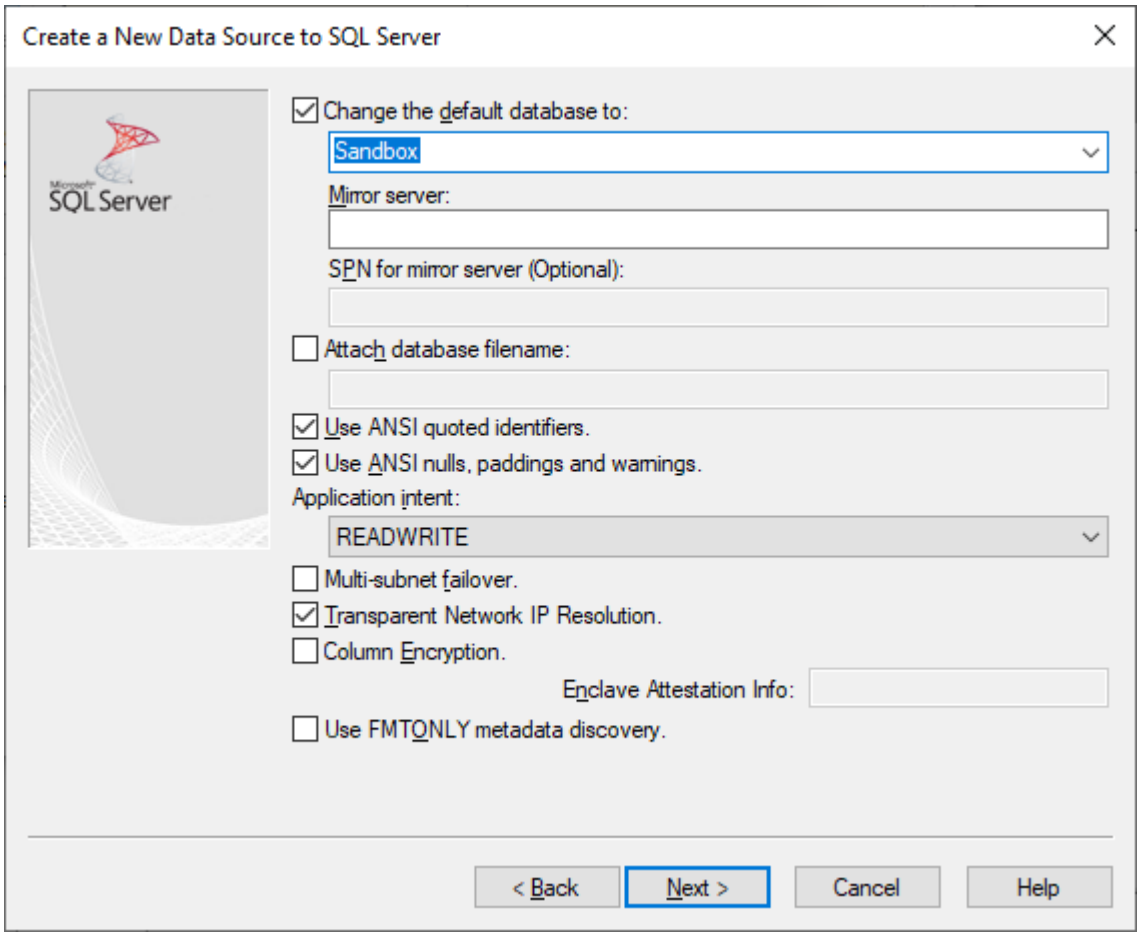
(C) 2015-2021 Altova GmbH



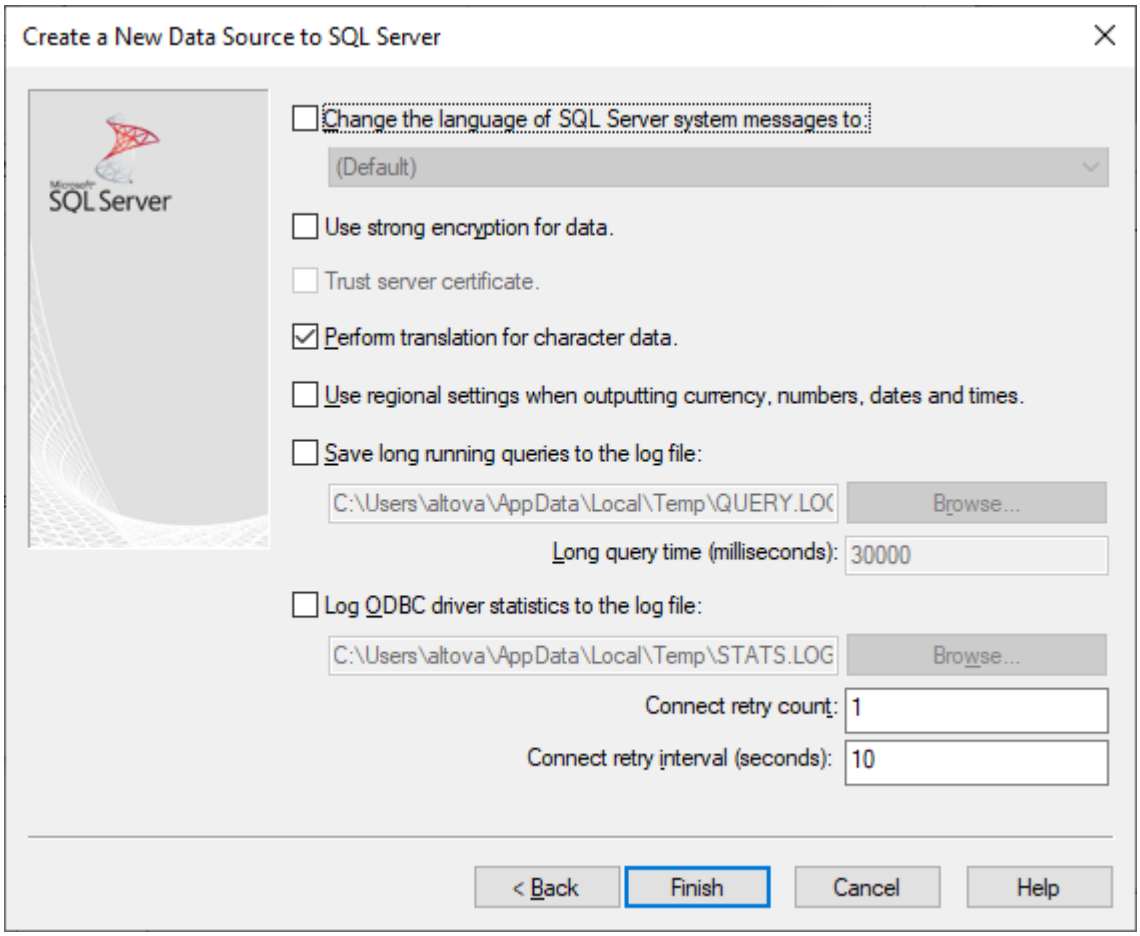
7. データベースサーバーがWindowsドメイン上の認証されているユーザーの接続を許可している場合、「**統合されたWindows認証**」を選択してください。それ以外の場合、適用可能なオプションから1つを選択してください。このサンプルは **SQL Server認証**...を使用しています。これはユーザー名とパスワードが関連するボックスに入力されることを必要としています。



8. オプションで **デフォルトのデータベースを変更** チェックボックスを選択して、接続するデータベースの名前を入力します（このサンプルでは**Sandbox**）。

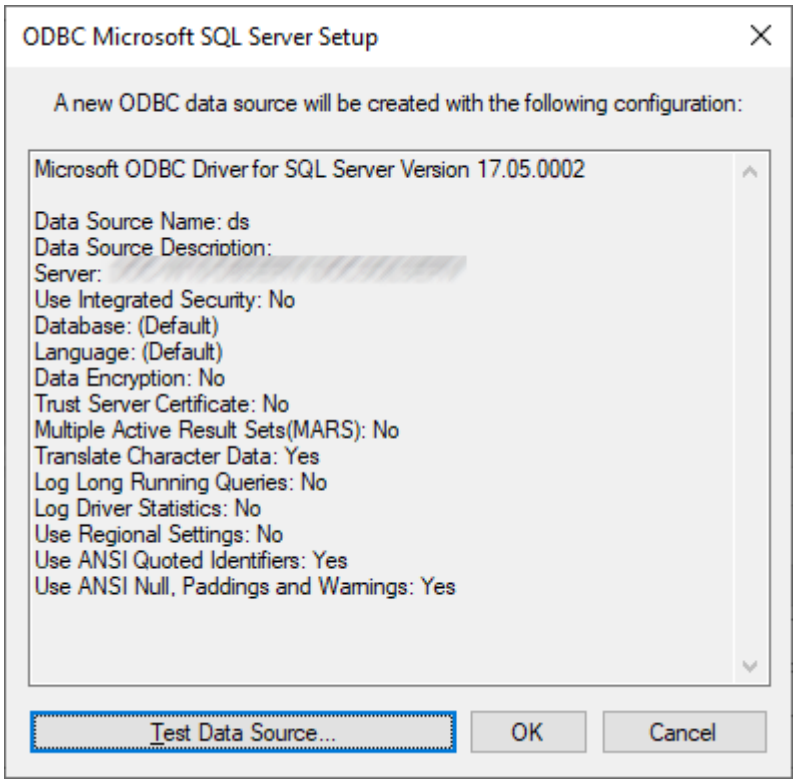


9. [次へ] をクリックして、任意で、この接続のための追加パラメーターを構成します。

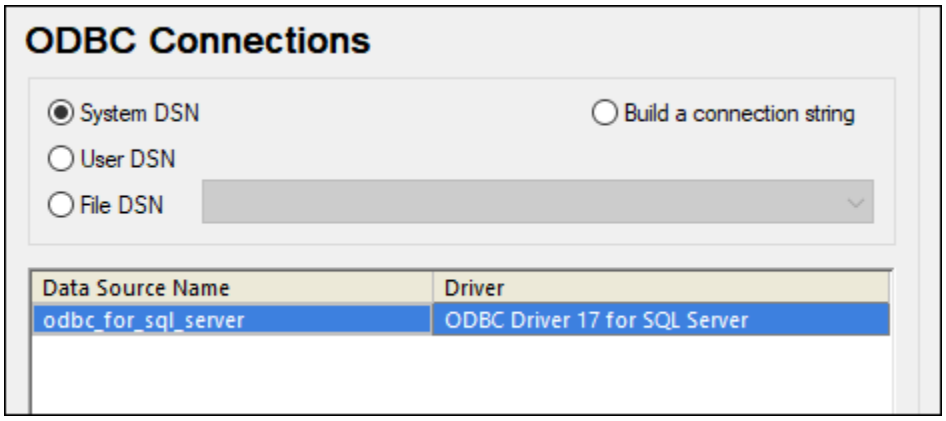


10. 「完了」をクリックします。接続の詳細をリストした確認ダイアログボックスが開かれます。

(C) 2015-2021 Altova GmbH



11. **[OK]**をクリックします。構成された通り ユーザー または システム データソースのリスト内でデータソースが表示されます。例:



19.1.10.12 MySQL (ODBC) への接続

このサンプルではODBC ドライバーを使用して、W indows マシンから MySQL データベースへ接続する方法について説明されています。W indows からは、MySQL ODBC ドライバーを入手することができませんので、個別にダウンロードおよびインストールする必要があります。このサンプルでは、MySQL コネクタ/ODBC 8.0 を使用しています。

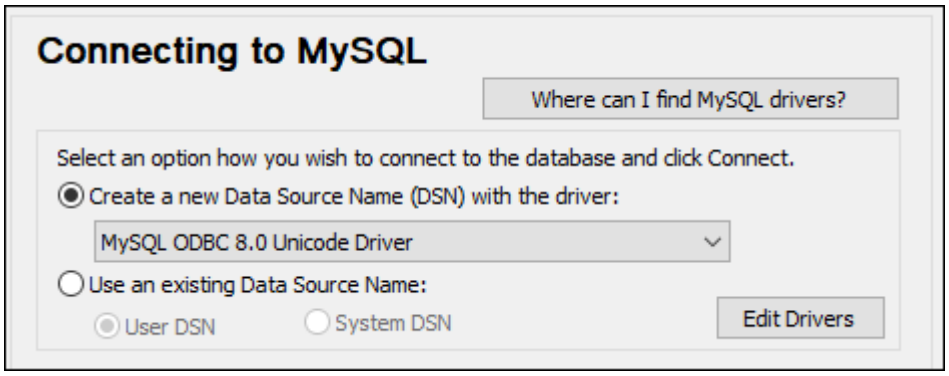
前提条件:

- MySQL ODBC ドライバーをオペレーティングシステム上にインストールする必要があります。使用するデータベースサーバーバージョンのために奨励されるドライバーバージョンのためのMySQLドキュメントを確認してください (https://dev.mysql.com/downloads/connector/odbc/ を確認してください)。
- データベース接続の詳細: ホスト、データベース、ユーザー名およびパスワード

64-ビットプラットフォームのためにMySQL Connector/ODBC をインストールしている場合、64-ビットプラットフォームのために XMLSpy をインストールしてください。

ODBC を介して MySQL に接続する方法:

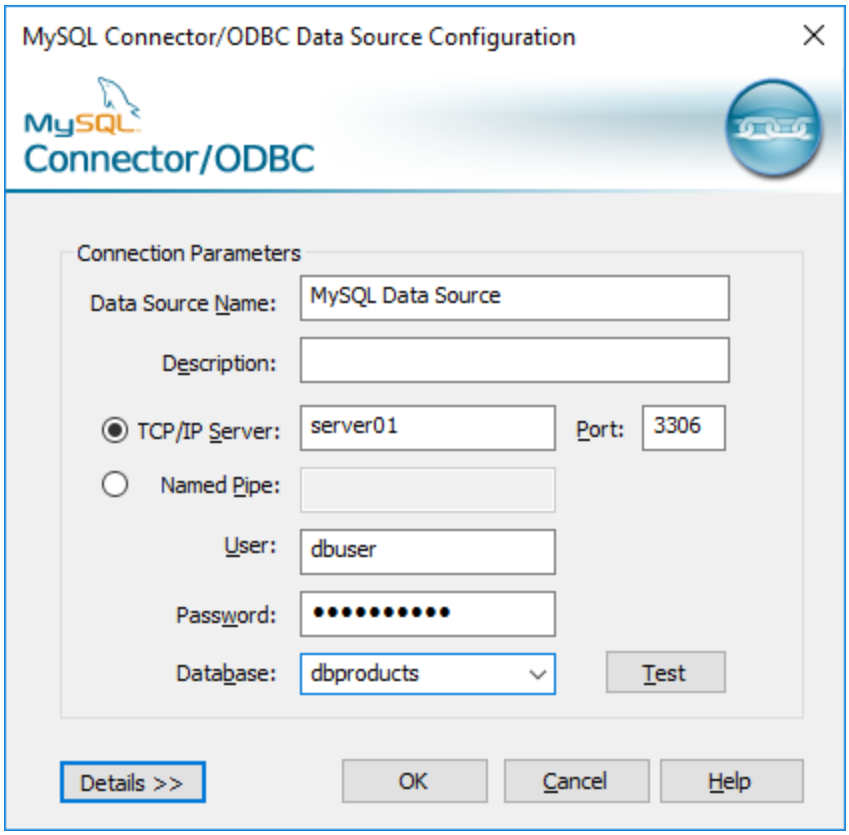
1. データベース接続ウィザードを開始します[848]。
2. **MySQL (ODBC)** を選択して、「次へ」をクリックします。



3. [ドライバーを使用して新しいデータソース名 (DSN) を作成] を選択して、MySQLドライバーを選択します。MySQLドライバーがリストにない場合は、[ドライバーを編集] をクリックして、使用可能なMySQLドライバーを選択します (使用中のオペレーティングシステムにインストールされているすべてのODBC ドライバーがリストされています)。

   XMLSpy 64-ビットをインストールすると、64-ビット ODBC ドライバーがリスト内に表示されます。それ以外の場合、32-ビット ODBC ドライバーが表示されます。使用可能な ODBC ドライバーの確認[867]も参照してください。

4. 「接続」をクリックします。



5. データソース名ボックスの中に、今後このODBC データソースを認識するために役に立つ詳細を入力します。
6. データベース接続資格情報（TCP/IP サーバー、ユーザー、パスワード）を入力して、データベースを選択し、「OK」をクリックします。

メモ　データベースサーバーがリモートの場合、サーバー管理者が使用中のコンピューターのIP アドレスからのリモート接続を承諾するように構成される必要があります。また、[詳細>>] をクリックすると使用することのできる追加パラメーターが表示されます。デフォルトの値を変更する前に、ドライバーのドキュメントを確認してください。

19.1.10.13　Oracle (JDBC) への接続

JDBC インターフェイスを使用して、クライアントマシンから、Oracle データベースサーバーへの接続の方法を説明します。接続は、Oracle Web サイトから使用することのできる、**Oracle Instant Client Package (Basic)** を使用する、Java 接続として作成されます。接続の型の利点は、Oracle Instant Client Package により与えられる Java 環境と .jar ライブラリのみを必要とし、更に複雑なデータベースクライアントをインストールする手間を省くことができる点です。

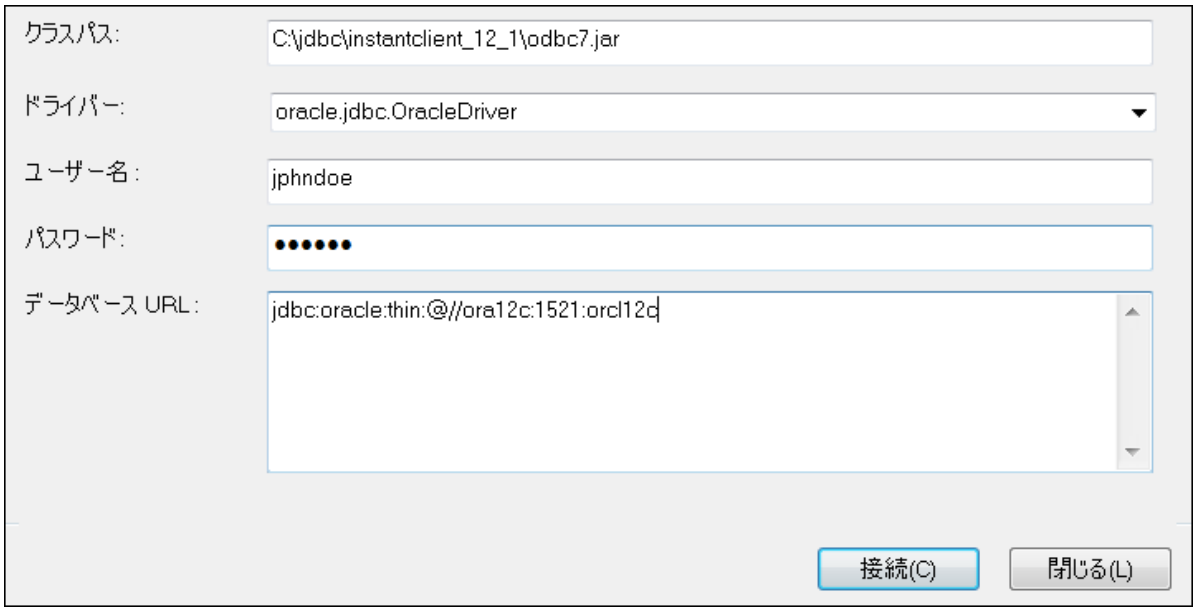
前提条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーションオプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- **Oracle Instant Client Package (Basic)** が使用中のオペレーティングシステムで使用可能である必要があります。パッケージは、Oracle の公式 Web サイトからダウンロードすることができます。このサンプルは、Windows 32-ビットのための Oracle Instant Client Package バージョン 12.1.0.2.0 、そして結果的に Oracle JDK 32-ビットを使用します。

- データベース接続の詳細が必要です: ホスト、データベース パス、またはエイリアス、ユーザー名、および パスワード。

クライアント パッケージ を使用して Oracle に接続する:

1. データベース接続ウィザードを開始します[848]。
2. 「JDBC 接続」をクリックします。
3. 「クラス パス」の横にデータベースへの接続を与える .jar ファイルへの パスを入力します。必要であれば、セミコロンで区切られた .jar ファイル パスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次の パスを使用して見つけることができます: **C:\jdbc\instantclient_12_1\ojdbc7.jar**。.jar ファイル パスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラス パス」テキスト ボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成[870])。
4. 「ドライバー」ボックスから、**oracle.jdbc.OracleDriver** または **oracle.jdbc.driver.OracleDriver** を選択します。このエントリは、有効な .jar ファイル パスが、「クラス パス」テキスト ボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます (前のステップを参照してください)。
5. データベースへのユーザー名と パスワードを対応するテキスト ボックスに入力します。



6. ハイライトされた値を、使用するデータベース サーバーに適用する値と置き換えて、データベース URL テキスト ボックス内にデータベース サーバーへの接続文字列を入力します。

```
jdbc:oracle:thin:@//host:port:service
```

7. 「接続」をクリックします。

19.1.10.14 Oracle (ODBC) への接続

このサンプルでは、ローカル オペレーティング システム上にインストールされている Oracle データベース クライアントを介して、ネットワーク マシン上で XMLSpy から Oracle データベース サーバーへの接続のよく使用されている状況について説明されています。

このサンプルでは、XMLSpy 内のデータベース接続ウィザードを使用して、ODBC データソース (DSN) のセットアップについて説明されています。既に DSN が作成されている場合は Windows 内の **ODBC データソース管理者** から直接作成しプロンプトされたら選択する事も可能です。ODBC データソースの詳細に関しては、ODBC 接続のセットアップ[865] を参照してください。

必要条件:

- (ODBC Oracle ドライバーを含む) Oracle データベースクライアントが使用中のオペレーティングシステムでインストール、および構成されていなければなりません。Oracle データベースクライアントのインストールおよび構成方法は Oracle ソフトウェアに付随するドキュメントを参照してください。
- **tnsnames.ora** ファイルのパスは Oracle ホームディレクトリがインストールされている場所によります。Oracle データベースクライアント 11.2.0 に関しては、デフォルト Oracle ホームディレクトリのパスは以下の通りです:

```
ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = server01)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SID = orcl)
      (SERVER = DEDICATED)
    )
  )
```

tnsnames.ora ファイルのパスは Oracle ホームディレクトリがインストールされている場所によります。Oracle データベースクライアント 11.2.0 に関しては、デフォルト Oracle ホームディレクトリのパスは以下の通りです:

```
C:\app\username\product\11.2.0\client_1\network\admin\tnsnames.ora
```

接続の詳細を貼り付け保存する、または使用可能であれば Oracle *ネット構成アシスタントウィザード* を実行することにより **tnsnames.ora** ファイルに新規エントリを追加することができます。

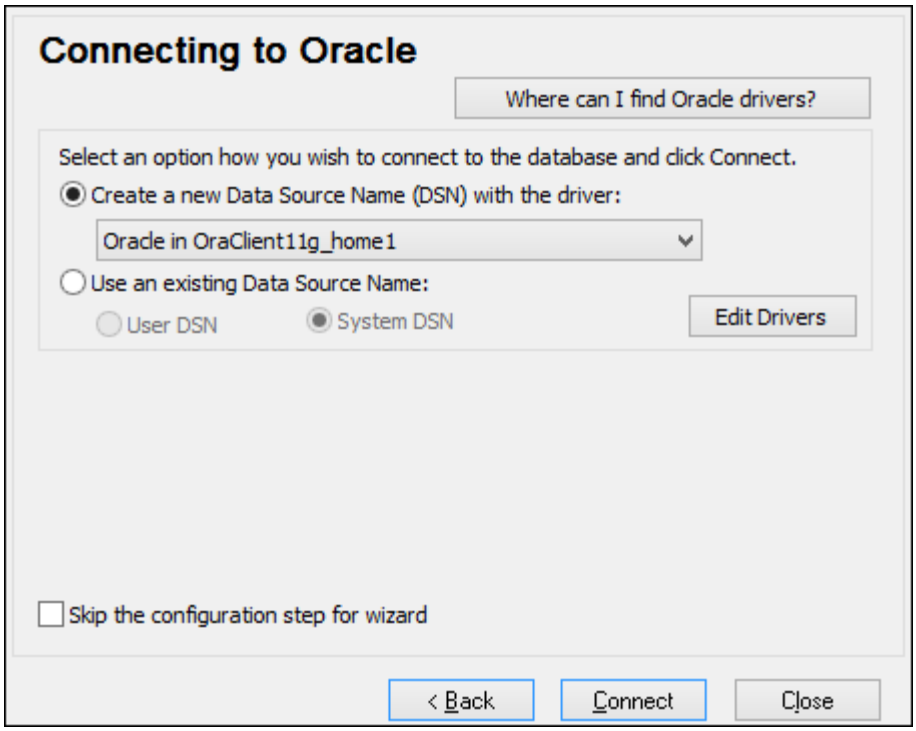
ODBC を使用しての Oracle への接続:

1. データベース接続ウィザードを開始します[848].
2. **[Oracle (ODBC / JDBC)]** を選択して、「次へ」をクリックします。

3. **[ODBC]** を選択します。

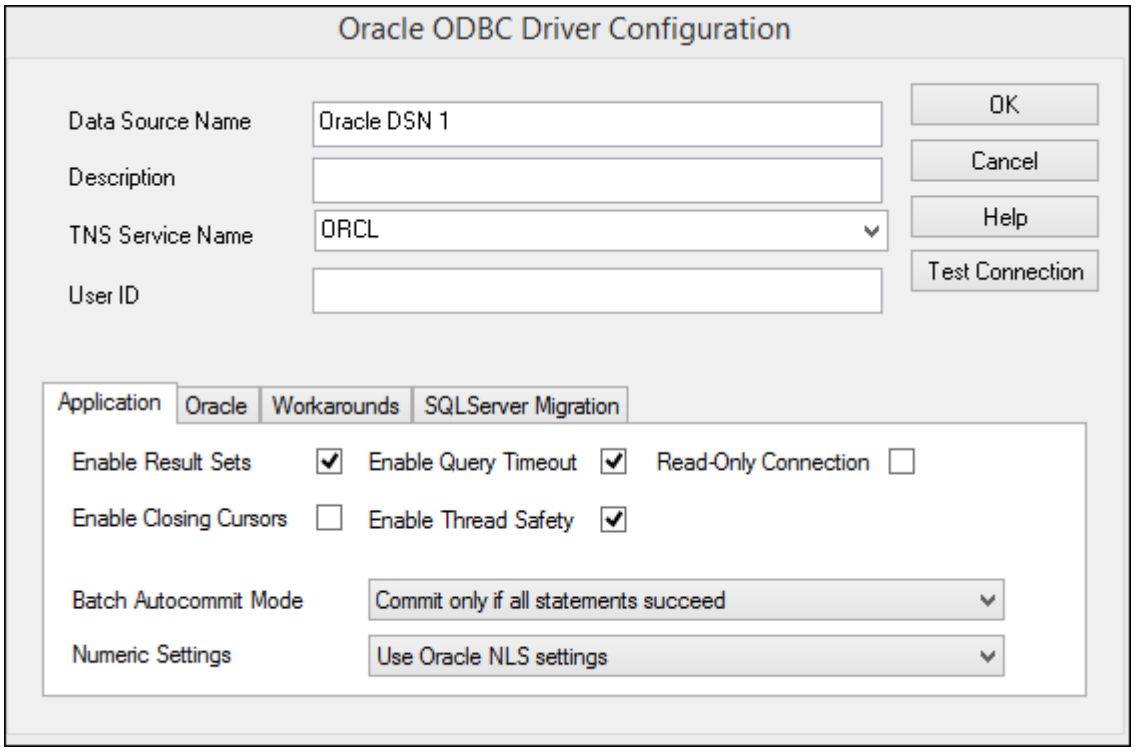
4. [ドライバーの編集] をクリックします。

5. 使用するOracleドライバーを選択します(このサンプルでは**Oracle in OraClient11g_home1**です)。Oracle クライアントをインストール後システムで使用可能なOracleドライバーのリストを表示します。
6. [戻る] をクリックします。
7. **[新しいデータソース(DSN) をドライバーを使用して作成]** を選択して、ステップ4 で選択されたOracleドライバーを選択します。

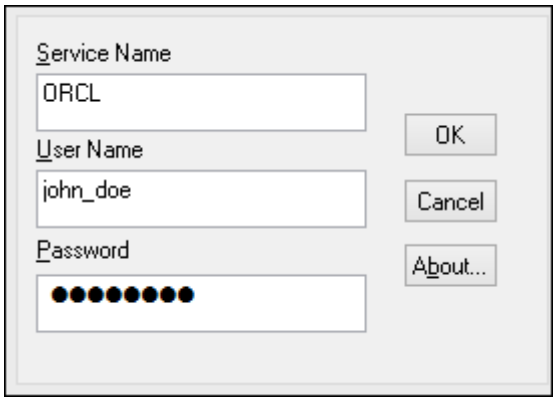


Microsoft ODBC for Oracle という名称のMicrosoftにより提供されるドライバーを使用することは避けてください。Microsoftは Oracle により提供されるODBCドライバーを使用することを奨励しています。(以下参照: http://msdn.microsoft.com/en-us/library/ms714756%28v=vs.85%29.aspx)。

8. 「接続」をクリックします。



9. データソース名テキストボックスにデータソースを識別する名前を入力します（このサンプルの場合は**Oracle DSN 1** です）。
10. TNS サービス名ボックスでは、**tnsnames.ora** ファイルで定義されている通りの接続名を入力します。（prerequisites 913 前提条件を参照してください）。このサンプルでは接続名 **ORCL** です。
11. 「OK」をクリックします。



12. データベースのユーザー名とパスワードを入力して、「OK」をクリックします。

19.1.10.15 PostgreSQL (ODBC) への接続

このトピックではODBC ドライバーを使用した、Windows マシンから PostgreSQL データベースへの接続のサンプル手順について説明されています。Windows からは、PostgreSQL ODBC ドライバーを入手することができませんので、個別にダウンロードおよびインストールする必要があります。このサンプルでは、公式 Web サイトからダウンロードされたpsqlODBC ドライバー（バージョン 11.0）を使用しています。（データベースドライバー概要 849 も参照してください）。

メモ　(ODBC ドライバー無しで直接 PostgreSQL データベースサーバーに接続するには、次を参照してください: PostgreSQL 接続のセットアップ[872]。

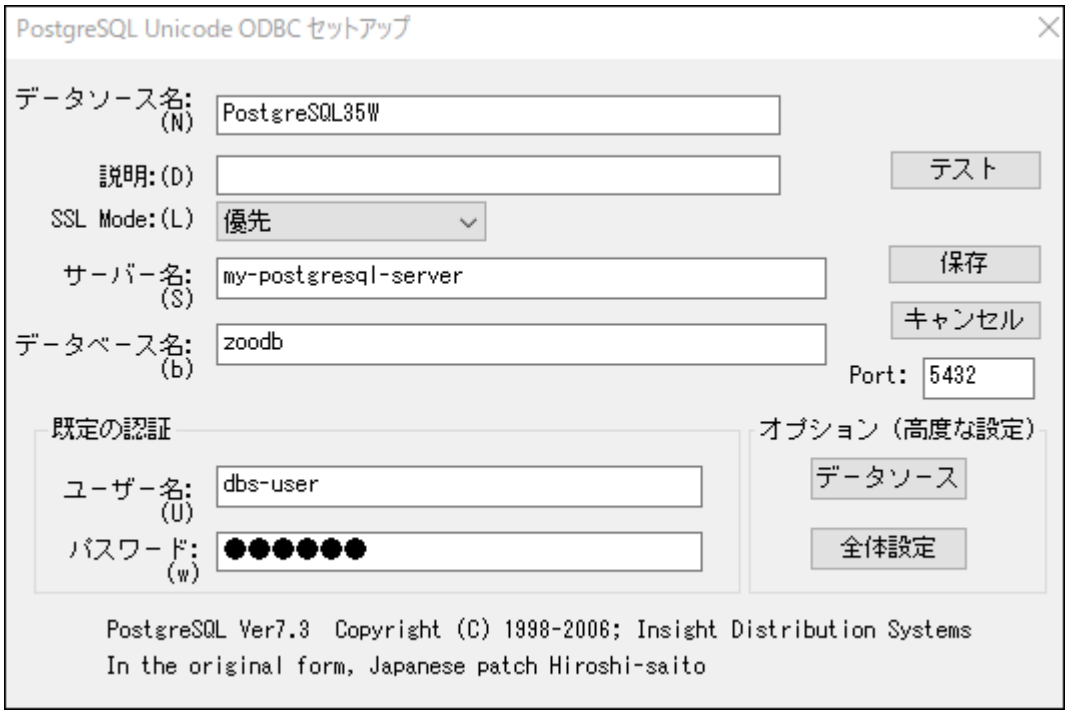
前提条件:

- *psqlODBC* ドライバーを使用中のオペレーティングシステムにインストールする必要があります。
- データベース接続の詳細: ホスト、データベース、ユーザー名およびパスワード

ODBC を使用した PostgreSQL への接続をセットアップする方法:

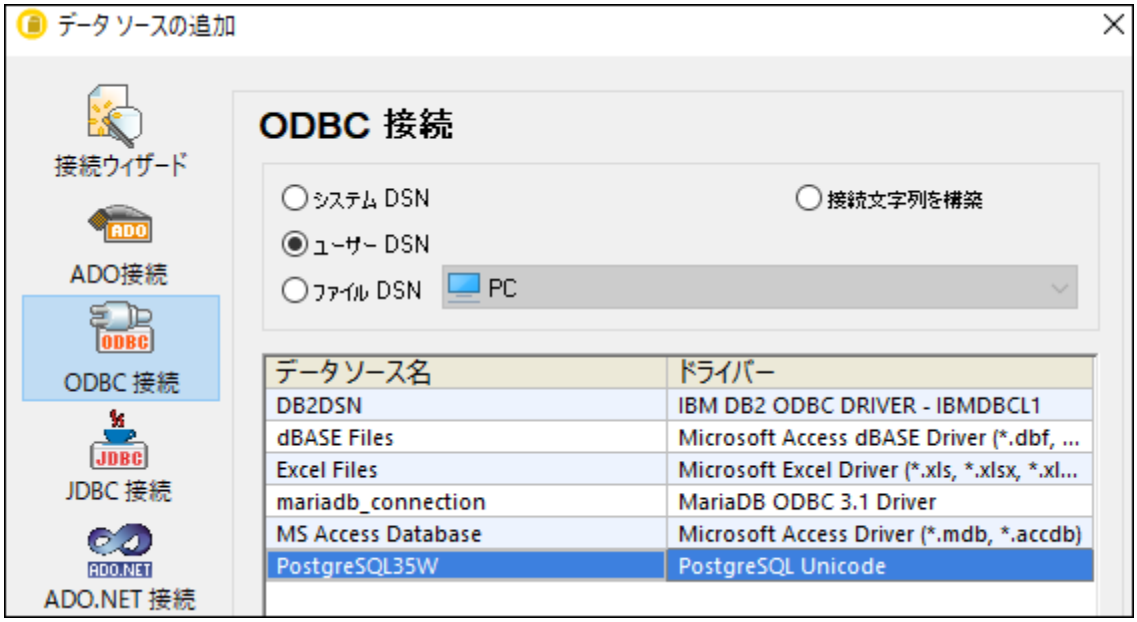
1. データベース接続ウィザードを開始します。[848]
2. **PostgreSQL (ODBC)** をクリックします。
3. **ユーザー DSN** オプションを選択します。
4. 「**新規の DSN の作成**」 をクリックし、ドロップダウンリストからドライバーを選択します。PostgreSQL ドライバーがリスト内で使用できない場合、上記の必要条件で説明されているとおり、PostgreSQL ODBC ドライバーがオペレーティングシステム上にインストールされていることを確認してください。

5. 「**ユーザー DSN**」をクリックします。



6. データベース資格情報を入力し、「保存」をクリックします（これらはデータベースオーナーにより提供されています）。

接続はODBC 接続のリスト内で使用できるようになります。データベースに接続するには、接続をダブルクリックし、「接続」をクリックします。



19.1.10.16 Progress OpenEdge (JDBC) への接続

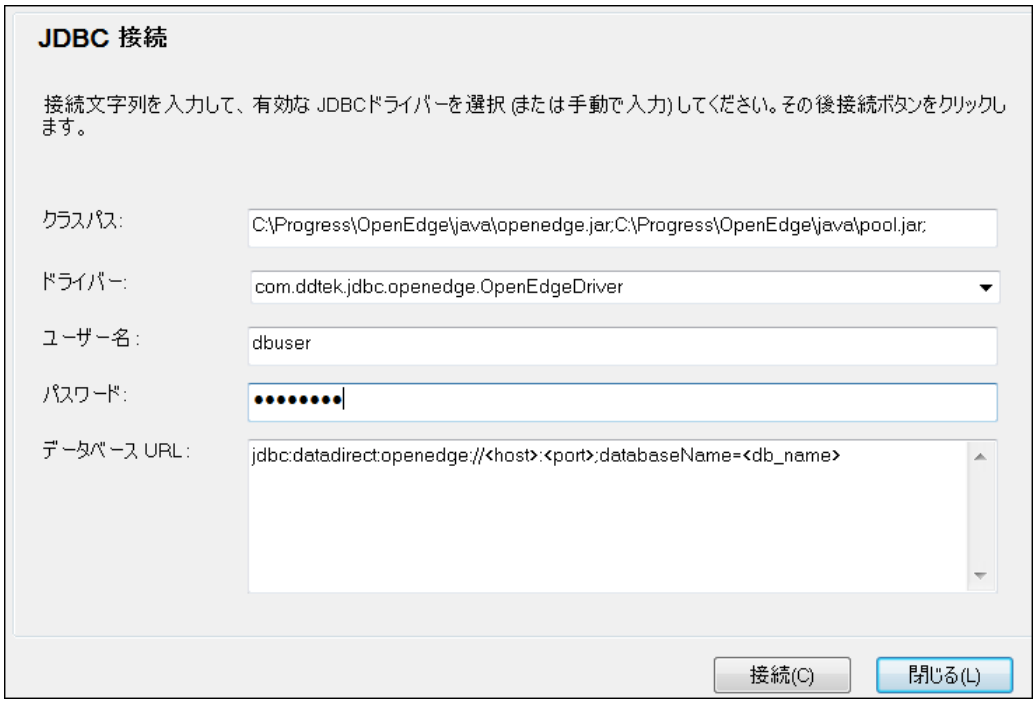
JDBC を使用してProgress OpenEdge 11.6 データベースサーバーへの接続するためのサンプルの手順が説明されています。

前提条件:

- JRE (Java Runtime Environment) またはJava Development Kit (JDK) がインストールされている必要があります。これはOracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションからJava Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタムJVM パス b) W indows レジストリ内で見つかったJVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームがJRE/JDK のプラットフォームに一致することを確認してください。
- オペレーティングシステムの`PATH` 環境変数は、JRE またはJDK インストールの`bin` ディレクトリへのパスを含んでいる必要があります。例、`C:\Program Files (x86)\Java\jre1.8.0_51\bin`.
- Progress OpenEdge JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります。このサンプルでは、JDBC 接続は、**C:\Progress\OpenEdge\java** にOpenEdge SDK のインストールの一部として存在する **openedge.jar** と **pool.jar** ドライバーコンポーネントファイルにより与えられています。.
- データベース接続の詳細があります: ホスト、データベース パス、またはエイリアス、ユーザー名、および パスワード。

JDBC を使用して OpenEdge へ接続する:

1. データベース接続ウィザードを開始します[848].
2. 「**JDBC 接続**」をクリックします。
3. 「**クラス パス**」の横にデータベースへの接続を与える.jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた.jar ファイル パスのリストを入力することもできます。このサンプルでは、必要とされる.jar ファイルは次のパスを使用して見つけることができます`C:\Progress\OpenEdge\java\openedge.jar;C:\Progress\OpenEdge\java\pool.jar;`。.jar ファイル パスをオペレーティングシステムのCLASSPATH 環境変数に追加した場合、「クラス パス」テキストボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成[870])。
4. 「**ドライバー**」ボックスから、**com.ddtek.jdbc.openedge.OpenEdgeDriver** を選択します。このエントリは、有効な .jar ファイル パスが、「クラス パス」テキストボックス内、またはオペレーティングシステムのCLASSPATH 環境変数内で検出された場合のみ使用することができます (前のステップを参照してください)。



5. データベースへのユーザー名とパスワードを対応するテキストボックスに入力します。
6. ハイライトされた値を、使用するデータベースサーバーに適用する値と置き換えて、データベースURL テキストボックス内にデータベースサーバーへの接続文字列を入力します。

```
jdbc:datadirect:openedge://host:port;databaseName=db_name
```

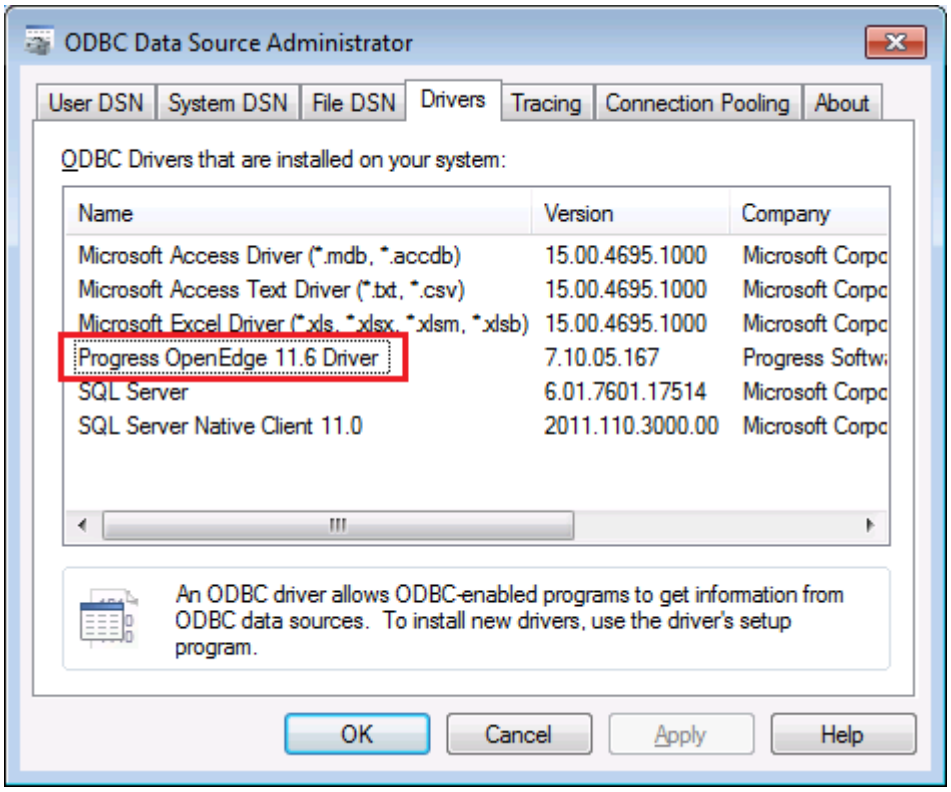
7. 「接続」をクリックします。

19.1.10.17 Progress OpenEdge (ODBC) への接続

このサンプルではProgress OpenEdge 11.6 ODBC ドライバーを使用して、Progress OpenEdge データベースサーバーへの接続の簡単な手順について説明されています。

前提条件:

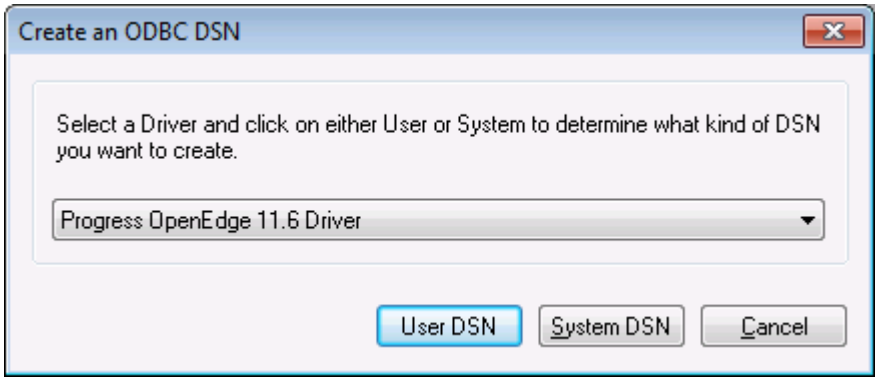
- *Progress OpenEdge ドライバーのための ODBC コネクタ*が使用中のオペレーティングシステムにインストールされている必要があります。Progress OpenEdge ODBC ドライバーは、ベンダーのW eb サイトからダウンロードすることができます(次も参照: データベースドライバーの概要 849)。XMLSpy の32 ビットバージョンを作動する場合は、32 ビット ドライバーをダウンロードし、64 ビットバージョンを作動する場合は、64 ビット ドライバーをダウンロードしてください。インストールの後、ODBC ドライバーが使用中のマシンで使用することができるかを確認してください(次も参照: 使用することのできるODBC ドライバーを確認する 867)。



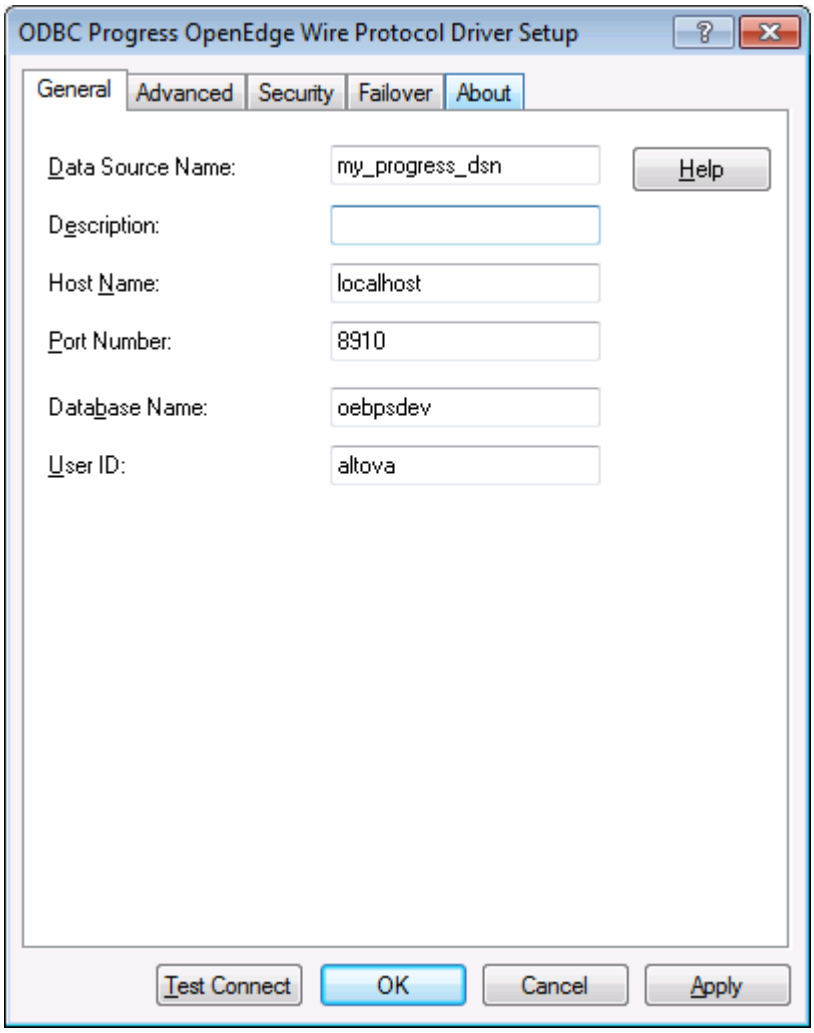
- データベース接続の詳細があります: ホスト名、ポート番号、データベース名、ユーザーID、および パスワード。

ODBC を使用して Progress OpenEdge に接続する:

1. データベース接続ウィザードを開始します[848].
2. 「ODBC 接続」をクリックします。
3. 「ユーザーDSN」をクリックします。(または、次の手順が類似している「システムDSN」、または、「ファイルDSN」をクリックします)。
4. 「追加」をクリックします。
5. 「Progress OpenEdge ドライバー」をリストから選択し、「ユーザーDSN」(または、適用できる場合は、「システムDSN」)をクリックします。

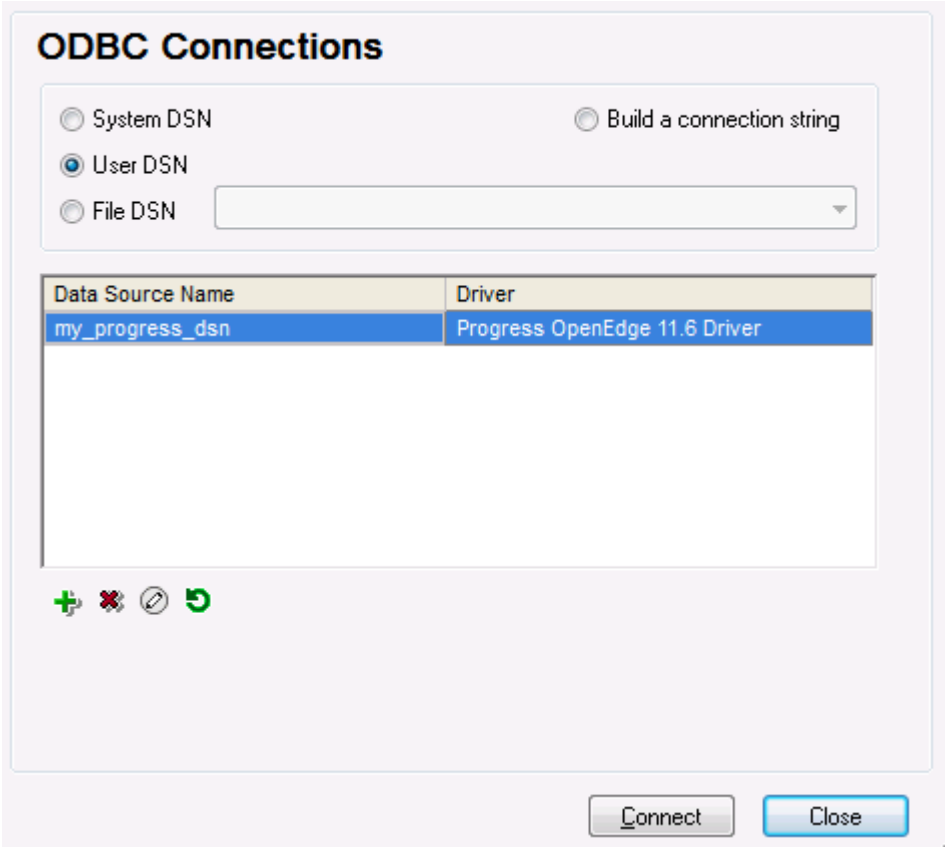


6. データベース接続の資格情報（データベース、サーバー、ポート、ユーザー名、パスワード）を入力し、「OK」をクリックします。入力されたデータを保存する前に、接続を検証するには、「**接続をテストする**」をクリックします。



7. 「OK」をクリックします。新規データソースがODBC データソースのリストに表示されます。

(C) 2015-2021 Altova GmbH



8. 「接続」をクリックします。

19.1.10.18 Sybase (JDBC) への接続

このサンプルでは、JDBC を使用して Sybase データベースサーバーに接続する手順について説明されています。

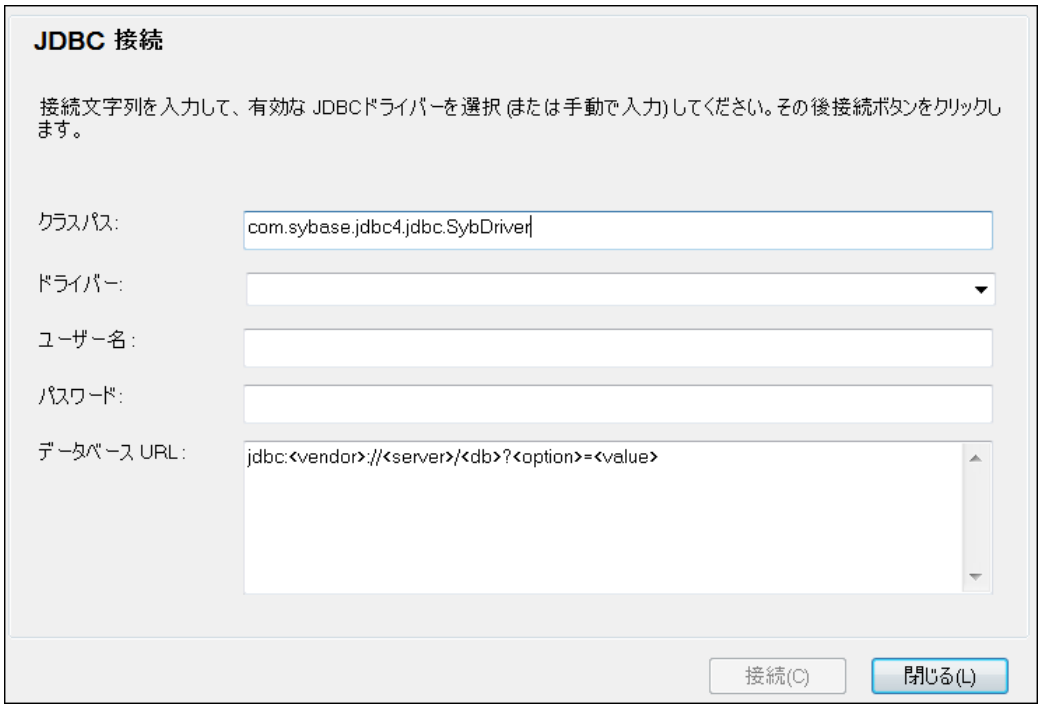
前提条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーションオプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) JAVA_HOME 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- Sybase *jConnect* コンポーネントがオペレーティングシステムにインストールされている必要があります (このサンプルでは *jConnect 7.0* が使用されており、*Sybase Adaptive Server Enterprise PC Client* インストールの一部としてインストールされています)。データベースクライアントのインストール命令に関しては、Sybase ドキュメントを参照してください。
- 次のデータベース接続詳細が存在します: ホスト、データベース、ポート、ユーザー名、および、パスワード。

JDBC を使用して Sybase へ接続する:

1. データベース接続ウィザードを開始します。[848]
2. 「JDBC 接続」をクリックします。

3. 「クラスパス」の横にデータベースへの接続を与える .jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた .jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次のパスを使用して見つけることができます: **C:\sybase\jConnect-7_0\classes\jconn4.jar**。.jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成 870)。
4. 「ドライバー」ボックスから、 を選択します。「ドライバー」ボックスから、**com.sybase.jdbc4.jdbc.SybDriver** を選択します。このエントリは、有効な .jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます (前のステップを参照してください)。



5. データベースのユーザー名とパスワードを対応するテキストボックスに入力します。
6. ハイライトされた値をデータベースサーバーに適用することのできる値と置き換えて、データベース URL テキストボックス内のデータベースサーバーの接続文字列を入力します。

```
jdbc:sybase:Tds:hostName:port/databaseName
```

7. 「**接続**」をクリックします。

19.1.10.19 Teradata (JDBC) への接続

このサンプルでは、JDBC を使用した Teradata データベースへの接続方法について説明されています。

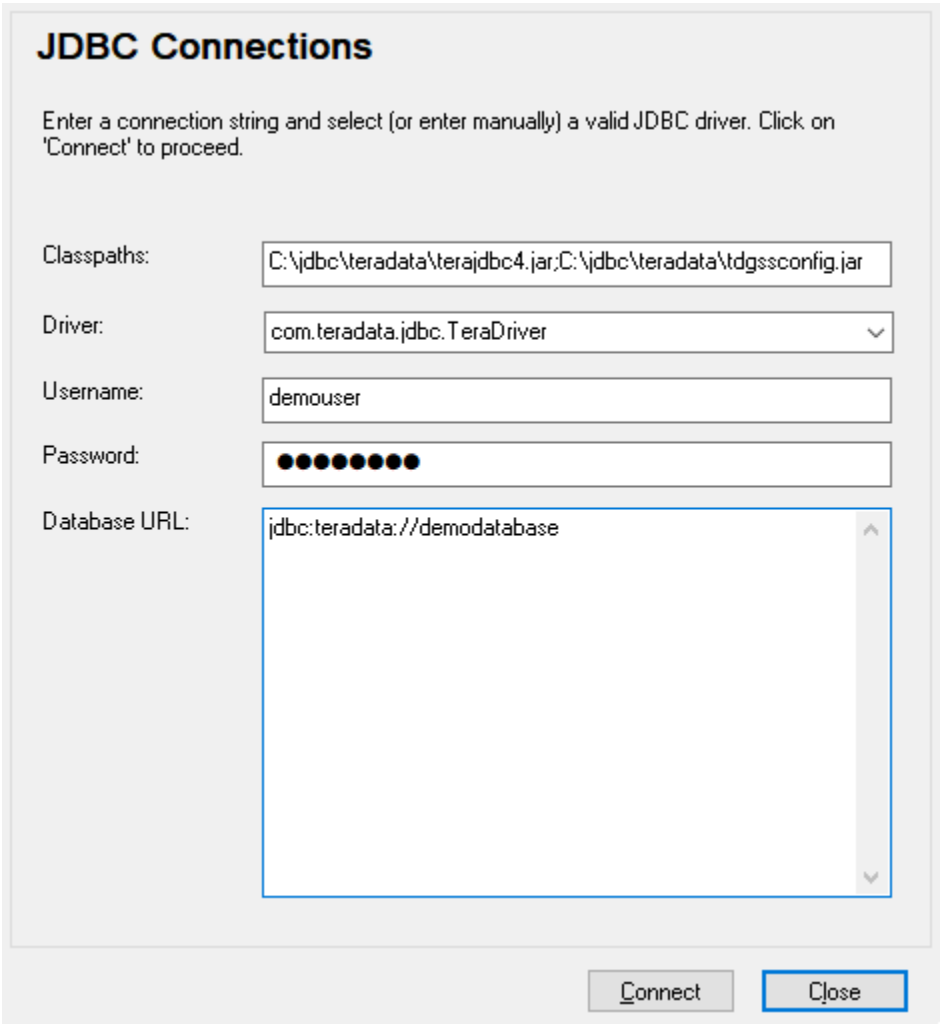
必要条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。XMLSpy は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーションオプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- XMLSpy (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。

- JDBC ドライバー（データベースへの接続を提供する1つまたは複数の.jar ファイル）がオペレーティングシステム上で使用できる必要があります。このサンプルでは、Teradata JDBC ドライバー16.20.00.02 が使用されます。詳細に関しては、次を参照してください：http://downloads.teradata.com/download/connectivity/jdbc-driver。
- 次のデータベース接続詳細が存在します：ホスト、データベース、ポート、ユーザー名、および、パスワード。

JDBC を使用した Teradata への接続の方法:

1. データベース接続ウィザードを開始します[848]。
2. 「JDBC 接続」をクリックします。
3. 「クラスパス」の横に、データベースへの接続を与える.jar ファイルへのパスを入力します。必要な場合、セミコロンで区切られた.jar ファイルパスのリストを入力することができます。このサンプルでは、.jar ファイルは、次のパスの場所に存在します：**C:\jdbc\teradata**。オペレーティングシステムのCLASSPATH 環境変数に.jar ファイルパスが追加されている場合、「クラスパス」テキストボックス空のままにすることができます（次も参照してください：CLASSPATH の構成[870]）。
4. 「ドライバー」ボックス内で、**com.teradata.jdbc.TeraDriver** を選択します。有効な.jar ファイルパスが「クラスパス」テキストボックスで、または、オペレーティングシステムのCLASSPATH 環境変数内に存在すると、使用することができます（前のステップを参照してください）。



5. ユーザー名とパスワードを対応するテキストボックスのデータベース内に入力します。
6. ハイライトされた値をデータベースサーバーと適用することのできる値を置換え、データベースURL テキスト内にデータベースサーバーに接続文字列を入力します。

```
jdbc:teradata://databaseServerName
```

7. 「接続」をクリックします。

19.1.10.20 Teradata (ODBC) への接続

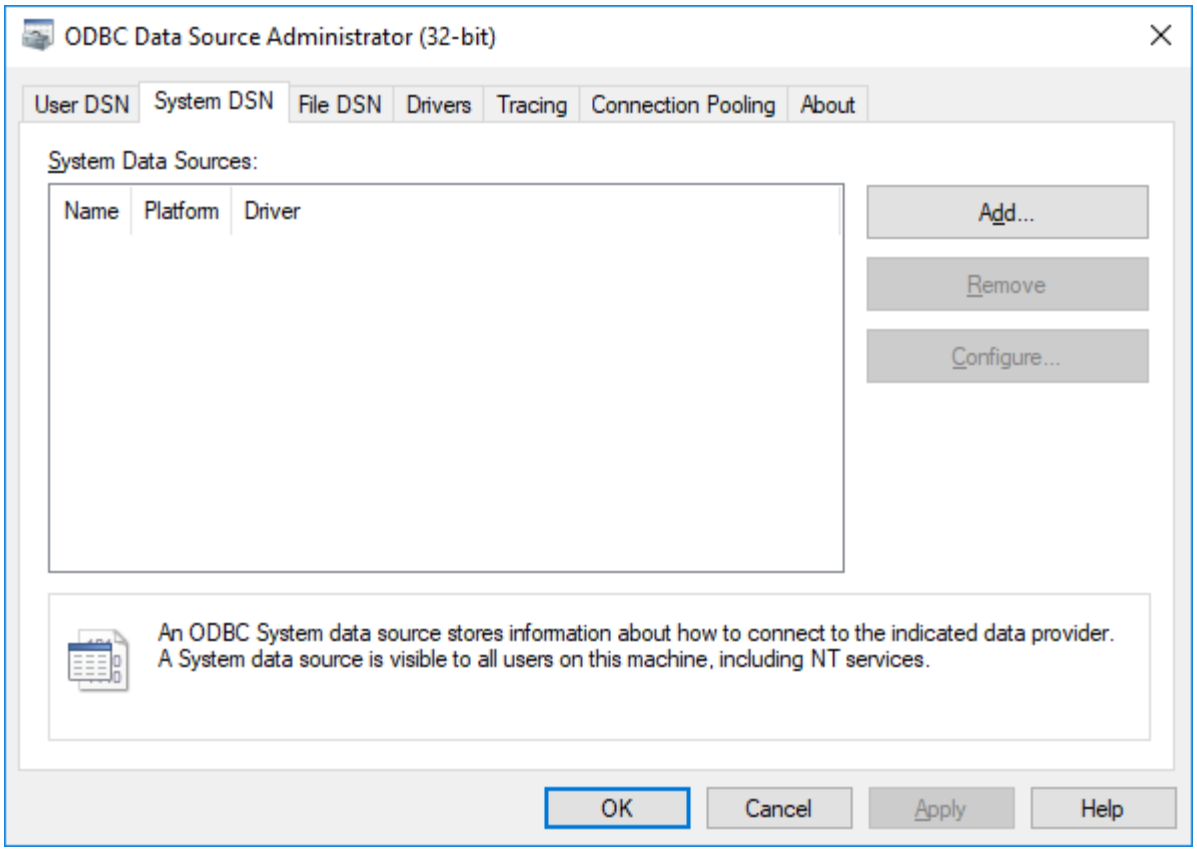
このサンプルでは、ODBC を使用したSQL Teradata データベースへの接続方法について説明されています。

前提条件:

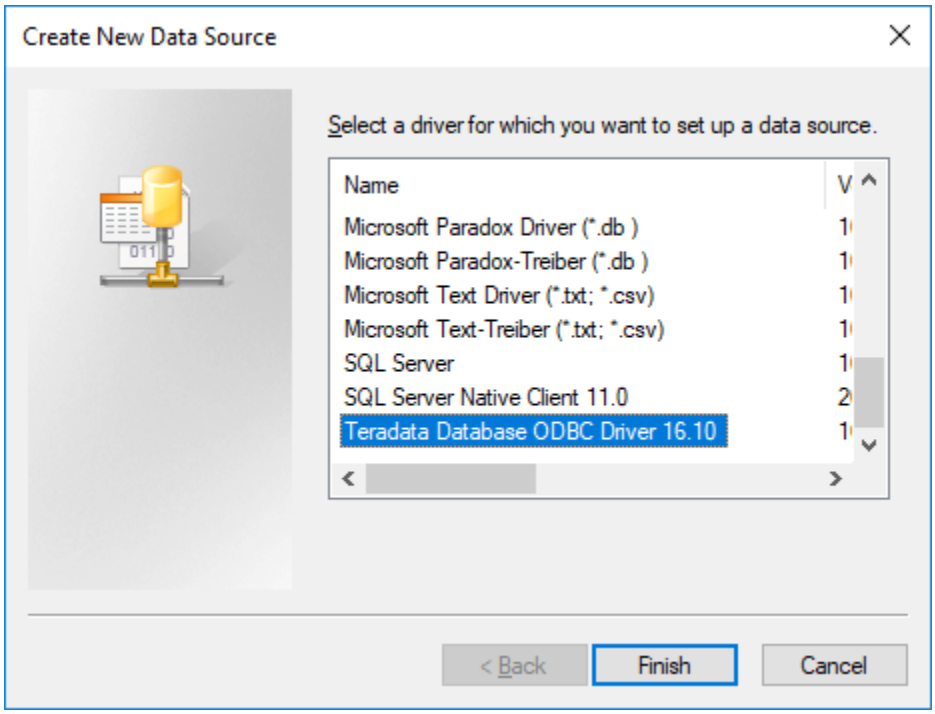
- Teradata ODBC ドライバーがインストールされる必要があります（参照: http://downloads.teradata.com/download/connectivity/odbc-driver/windows。このサンプルは、W indows バージョン16.20.00 のためにODBC ドライバーを使用しています。
- 次のデータベース接続詳細が存在します: ホスト、ユーザー名、および、パスワード。

ODBC を使用したTeradata への接続の方法:

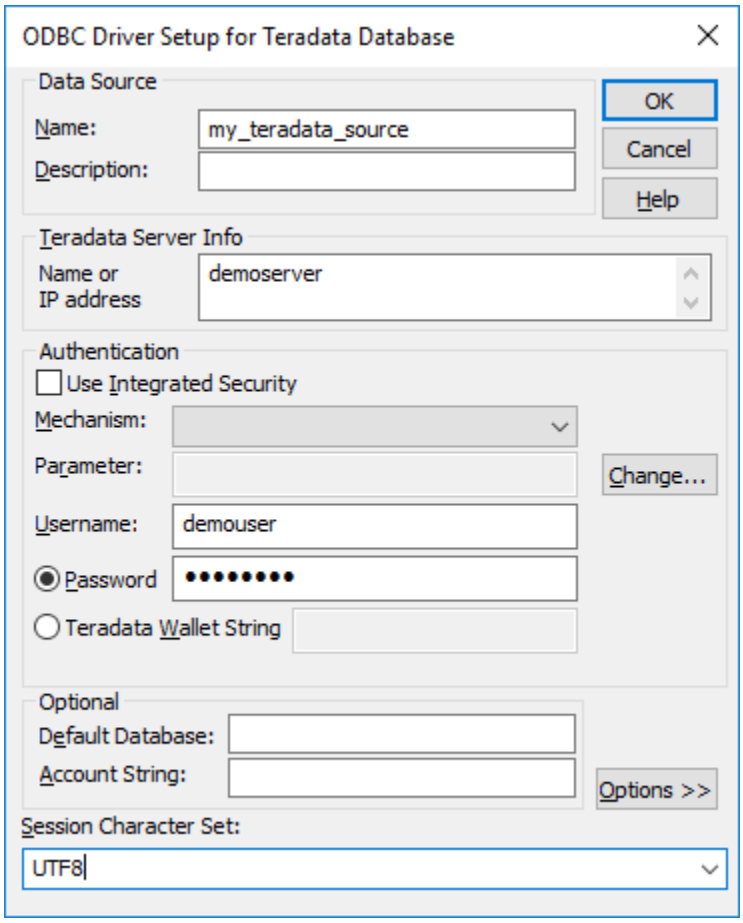
1. **Windows** キーを押して、「ODBC」を入力し、「 ODBC データソースのセットアップ（32-ビット）」を提案されるリストから選択します。64-ビット ODBC ドライバーが存在する場合、「 ODBC データソースのセットアップ（64-ビット）」を選択し、次のステップ内で64-ビット XMLSpy を使用します。



2. 「System DSN」 タブをクリックして、「追加」をクリックします。

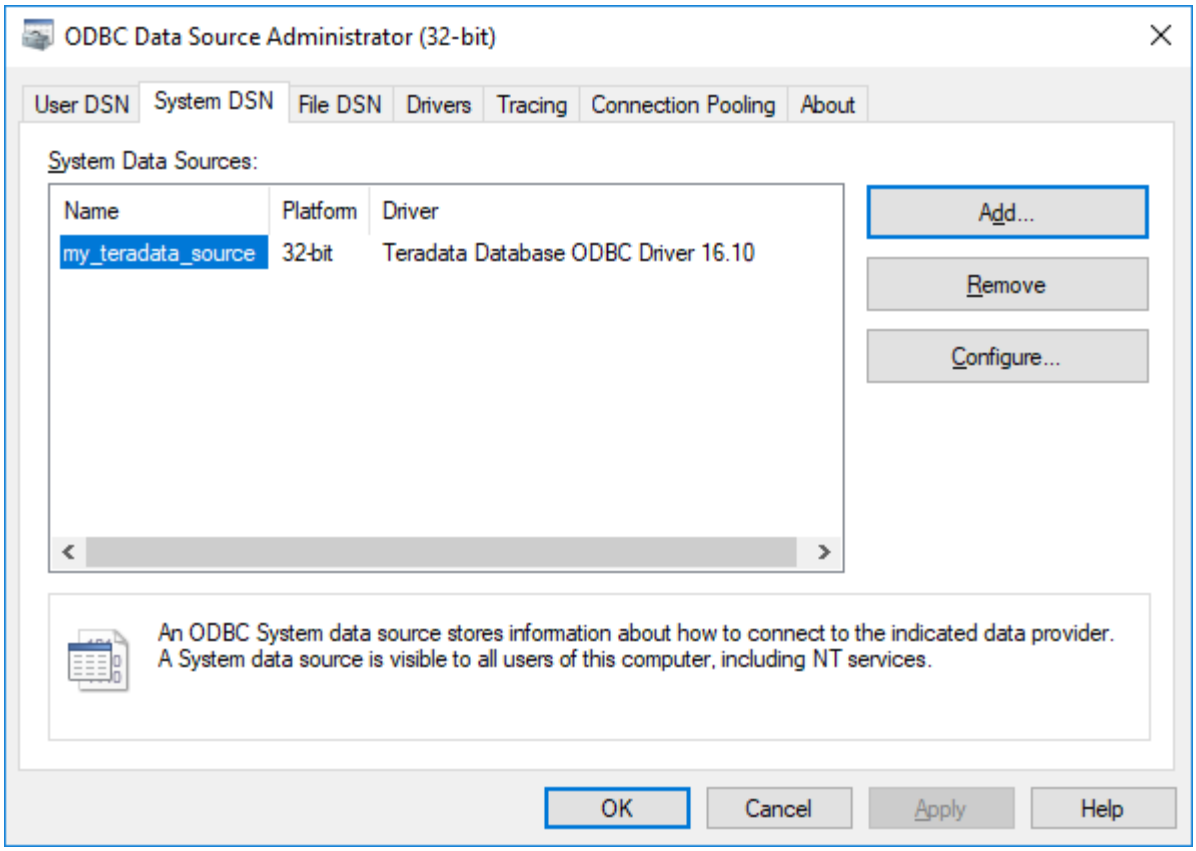


3. 「Teradata データベース ODBC ドライバー」を選択して、「終了」をクリックします。

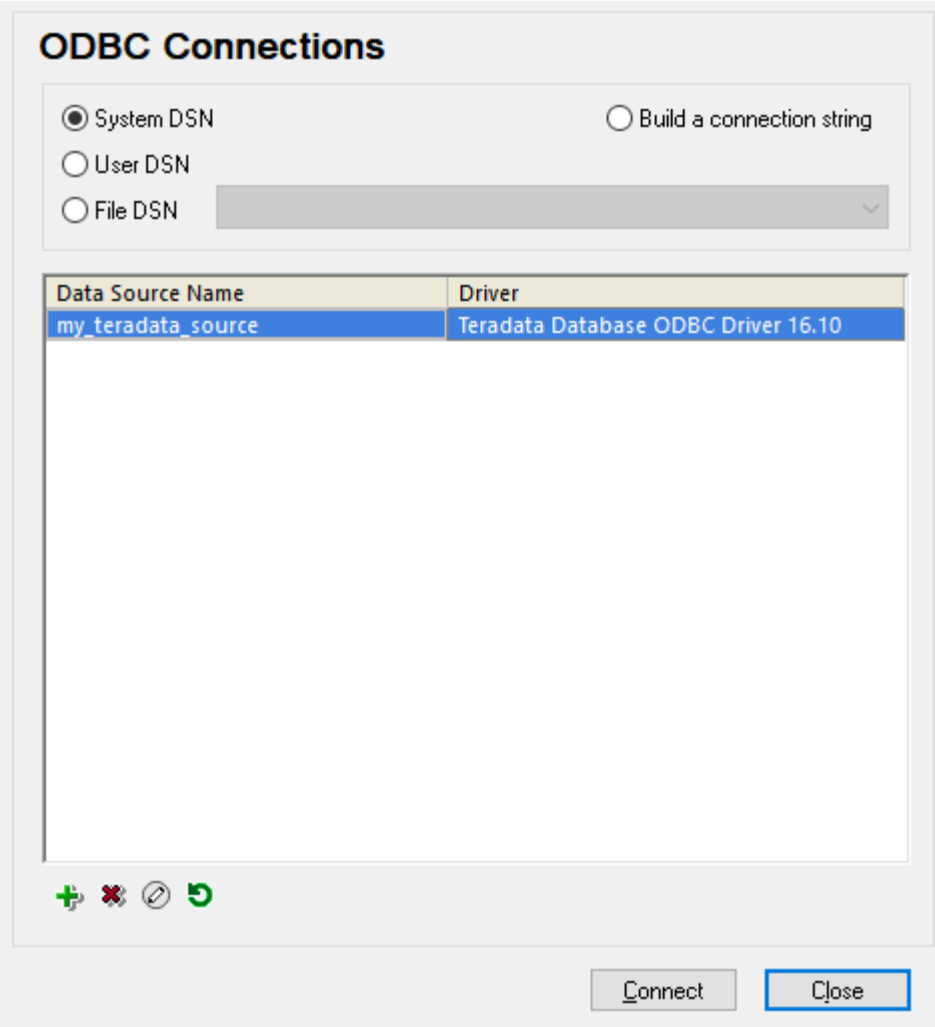


4. 名前を入力し、任意で、このODBC データソースを将来識別できる詳細を与えます。また、データベース接続資格情報（データベース サーバー、ユーザー、パスワード）を入力し、オプションでデータベースを選択します。
5. 「OK」をクリックします。データソースがリスト内に表示されます。

(C) 2015-2021 Altova GmbH



6. XMLSpy を作動し、データベース接続ウィザードを開始します[848]。
7. 「ODBC 接続」をクリックします。



8. 「**System DSN**」をクリックし、前に作成されているデータソースを選択し、「**接続**」をクリックします。

メモ 次のエラーが発生した場合: 「ドライバーが無効として返されました（または返しませんでした） SQL_ DRIVER_ ODBC_ VER: 03.80」「ドライバーが無効と返されました」。ODBC クライアントへのパスが使用中のシステムのPATH 環境変数内に存在することを確認してください（例えば、このロケーションにインストールされている場合、**C:\Program Files\Teradata\Client\16.10\bin** ）。このパスが不足している場合、手動で追加します。

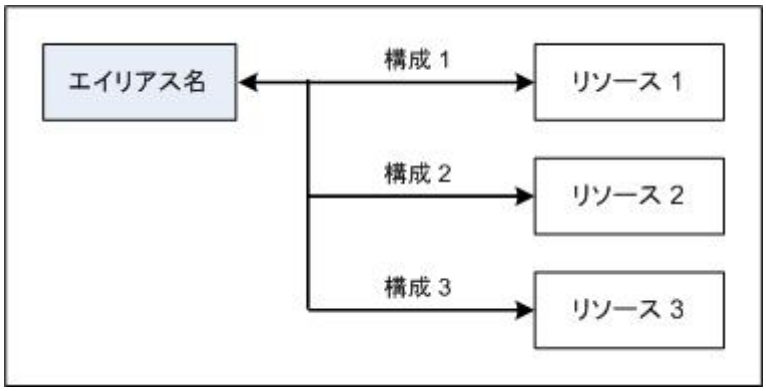
19.2 サポートされているデータベース

次のデータベースはサポートされています。各データベースの使用できるルート オブジェクトもリストされています。Altova は他の ODBC/ADO データベースへのサポートの向上のための努力をしていますが、以下にリストされたデータベースのみで接続とデータ処理のテストに成功しています。使用中の Altova アプリケーションが、64 ビット バージョンの場合、接続されているデータベース特定の 64 ビット データベース ドライバーにアクセスしてください。Altova アプリケーションは IBM iSeries データベースのロジカル ファイルをサポートし、ロジカル ファイルを表示します。

データベース	メモ
Firebird 2.5.4, 3.0	
IBM DB2 8.x, 9.1, 9.5, 9.7, 10.1, 10.5	
IBM Db2 for i 6.1, 7.1, 7.2, 7.3, 7.4	ロジカルファイルはサポートされ、ビューとして表示されています。
IBM Informix 11.70, 12.10, 14.10	
MariaDB 10.2, 10.3, 10.4, 10.5	
Microsoft Access 2003, 2007, 2010, 2013, 2016, 2019	ドキュメントの作成時（2019 年9 月初旬）Access 2019 のために使用可能な Microsoft Access Runtime が存在しませんでした。Microsoft Access 2016 Runtime がインストールされており、データベースが「大量」のデータ型を使用しない場合のみ Altova 製品から Access 2019 にデータベース接続することができます。
Microsoft Azure SQL Database	SQL Server 2016 codebase
Microsoft SQL Server 2005, 2008, 2012, 2014, 2016, 2017, 2019 Microsoft SQL Server on Linux	
MySQL 5.0, 5.1, 5.5, 5.6, 5.7, 8.0	
Oracle 9i, 10g, 11g, 12c, 18, 19	
PostgreSQL 8.0, 8.1, 8.2, 8.3, 9.0.10, 9.1.6, 9.2.1, 9.4, 9.5, 9.6, 10, 11, 12.1	PostgreSQL 接続は、ODBC または JDBC などのインターフェイス（ドライバ）によりネイティブとドライバーベースの接続でサポートされています。ネイティブな接続はドライバを必要としません。
Progress OpenEdge 11.6	
SQLite 3.33.0	SQLite 接続は、ネイティブとしてサポートされます。SQLite データベースファイルへの直接の接続です。 別途のドライバーは必要ありません。Authentic ビューでは、SQLite からのデータを編集することはできません。Authentic ビューで、SQLite からのデータを保存する場合は、既知の制限についてのメッセージが通知されます。
Sybase ASE 15, 16	
Teradata 16	

20 Altova グローバルリソース

Altova グローバルリソースはファイル、フォルダー、そしてデータベースリソースのエイリアス集合のことです。各エイリアスは複数の構成を持ち、各構成を1つのリソースにマッピングすることができます（下のスクリーンショット参照）。そのため、グローバルリソースを入力として使用することで、グローバルリソースを構成とともに切り替えることができます。この操作は、アクティブな構成を選択することのできるGUI 上で簡単に行うことができます。例えば、XML ドキュメントを変換するXSLT スタイルシートがグローバルリソース（エイリアス）により割り当てられている場合、このグローバルリソースに対して複数の構成を作成し、各構成が異なるXSLT ファイルを指すようにすることができます。このようにグローバルリソースを設定すると、構成を切り替えることで変換に使用されるXSLT ファイルを変更することができます。



グローバルリソースはAltova 製品内部におけるリソースの切り替えだけでなく、他のAltova 製品から生成、使用されるリソースに対しても使用することができます。そのため、1つのAltova 製品から他のAltova 製品で使用されるファイルを一括して生成することができます。このような機能により、開発ならびにテストスピードが飛躍的に向上します。例えば、XMLSpy にあるXSLT スタイルシートを使って、Altova MapForce で生成されたファイルを変換することができます。

Altova グローバルリソースの使用には、2 つのステップがあります:

- グローバルリソースの定義 935: XML ファイルによりリソースが定義されます。これらのリソースは複数のAltova 製品間で共有することができます。
- グローバルリソースの使用 944: XMLSpy 内部ではファイルパスの代わりにグローバルリソースによりファイルを指定することができます。グローバルリソースを使用すると、XMLSpy にてアクティブな構成を切り替えることで、使用するリソースを簡単に変更することができるようになります。

その他のAltova 製品におけるグローバルリソース

現在のところ、以下のAltova 製品でグローバルリソースを使用することができます: XMLSpy、StyleVision、MapForce、Authentic Desktop、MobileTogether Designer およびDatabaseSpy。

(C) 2015-2021 Altova GmbH

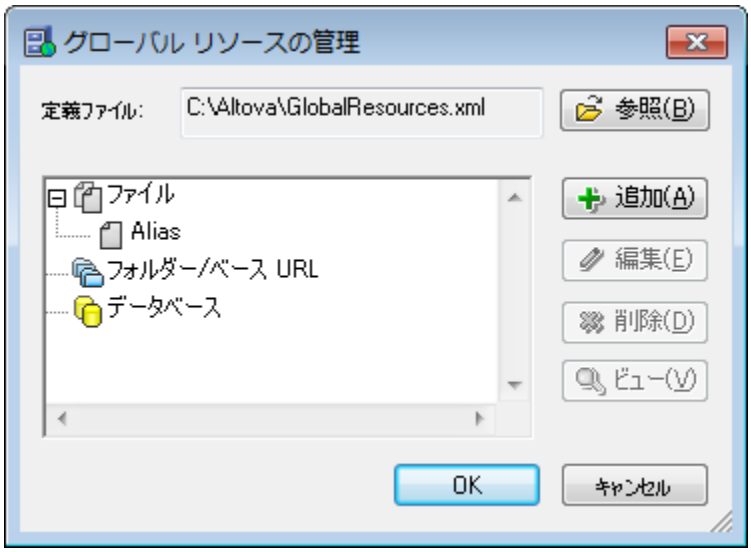
20.1　グローバル リソースの定義

Altova グローバルリソースは、以下の方法によりアクセスすることのできるグローバルリソースの管理ダイアログで定義されます:

- メニューコマンドの「ツール | グローバルリソース」を選択する。
- グローバルリソース ツールバーにある「グローバルリソースの管理」アイコンをクリックする（下のスクリーンショット）。

グローバルリソース定義ファイル

グローバルリソースに関する情報はグローバルリソース定義ファイルと呼ばれるXML ファイルに収められています。このファイルは、グローバルリソースの管理ダイアログにて初めてグローバルリソースが定義ならびに保存された際に作成されます（下のスクリーンショットを参照）。



グローバルリソースの管理ダイアログを初めて開くと、グローバルリソース定義ファイルのデフォルトの場所と名前が、定義ファイルテキスト ボックスにて指定されます（上のスクリーンショットを参照）:

```
C:\Users\<username>\My Documents\Altova\GlobalResources.xml
```

このファイルは、すべてのAltova アプリケーションに対するデフォルトのグローバルリソース定義ファイルとしてセットされます。従って、任意のAltova アプリケーションからこのファイルに対して保存されたグローバルリソースは、その他すべてのAltova アプリケーションにて直ちにグローバルリソースとして使用することができることになります。グローバルリソースを定義して、グローバルリソース定義ファイルに保存するには、グローバルリソースの管理ダイアログにてグローバルリソースを追加し、「OK」ボタンをクリックしてください。

既に存在するグローバルリソース定義ファイルを、実行中のAltova アプリケーションにおけるアクティブな定義ファイルとして選択するには、定義ファイルテキスト ボックスの隣にある「参照」ボタンから、そのファイルを選択してください。（上のスクリーンショットを参照）

メモ:　グローバルリソース定義ファイルの名前は自由に変更することができ、Altova アプリケーションからアクセスできる任意の場所へ配置することができます。この場合、各アプリケーションにて（定義ファイルテキスト ボックスから）このファイルをアプリケーションのグローバルリソース定義ファイルとして指定する必要があります。複数のAltova 製品にて同一の定義ファイルを指定することで、これらのAltova 製品にてリソースが利用できるようになります。

メモ: 複数のグローバルリソース定義ファイルを作成することもできます。この場合、各 Altova アプリケーションにて同時に選択できるアクティブな定義ファイルは 1 つだけで、そのファイルの中に含まれている定義だけがアプリケーションでは利用できるということに注意してください。そのため、必要に応じて同じリソースを複数の定義ファイルで記述する必要が生じることもあります。

グローバルリソースの管理: 追加、編集、削除、保存

グローバルリソースの管理ダイアログでは、選択されたグローバルリソース定義ファイルへグローバルリソースを追加することができるほか、選択されたグローバルリソースを編集または削除することができます。ファイル、フォルダー、そしてデータベースのグループに追加されたグローバルリソースは、グローバルリソース定義ファイルにより管理されます(上のスクリーンショットを参照)。

グローバルリソースを追加するには、「追加」ボタンをクリックして、表示される「**グローバルリソース**」ダイアログにてグローバルリソースを定義します(このセクション以下のサブセクションにある「ファイル」937、「フォルダー」941、「データベース」935 の記述を参照ください)。グローバルリソースを定義して(グローバルリソースの管理ダイアログにある「OK」ボタンをクリックすることで) 保存すると、選択されたグローバルリソース定義ファイルのグローバルリソース定義ライブラリに、そのグローバルリソースが追加されます。グローバルリソースはエイリアスにより識別されます。

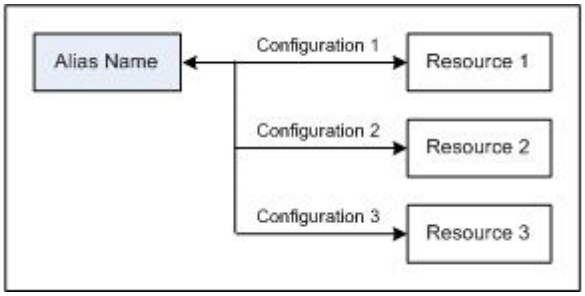
グローバルリソースを編集するには、「**編集**」ボタンをクリックします。関連するグローバルリソース ダイアログが表示され、必要な変更を行うことができます(「ファイル」937、「フォルダー」941、「データベース」935 サブセクションの記述を参照ください)。

グローバルリソースを削除するには、「**削除**」ボタンをクリックします。

グローバルリソースの追加、編集、または削除を終えた後には、グローバルリソースの管理ダイアログにある「OK」ボタンをクリックして、グローバルリソース定義ファイルに**変更点を保存**するようにしてください。

構成によりグローバルリソースをエイリアス名へ関連付ける

グローバルリソースを定義すると、エイリアス名をリソース(ファイル、フォルダー、データベース) へマッピングすることになります。1 つのエイリアス名から複数のリソースへのマッピングを行うことができ、各マッピングは構成と呼ばれます。そのため、単一のエイリアス名から、複数の構成を介して複数のリソースへの関連付けを行うことができます(以下の図を参照)。



Altova アプリケーションではファイルの代わりにエイリアスを割り当てることができます。アプリケーションのアクティブなグローバルリソース構成を切り替えることで、各エイリアスに割り当てられたリソースを切り替えることができます。例えば Altova XMLSpy アプリケーションにて、MyXML.xml という名前の XML ドキュメントに対して XSLT 変換を行う場合、MyXSLT という名前のエイリアスを使用される XSLT としてグローバルリソースに割り当てることができます。XMLSpy では、アクティブな構成を変更することで、異なる XSLT ファイルを使用することができます。構成-1 にて First.xslt が MyXSLT へマッピングされており、構成-1 がアクティブな構成として選択されている場合、First.xslt が変換に使用されます。これにより、複数の構成を使用することで、単一のエイリアスから複数のリソースへアクセスすることができるようになります。このメカニズムはリソースのテストや比較を行う際に便利な機能となります。更に、グローバルリソースは複数の Altova 製品間で使用することができるため、複数の Altova 製品にてリソースのテストや比較を行うことができます。

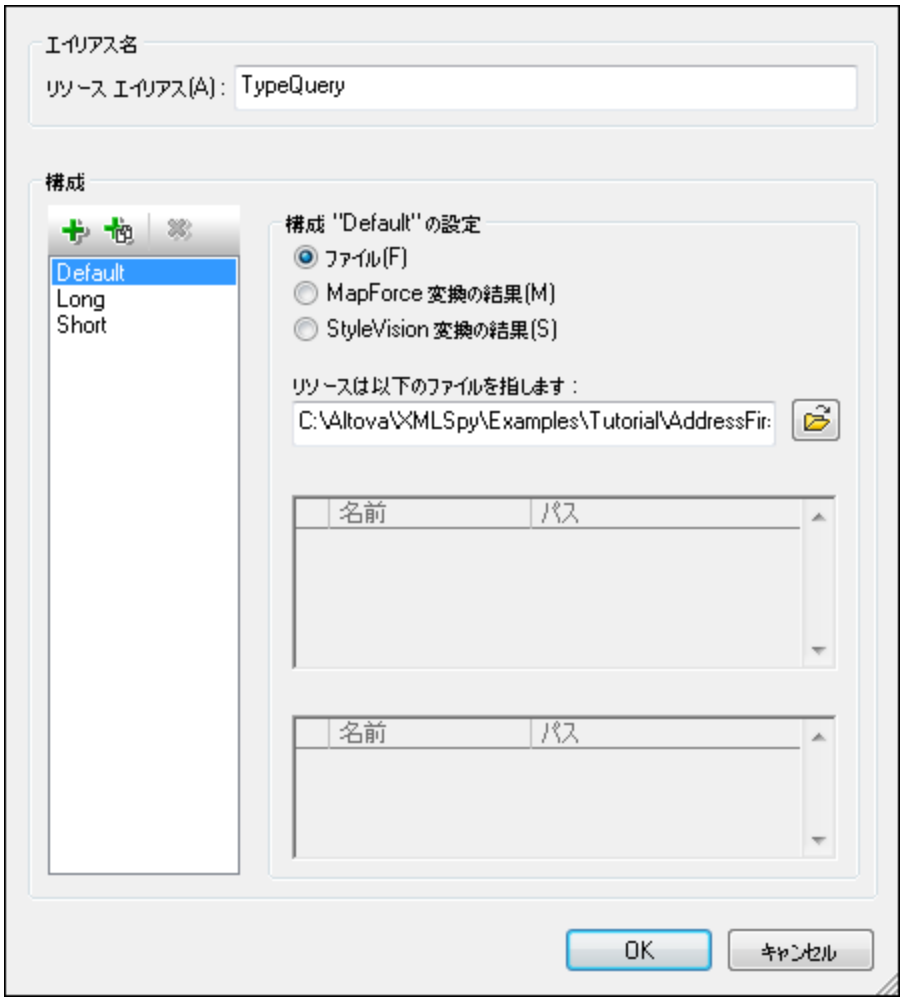
20.1.1　ファイル

「グローバルリソースの管理」935ダイアログにある「追加 | ファイル」コマンドにより、ファイルのグローバルリソースダイアログへアクセスすることができます（以下のスクリーンショットを参照）。グローバルリソースダイアログでは、リソースエイリアステキストボックスにて入力されたエイリアスの構成を定義することができます。以下に記される方法により構成のプロパティを指定して、「OK」をクリックしてエイリアスの定義を保存してください。

エイリアスの定義を保存した後には上記の操作を（「グローバルリソースの管理」935ダイアログにある「追加 | ファイル」コマンドを選択するところから）再度行うことで、別のエイリアスを追加することができます。

グローバルリソースダイアログ

エイリアスはグローバルリソースダイアログにて定義されます（以下のスクリーンショットを参照）。



グローバルリソースダイアログのアイコン

構成の追加: 追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

構成をコピーして追加: 選択された構成のコピーとして追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

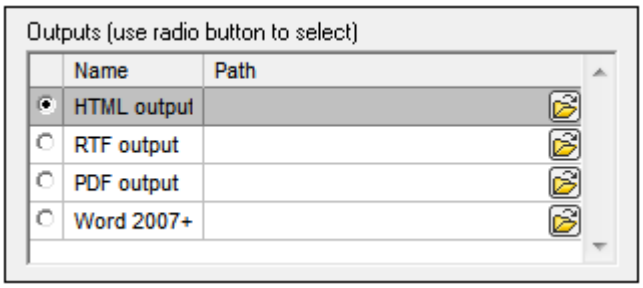
削除: 選択された構成を削除します。

開く: グローバルリソースとして作成されるファイルを参照します。

エイリアスの定義

以下の操作により、エイリアスの名前と構成を定義することができます:

1. *エイリアスに名前を付ける*: リソース エイリアス テキスト ボックスにエイリアスの名前を入力します。
2. *構成を追加する*: 構成ペインには、デフォルトで(名前変更や削除を行うことができない) Default という名前の構成が表示されます。(i) 構成の追加または構成のコピーを追加アイコンをクリックして、(ii) 表示されるダイアログにて構成名を入力することで、必要なだけの構成を追加することができます。追加された各構成は構成リストに表示されます。上のスクリーンショットでは、Long と Short という名前の構成が構成リストに追加されています。構成のコピーを追加コマンドにより、選択されている構成をコピーして、修正することもできます。
3. *各構成に対してリソースの種類を選択する*: 構成のリストから目的の構成を選択し、構成の設定ペインにて、(i) ファイル、(ii) MapForce 変換の結果、または(iii) StyleVision 変換の結果から、構成のリソースを指定します。対応するラジオ ボタンを選択してください。MapForce または StyleVision 変換の結果が選択された場合、.mfd または .sps ファイルと対応する入力ファイルを使って MapForce または StyleVision により変換が行われ、その変換結果がリソースとなります。
4. *リソースの種類に対するファイルを選択する*: リソースの種類にファイルが選択された場合、リソースのファイル テキスト ボックスにてファイルを選択してください。リソースの種類に変換の結果が選択された場合、ファイルの選択テキスト ボックスにて(MapForce 変換の場合は) .mfd ファイルまたは(StyleVision 変換の場合は) .sps ファイルを選択してください。変換に対して複数の入力または出力が可能な場合、選択するためのオプションが表示されます。例えば、StyleVision 変換の出力オプションが、インストールされている StyleVision のエディションに合わせて表示されます(以下のスクリーンショットでは、Enterprise Edition の出力が示されています)。



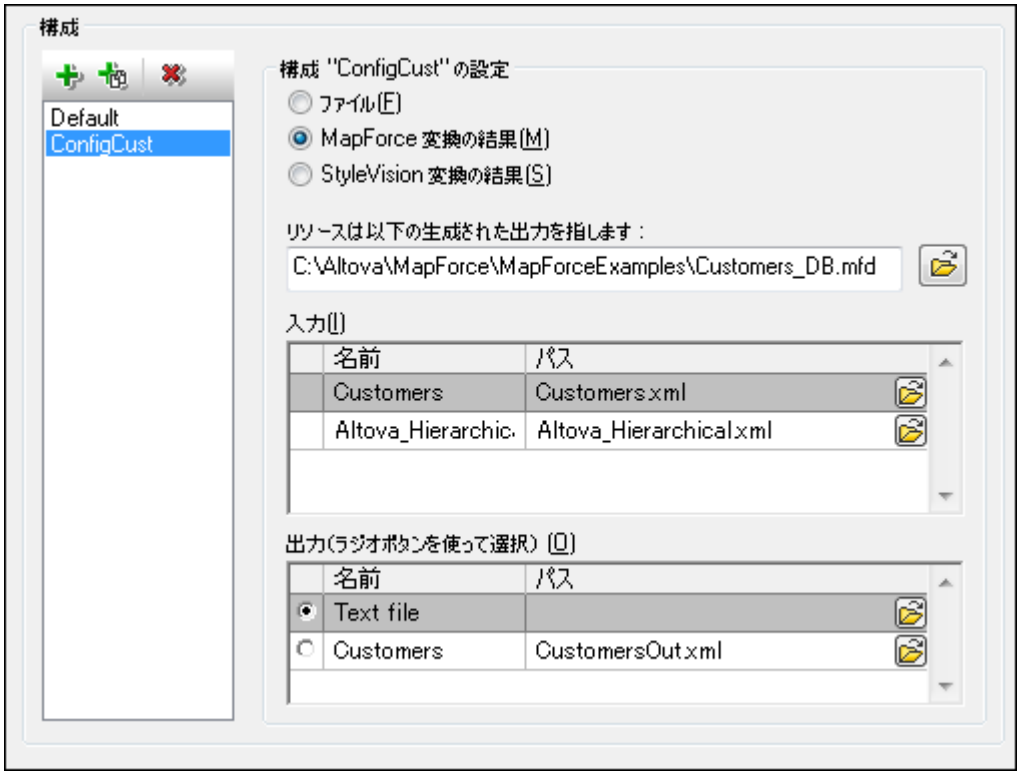
目的のオプションに対応するラジオ ボタンを選択します(上のスクリーンショットでは、「HTML 出力」が選択されています)。変換の結果をリソースとする場合、出力をファイルまたはグローバルリソースとして保存することができます。 アイコンをクリックして、「他のグローバルリソースを選択」または、「参照」を選択してください。どちらのオプションも選択されていない場合、グローバルリソースが使用された際に変換の結果が一時ファイルとしてロードされます。

5. *必要に応じて複数の構成を定義する*: その他の構成を必要な数だけ追加して、各構成に対してリソースを指定します。各構成に対して上記のステップ 3 と 4 を実行してください。エイリアスに対する新たな構成はいつでも追加することができます。
6. *エイリアスの定義を保存する*: 「OK」をクリックしてエイリアスと、すべての構成をグローバルリソースとして保存します。グローバルリソースが、「グローバルリソースの管理」ダイアログ 935 にあるファイル以下に表示されます。

MapForce 変換の結果

Altova MapForce では、(既存の) 入力ドキュメント スキーマから出力ドキュメント スキーマへのマッピングが行われます。MapForce ユーザーにより作成されたこのマッピングは、MapForce Design (MFD) と呼ばれます。入力スキーマに関連付けられた XML ファイル、テキスト ファイル、データベースは、データ ソースとして使用することができ、MapForce により出力ドキュメント スキーマに対応した出力データ ファイルが生成されます。「MapForce 変換の結果」が出力ドキュメントで、グローバルリソースになります。

MapForce により生成されたデータ ファイルをグローバルリソースとしてセットするには、グローバルリソース ダイアログにて、以下の設定を行う必要があります (以下のスクリーンショットを参照)。



- **.mfd (MapForce Design) ファイル**: 「リソースは以下の生成された出力を指します」テキスト ボックスにて、.mfd ファイルを指定する必要があります。
- **入力データ ファイル**: MFD ファイルが指定されると、そのファイルが解析され、中に含まれる入力スキーマ情報をベースにデフォルトのデータ ファイルが入力ペインに入力されます (上のスクリーンショットを参照)。他のファイルを指定することで、各入力スキーマに対するデフォルト ファイルの選択を変更することができます。
- **出力ファイル**: MFD ドキュメントに複数の出力スキーマが含まれる場合、出力ペインにそれらの出力すべてが表示されます (上のスクリーンショットを参照)。個々の出力スキーマに関連付けられた出力ファイルの場所が MFD ドキュメントにて指定されている場合、その出力スキーマに対するファイルの位置が出力ペインに表示されます。上のスクリーンショットからは、Customers 出力スキーマにデフォルトの XML データ ファイル (CustomersOut.xml) が与えられており、Text ファイル出力 スキーマにはそのような割り当てが行われていないことが理解できます。出力ペインに表示あるデフォルトのファイルの場所を使用することができるほか、自分で指定することができます。MapForce 変換の結果が、選択された出力スキーマがある場所へ保存され、そのファイルがグローバルリソースとして使用されます。

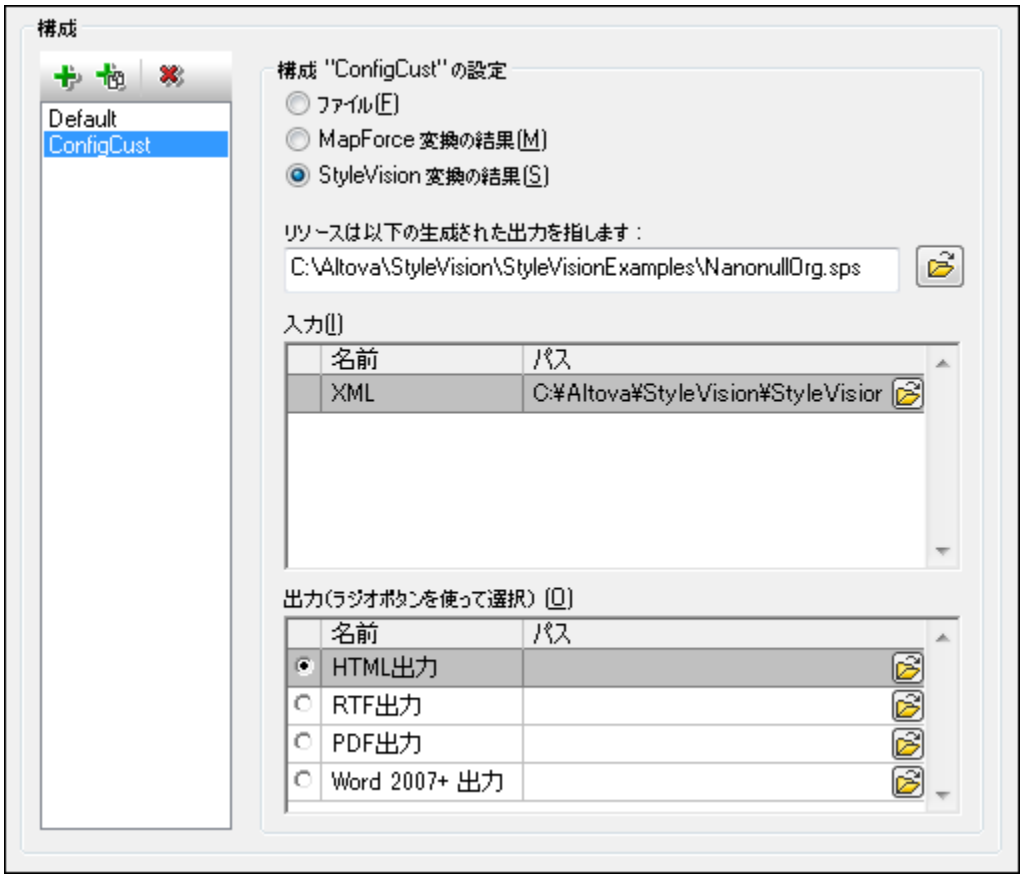
メモ　このオプション (MapForce の結果) により、グローバルリソースが呼び出された時に変換が行われるようになります。これで (入力ファイルから得られた) 最新のデータがグローバルリソースに含まれることが保証されます。

メモ 変換を行うためにはMapForce が使用されるため、この機能を使用するにはAltova MapForce をインストールする必要があります。

StyleVision 変換の結果

Altova StyleVision はStyleVision Power Stylesheet（SPS）ファイルを作成するために使用され、これらSPS ファイルにより、XML ドキュメントを様々なフォーマット（HTML、PDF、RTF、W ord 2007+ など）の出力ドキュメントへ変換するためのXSLT スタイルシートが生成されます。「StyleVision 変換の結果」オプションを選択すると、StyleVision により作成された出力ドキュメントが、選択された構成に関連付けられたグローバルリソースとなります。

グローバルリソースダイアログにて「StyleVision 変換の結果」オプションを選択すると、以下のファイルを指定する必要があります。



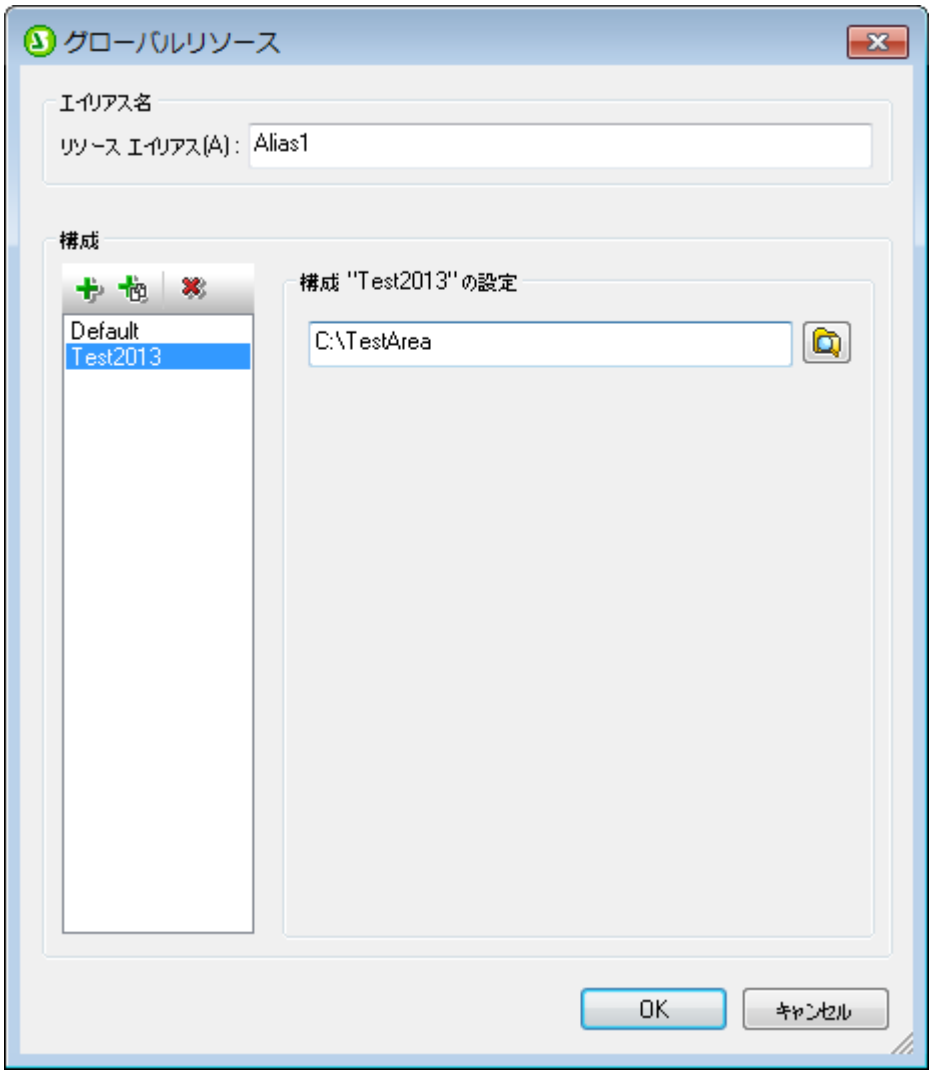
- *.sps*（SPS）ファイル: 「リソースは以下の生成された出力を指します」テキスト ボックスにて、ファイルを指定する必要があります（上のスクリーンショットを参照）。
- 入力ファイル: 入力ファイルがSPS ファイル内部で既に指定されている場合、SPS ファイルが選択された段階で、そのファイルが入力ペインに表示されます。このエントリーは変更することができるほか、エントリーが指定されていない場合、追加する必要があります。
- 出力ファイル: 出力ペインにて出力フォーマットを選択し、そのフォーマットに対する出力ファイルの場所を指定します。

メモ このオプション（StyleVision の結果）により、グローバルリソースが呼び出された時に変換が行われるようになります。これで、（入力ファイルから得られた）最新のデータがグローバルリソースに含まれることが保証されます。

メモ 変換を行うためにはStyleVision が使用されるため、この機能を使用するにはAltova StyleVision をインストールする必要があります。

20.1.2　フォルダー

フォルダーに対するグローバルリソース ダイアログでは、以下の操作によりフォルダーリソースを追加することができます（以下のスクリーンショットを参照）。



グローバルリソース ダイアログのアイコン

構成の追加: 追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

構成をコピーして追加: 選択された構成のコピーとして追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

削除: 選択された構成を削除します。

開く: グローバルリソースとして作成されるファイルを参照します。

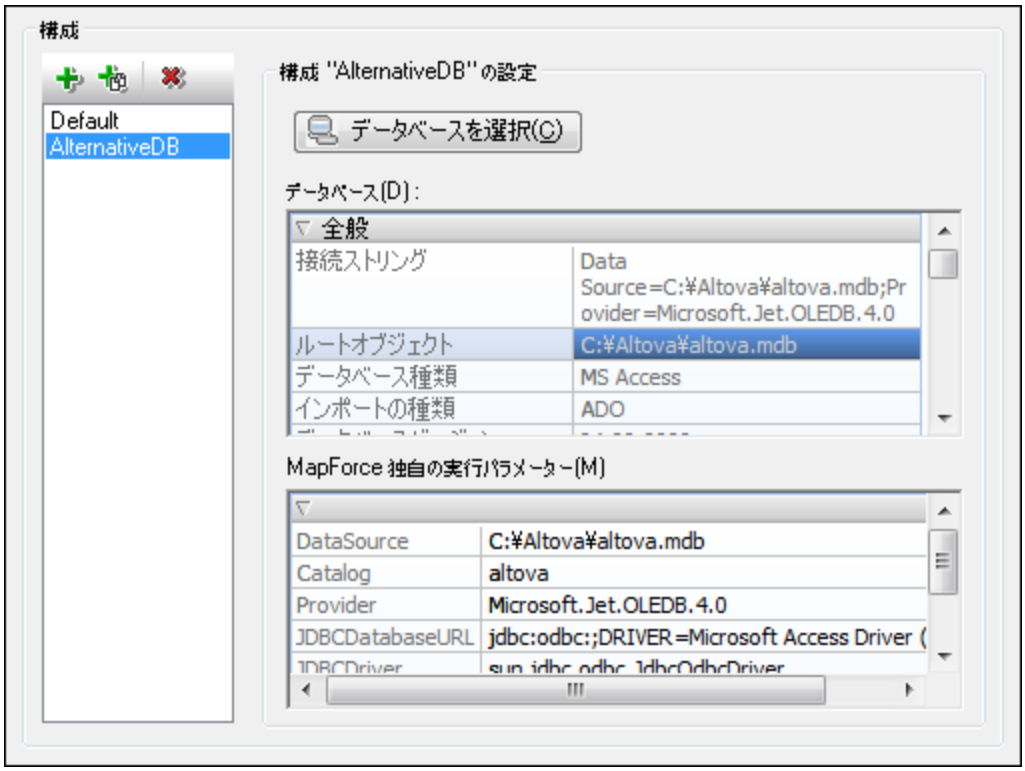
エイリアスの定義

以下の操作によりエイリアス(名前と構成) を定義することができます:

1. **エイリアスに名前を付ける:** リソース エイリアス テキスト ボックスにエイリアスの名前を入力します。
2. **構成を追加する:** 構成ペインには、デフォルトで(名前変更や削除を行うことができない) Default という名前の構成が表示されます。**構成の追加**または**構成のコピー**を追加アイコンをクリックして、表示されるダイアログにて構成名を入力することで、必要なだけの構成を追加することができます。構成の追加または構成のコピーを追加アイコンをクリックして、表示されるダイアログで構成の名前を入力することで、構成が追加されます。「OK」をクリックすることで新たな構成が構成ペインに表示されます。同じ作業を必要な数だけ繰り返してください。
3. **構成のソースとなるフォルダーを選択する:** 構成ペインに表示された構成を選択し、グローバルリソースとして作成するフォルダーを参照します。セキュリティ資格情報がフォルダーにアクセスするために必要とされる場合、ユーザー名とパスワードフィールドでこれらを指定してください。フォルダーにアクセスするためにセキュリティの資格条件が必要な場合、ユーザー名とパスワードフィールドを指定してください。
4. **必要に応じて複数の構成を定義する:** その他の構成を必要な数だけ追加して、各構成に対してリソースを指定します。各構成に対して上記のステップ 2 と3 を実行してください。エイリアスに対する新たな構成はいつでも追加することができます。
5. **エイリアスの定義を保存する:** 「OK」をクリックしてエイリアスと、すべての構成をグローバルリソースとして保存します。グローバルリソースが、「グローバルリソースの管理」ダイアログ[935]にあるファイル以下に表示されます。

20.1.3 データベース

データベースに対するグローバルリソース ダイアログでは、以下の操作によりデータベースのソースを追加することができます:



グローバル リソース ダイアログのアイコン

　構成の追加: 追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

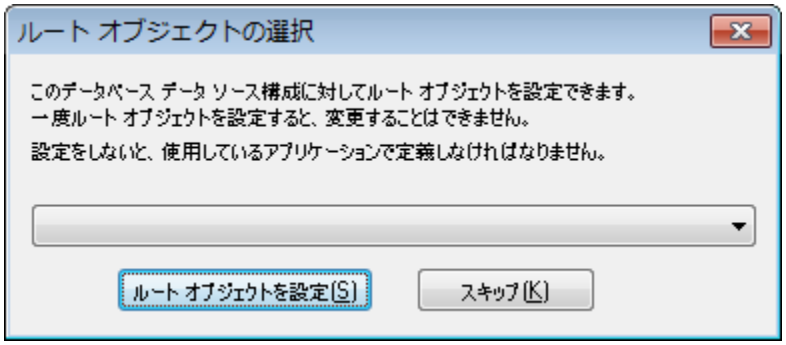
　構成をコピーして追加: 選択された構成のコピーとして追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

　削除: 選択された構成を削除します。

エイリアスの定義

以下の操作によりエイリアス(名前と構成) を定義することができます:

1. *エイリアスに名前を付ける*: リソース エイリアス テキスト ボックスにエイリアスの名前を入力します。
2. *構成を追加する*: 構成ペインには、デフォルトで(名前変更や削除を行うことができない) Default という名前の構成が表示されます。構成の追加または構成のコピーを追加アイコンをクリックして、表示されるダイアログにて構成名を入力することで、必要なだけの構成を追加することができます。構成の追加または構成のコピーを追加アイコンをクリックして、表示されるダイアログで構成の名前を入力することで、構成が追加されます。「OK」をクリックすることで新たな構成が構成ペインに表示されます。同じ作業を必要な数だけ繰り返してください。
3. *構成のリソースとなるデータベースを選択する*: 構成ペインに表示された構成を選択し、「**データベースを選択**」ボタンをクリックします。グローバルリソース接続を作成ダイアログが表示されます。フォルダーにアクセスするためにセキュリティの資格条件が必要な場合、ユーザー名とパスワードフィールドを指定してください。
4. *データベースへ接続する*: 接続ウィザード、既存の接続、 ADO 接続、ODBC 接続、またはJDBC 接続を使用して、データベースへの接続を作成します。データベースへの接続[847]のセクションで説明されている接続メソッドの定義を完成します。接続ウィザード[848]、ADO 接続[852]、または、ODBC 接続[865]を使用することができます。 XMLSpy から既にデータベースへの接続が存在する場合は、既存の接続アイコンをクリックして、表示される接続リストからデータベースを選択します。
5. *ルート オブジェクトの選択*: ルート オブジェクトを選択することのできるデータベースへ接続した場合、サーバー上にあるルート オブジェクトを選択するためのダイアログが表示されます(以下のスクリーンショットを参照)。ルート オブジェクトを選択して、「**ルート オブジェクトを設定**」をクリックしてください。選択されたルート オブジェクトが、この構成が使用された時にロードされるルート オブジェクトとなります。



ルート オブジェクトを選択しなかった場合(「**スキップ**」ボタンをクリックした場合)、グローバルリソースがロードされた際にルート オブジェクトを選択することになります。

6. *複数の構成を必要に応じて定義する*: 作成されたその他の構成に対してデータベース リソースを指定します(作成された構成に対して上記のステップ 3 から 5 までを繰り返します)。新たな構成をエイリアス定義へ追加することもできます。
7. *エイリアスの定義の保存*: 「OK」をグローバルリソースダイアログ内でクリックし、エイリアスとその構成をグローバルリソースとして保存します。グローバルリソースは、グローバルリソースの管理ダイアログ内でデータベースの下にリストされています。

20.2 グローバルリソースを使用する

グローバルリソースには、ファイル型、フォルダー型、そしてデータベース型と、複数の種類が存在します。 XMLSpy で使用することのできる様々なシナリオについて以下で紹介されています: ファイルとフォルダー[944] および データベース[947]。

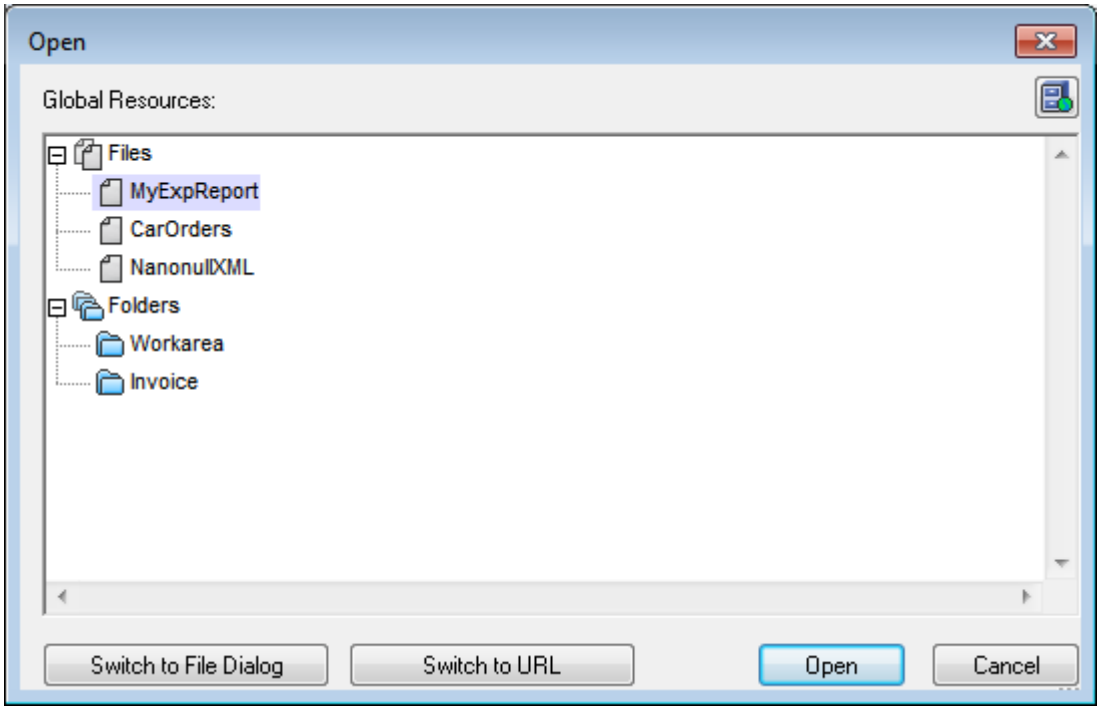
使用するリソースの選択

アプリケーション全体で使用することのできるグローバルリソースの選択、ならびに実際に使用されるグローバルリソースの選択を行うには方法が2つあります:

- 「グローバルリソース」ダイアログ[935]にてアクティブなグローバルリソース XML ファイルを選択します。アクティブなグローバルリソース XML ファイル内に収められている利用可能なグローバルリソースの定義は、アプリケーション内の開かれているファイル全てで使用することができます。使用することができるのは、アクティブなグローバルリソース XML ファイルにある定義だけとなります。アクティブなグローバルリソース XML ファイルはいつでも変更することができ、新しいファイルに収められたグローバルリソース定義が、それまで使用されていたアクティブなファイルの内容に置き換えられます。従って、アクティブなグローバルリソース XML ファイルにより、(i) 割り当て可能なグローバルリソースと、(ii) 検索可能なグローバルリソースが決定されます(例えば、あるグローバルリソース XML ファイル内のグローバルリソースに割り当てが行われているが、現在アクティブなグローバルリソース XML ファイルにて、対応する名前のリソースがない場合、割り当てられたグローバルリソースの検索を行うことはできなくなります)。
- 「ツール | アクティブな構成」[1434] または グローバルリソース ツールバーからアクティブな構成 を選択します。メニューコマンド(またはツールバーのドロップダウンリスト)をクリックすることで、全てのエイリアスに対する構成のリストが表示されます。構成を選択することで、その構成がアプリケーションでアクティブな構成となります。この操作により、グローバルリソース(またはエイリアス)が使用される際には、各エイリアスに対して対応するアクティブな構成が適用されます。アクティブな構成は使用される全てのエイリアスに対して適用されます。アクティブな構成と同じ名前の構成をエイリアスが持たない場合、そのエイリアスに対してはデフォルトの構成が適用されます アクティブな構成はリソースの割り当てに影響を与えません。アクティブな構成は、実際にリソースが使用されるときに重要になります。

20.2.1 ファイルとフォルダーの割り当て

このセクションでは、ファイル型ならびにフォルダー型のグローバルリソース割り当て方法について記述します。ファイル型とフォルダー型のグローバルリソースは異なる方法で割り当てられます。以下にある使用シナリオでは、「グローバルリソースに切り替え」ボタンにてグローバルリソースを開くダイアログが表示されます(以下のスクリーンショットを参照)。



グローバルリソースの管理: 「グローバルリソースの管理」[935]ダイアログが表示されます。

ファイル型のグローバルリソースを選択することで、ファイルが割り当てられます。フォルダー型のグローバルリソースを選択することで、ファイルを開くダイアログが表示され、目的のファイルを選択することができます。選択されたファイルへのパスは、フォルダーリソースからの相対パスとなります。フォルダー型のグローバルリソースに2つの構成が含まれ、各構成が異なるフォルダーを指す場合（そして、それらのフォルダー内にあるファイルの名前が同一の場合）、構成切り替えることで2つの異なるフォルダーをターゲットにすることができます。これは試験などを行う際に便利な機能となります。

ダイアログ下部にあるボタンをクリックすることで、ファイルダイアログまたはURL ダイアログに切り替えることもできます。「**グローバルリソースの管理**」アイコンをクリックすることで、「グローバルリソースの管理」[935]ダイアログが表示されます。

使用できるシナリオ

ファイル型やフォルダー型のグローバルリソースに対して以下のような操作を行うことができます:

- グローバルリソースを開く[945]
- グローバルリソースとして保存[946]
- XSLT 変換にファイルを割り当てる[946]
- XSLT 変換[947]
- XQuery の実行[947]
- SPS の割り当て[947]

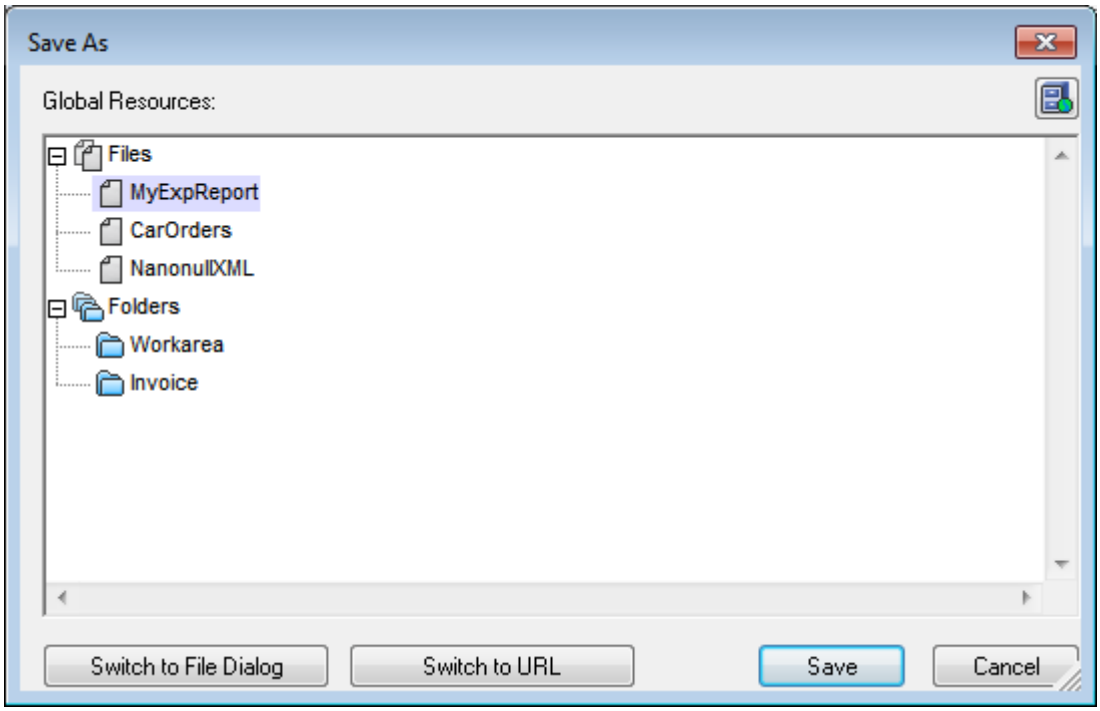
グローバルリソースを開く

グローバルリソースはXMLSpy の「ファイル | 開く（グローバルリソースに切り替え）|」[1132]コマンドから開き、編集することができます。ファイル型グローバルリソースの場合、ファイルが直接開かれます。フォルダー型グローバルリソースの場合、割り当てられたフォルダーが選択された状態でファイルを開くダイアログが表示され、フォルダー以下にある目的のファイルを選択することになります。グローバルリソースを通してファイ

ルを選択し、編集する利点は、構成を切り替えるだけで、同一のグローバルリソース内にある(別構成の)ファイルへアクセスできることです。構成を変更する前に、編集がなされたファイルを保存を行う必要があります。

グローバルリソースとして保存

新たに作成されたファイルもグローバルリソースとして保存することができます。同様に、既存のファイルを開いて、グローバルリソースとして保存することもできます。「ファイル | 保存」または「ファイル | 名前を付けて保存」コマンドをクリックすると、保存ダイアログが表示されます。「グローバルリソースに切り替え」ボタンをクリックして、現在使用されているグローバルリソース XML ファイルにて定義されたエイリアスであるグローバルリソースへアクセスします(以下のスクリーンショットを参照)。



エイリアスを選択し、「保存」をクリックします。エイリアスがファイルエイリアス[937]の場合、そのファイルに直接保存されます。エイリアスがフォルダーエイリアス[941]の場合、ダイアログが表示され、保存するファイルの名前を入力するよう促されます。どちらの場合でも、ファイルは現在アクティブな構成[948]により定義された位置に保存されます。

メモ 各構成ではファイルの位置が構成の定義により指定されます。グローバルリソースとして保存しようとするファイルが、構成で定義された位置にあるファイルと同一の拡張子を持たない場合、このグローバルリソースが開かれたときに、XMLSpy にて編集エラーや検証エラーが発生する可能性があります。これは構成の定義により指定されたファイルの種類を想定して XMLSpy によりファイルが開かれるためです。

XSLT 変換にファイルを割り当てる

グローバルリソースを介して XSLT ファイルを XML ドキュメントに、XML ファイルを XSLT ドキュメントに割り当てることができます。XSLT ファイルを割り当てるコマンド「XSL/XQuery | XSL の割り当て」[1287]ならびに「XSL/XQuery | XSL:FO の割り当て」[1288]や XML ファイルを割り当てるコマンド「XSL/XQuery | サンプル XML の割り当て」[1288]がクリックされると、割り当てダイアログが表示されます。「参照」ボタンをクリックすることで、ファイルを開くダイアログが表示され、グローバルリソースを開くダイアログに切り替え、目的のグローバルリソースを選択することができます。グローバルリソースを使用することで、XSLT 変換を行うために指定した XSLT(または XML)ファイルを、XMLSpy の構成を切り替えるだけで変更することができます。異なるファイルに対して変換が必要になった時も、ファイルの割り当て変更を行う必要はありません。XSLT 変換が開始されると、アクティブな構成で指定されるファイルが使用されます。

XSLT 変換 ならびにXQuery の実行

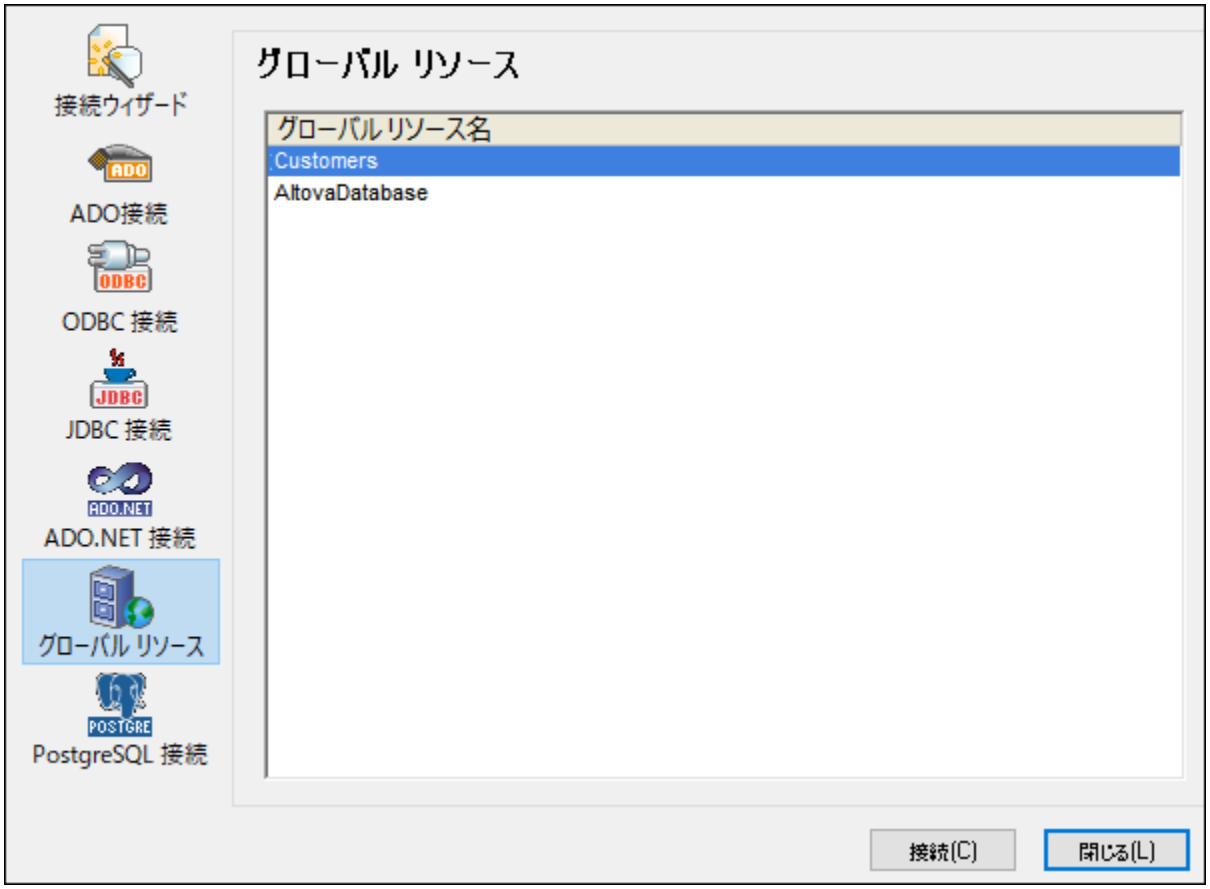
コマンド「XSL/XQuery | XSL 変換」[1279] または「XSL/XQuery | XSL:FO 変換」[1280] または「XSL/XQuery | XQuery 実行」[1285] をクリックすると、XQuery または XML ファイルを指定するためのダイアログが表示されるので、、XSLT、XQuery、または XML ファイルを選択します。「参照」ボタンをクリックして、「グローバルリソースに切り替え」ボタンをクリックすることで、グローバルリソースを開くダイアログが表示されます(セクション最上部のスクリーンショットを参照[944])。選択されたグローバルリソースの現在アクティブな構成に関連付けられているファイルが変換に使用されます。

SPS の割り当て

StyleVision スタイルシートを XML ファイルに割り当てる際「Authentic | SPS ファイルの割り当て」、スタイルシートを指定するためにグローバルリソースを使用することができます。「参照」ボタンをクリックして、グローバルリソースリソースに切り替えボタンをクリックすることで、グローバルリソース開くダイアログが表示されます(セクション最上部のスクリーンショットを参照[944])。グローバルリソースを割当先として選択することで、XMLSpy のアクティブな構成を切り替えるだけで、XML ドキュメントの Authentic View を変更することができます。

20.2.2　データベースの割り当て

DB から XMLSpy へ、データや(XML スキーマとして)データ構造をインポートするコマンドを実行すると(例えば、「変換 | データベースのデータをインポート」コマンドを使用)、オプションとしてグローバルリソースを使用することができます(以下のスクリーンショットを参照)。メニューにあるデータベースに関連するコマンドでも、データベース型のグローバルリソースを使用することができます。メニューにあるデータベースに関連するコマンドでも、データベース型のグローバルリソースを使用することができます。



データベースを開くダイアログにてグローバルリソースアイコンをクリックすると、現在アクティブなグローバルリソース XML ファイル[935]に定義されている全てのデータベース型グローバルリソースが表示されます。目的のグローバルリソースを選択して、「**接続**」をクリックします。選択されたグローバルリソースに2つ以上の構成が割り当てられている場合、現在アクティブな構成に対応したグローバルリソースが使用され(「**ツール | アクティブな構成**」またはグローバルリソースツールバーを確認してください)、接続が確立されます。DB から XML スキーマを作成[1339]や DB データのインポート[1334]に記述されているように、使用するデータ構造やデータを選択する必要があります。

20.2.3 構成の切り替え

同時にアクティブにすることのできる構成は1つだけで、その設定はアプリケーション全体に対して適用されます。これは、アクティブな構成が、現在開いている全てのファイルやデータソース接続のエイリアスに対して有効であることを意味します。エイリアスにアクティブな構成と同じ名前の構成が割り当てられていない場合、デフォルトの構成がそのエイリアスに対して使用されます。構成を変更する例として、XSLT ファイルが複数の構成を持つグローバルリソースを介して XML ドキュメントに割り当てられたと仮定しましょう。各構成は異なるファイルにマップされます。ですから、ファイルの選択はアプリケーションのアクティブな構成としてどの構成が選択されているかにより決まります。

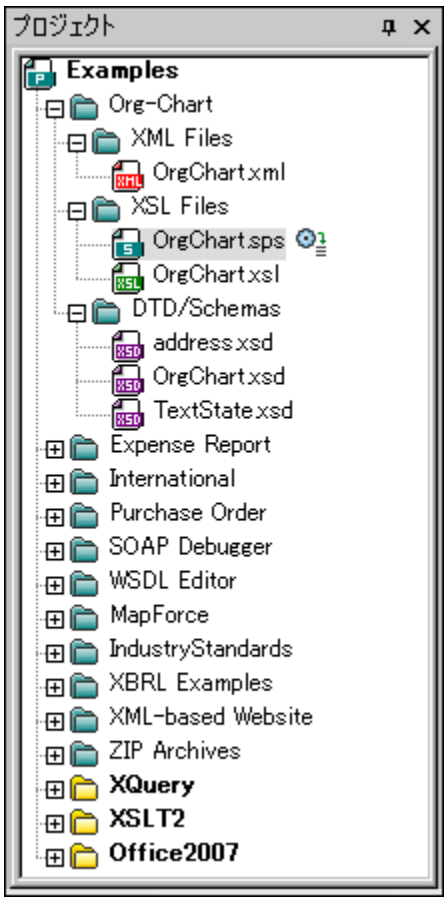
構成の切り替えるには2つの方法があります:

- メニューコマンド「**ツール | アクティブな構成**」を選択すると、グローバルリソース XML ファイル内にある全ての構成がサブメニューに表示されます。目的の構成を選択します。
- グローバルリソースコンボボックスのツールバーから、目的の構成を選択します(以下のスクリーンショットを参照)。

アクティブな構成を切り替えるという方法により、グローバルリソースを介して割り当てられたソースファイルを変更することができます。

21 プロジェクト

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



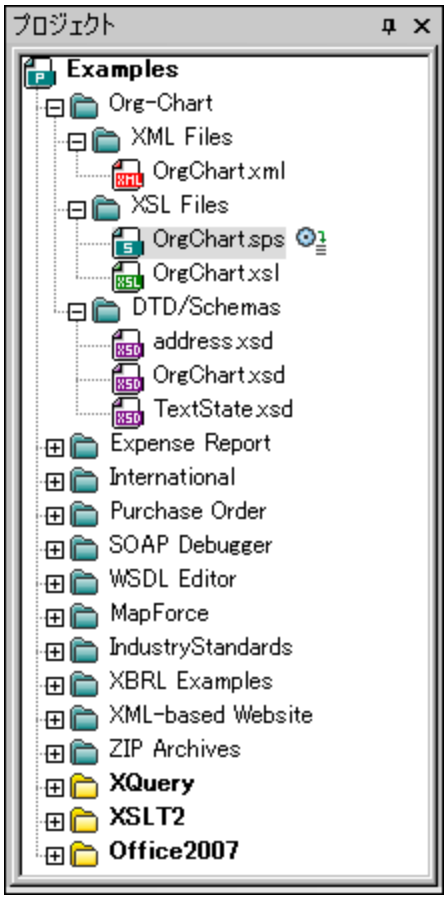
プロジェクトを使用することで、一緒に使用されるファイルをまとめて、ファイルへのアクセスを素早く行うことができます。更に、各フォルダーに対してスキーマやXSLT ファイルを定義することもでき、フォルダーに対するバッチ処理を行うこともできます。

このセクションでは[プロジェクトの作成と編集方法](951)ならびに[プロジェクトの使用方法](955)を説明します。

(C) 2015-2021 Altova GmbH

21.1　プロジェクトの作成と編集

プロジェクトは(以下のスクリーンショットにある)プロジェクト ウィンドウ[110]とプロジェクト メニュー[1168]により管理されます。プロジェクトは一度に1つしか開くことができません。開かれたプロジェクトはプロジェクト ウィンドウ[110]に表示されます。



プロジェクトの新規作成と既存プロジェクトを開く

メニューコマンド「**プロジェクト | 新規プロジェクトの作成**」により新たなプロジェクトが作成されます。既存のプロジェクトは「**プロジェクト | プロジェクトを開く**」メニューコマンドにより開くことができます。新たに開かれたプロジェクト(新規プロジェクトか既存のプロジェクトかに関わらず)は、それまでプロジェクト ウィンドウで開かれていたプロジェクトを置き換えます。それまで開かれていたプロジェクトに(下のスクリーンショットにあるように、フォルダーの隣のアスタリスクとして示される)保存されていない変更が含まれている場合、変更を保存するかどうかを決定することになります。

プロジェクトの名前付けと保存

新規プロジェクトの名前は保存時に決定されます。プロジェクトの保存には、「**プロジェクト | プロジェクトの保存**」コマンドを使用し、プロジェクトは .spp というファイル拡張子で保存されます。プロジェクトが変更されると、その変更を反映するためにプロジェクトを保存しなければなりません。(プロジェクト ウィンドウのトップレベルにある)プロジェクトの名前は、Windows のファイルシステムを使ってしか変更することができず、XMLSpy のGUI を使っての名前変更はできないことご注意してください。サブフォルダーの名前はGUI により変更することができます。

プロジェクトの構成

プロジェクトはフォルダーとファイルから成るツリー構造により管理されます。フォルダーとファイルはどの階層にも作成することができ、その深さにも制限はありません。サブフォルダーやファイル、またはリソースを作成するには、プロジェクトウィンドウ内のフォルダーを選択し、「**プロジェクト**」メニューにあるコマンド、またはコンテキストメニューを使用します。プロジェクトに追加されたファイル、フォルダー、そしてリソースを削除し、ドラッグによりプロジェクトツリー内の他の場所に移動することができます。

新規プロジェクトが作成されると、ファイルをその種類(XML やXSL など)により管理するデフォルトの構造が表示されます(以下のスクリーンショットを参照)。

フォルダーに定義されたプロパティにより、各フォルダーにはファイル拡張子が割り当てられています。ファイルがフォルダーに追加されると、そのファイル拡張子に従って、ファイルは自動的に対応するフォルダーに追加されます。各フォルダーに対して、対応するファイル拡張子を割り当てることができます。

プロジェクトに追加することのできるもの

フォルダー、ファイル、そしてその他のリソースは、トップレベルのプロジェクトフォルダーまたはプロジェクト内にある任意の階層のフォルダーに追加することができます。フォルダーには、(i) プロジェクトフォルダー、(ii) 外部フォルダー、(iii) 外部 Web フォルダー、の3種類あります。

オブジェクトを追加するには、関連するフォルダーを選択し、必要なコマンドをそのフォルダーのコンテキストメニュー、または「**プロジェクト**」メニューから選択します。以下のオブジェクトをプロジェクトフォルダーに追加することができます:

- *プロジェクトフォルダー(緑色)* はプロジェクトのコンテンツを構造化するためにプロジェクトに追加されるフォルダーです。プロジェクトフォルダーには(そのフォルダーのプロパティから)対応するファイルの拡張子を割り当てることができます。ファイルがフォルダーに追加されると、そのファイルの拡張子にマッチした最初のフォルダー内に配置されます。従って、複数のファイルがフォルダーへ追加されると、それぞれのファイルの拡張子に従って、対応するフォルダーに配置されることになります。
- *外部フォルダー(黄色)* はファイルシステムのフォルダーです。外部フォルダーがプロジェクトに追加されると、その外部フォルダー内にある全てのファイル、サブフォルダー、そしてサブフォルダーのファイルがプロジェクトに追加されます。外部フォルダーに対してファイル拡張子を定義すると、そのフォルダー内のファイルをフィルタリングする効果を持ちます。
- *外部 Web フォルダー*は外部フォルダーに似ていますが、Web サーバー上にあるフォルダーを指定し、アクセスのためにユーザー認証を行う必要があります。外部のWeb フォルダーに対してファイル拡張子を定義すると、そのフォルダー内のファイルをフィルタリングする効果を持ちます。
- *ファイル*はフォルダーの選択を行ない、以下の3つのファイル追加コマンドのどれかを選択することでフォルダーに追加することができます: (i)「**プロジェクトにファイルを追加**」コマンドにより開かれるダイアログにて追加するファイルを選択するか、(ii)「**プロジェクトにアクティブなファイルを追加**」コマンドによりメインウィンドウでアクティブなファイルをプロジェクトに追加する、または(iii) プロジェクトに「**アクティブならびに関連するファイルを追加**」コマンドによりアクティブな XML ファイルに関連するファイル(例えばXML スキーマやDTD)も一緒にプロジェクトにファイルを追加します。(例えばXSLT ファイルのような)処理命令により割り当てられているファイルは関連するファイルとは見なされないことに注意してください。
- *グローバルリソース*はファイル、フォルダー、そしてデータベースリソースへのエイリアスです。その定義方法と使用方法については、グローバルリソース[934]のセクションにて記述されます。*URL* を使えばURL によりオブジェクトを指定することができます。

- *Altova スクリプトプロジェクト* は.asprj 拡張子を持つファイルで、XMLSpy プロジェクトに関連付けることができます。これによりマクロやその他のスクリプトがプロジェクトで利用できるようになります。スクリプトプロジェクトの作成方法とXMLSpy プロジェクトへの割り当て方法は[スクリプティング](1505)のセクションを参照ください。

Project and folder properties

(検証のためのスキーマと変換のためのXSLT など) プロパティをプロジェクト全体だけでなく、個別のフォルダーのために設定することができます。プロジェクト全体または個別のフォルダーのために検証と変換などのアクションを実行することができます。アクションを実行するには、プロジェクトまたはフォルダーを右クリックして、表示されるコンテキストメニューから実行するアクションを選択します。

フォルダーのプロパティはフォルダーを選択してメニューコマンド [**プロジェクト | プロパティ**] (またはフォルダーのコンテキストメニューコマンドプロパティ)を使用してアクセスするそのフォルダーのプロパティダイアログに保管されています。 フォルダーの以下のプロパティはプロパティダイアログ内で定義し編集可能です:

- *フォルダー名* : トップレベルのプロジェクトフォルダーに対しては編集できません(フォルダー名の代わりにファイルパスが表示されます)。
- *ファイル拡張子* : プロジェクトのトップレベルフォルダーに対しては編集できません。これはセミコロンで区切られたファイル拡張子のリストです(例、`xml;svg;wml`)。このリストはファイルがプロジェクトに追加される際どのファイルがフォルダーに追加されるかを決定します。例えば、アクティブで関連するファイルがプロジェクトに追加されると、がファイルが追加されるフォルダー決定します。
- *検証* : フォルダー内のXML ファイルの検証において使用されるDTD またはXML スキーマを指定します。
- *変換* : (i) フォルダー内のXML ファイルの変換において使用されるXSLT ファイルと、(ii) フォルダー内のXSLT ファイルが変換を行うXML ファイルを指定します。
- *出力先* : 変換の結果生成されるファイルを保存するフォルダーと拡張子を指定します。*Authentic ビューのSPS ファイル* : フォルダー内のXML ファイルをAuthentic ビューにより閲覧、編集するためのSPS ファイルを指定します。

以下の点に注意してください:

- フォルダーで設定されたプロパティはプロジェクトの同じプロパティをオーバーライドします。
- プロパティがプロジェクトで設定されている場合、同じプロパティが設定されていないすべてのフォルダーに適用されます。
- プロジェクトでアクションが実行されると、プロジェクトの全てのフォルダー内の全ての適用可能なファイル型に適用されます。例えば、検証がプロジェクトで実行さえる場合、検証はプロジェクトの全てのフォルダー内の全てのXML ファイルで実行されます。この場合、プロジェクトのために設定されたスキーマは他のスキーマにスキーマ検証プロパティが設定されているフォルダー内のXML ファイル以外のすべての検証のために使用されます。

詳細に関しては、[**「プロジェクト | プロパティ」**](1194) コマンドの説明を参照してください。

プロジェクト内のソース管理

プロジェクトではMicrosoft Visual SourceSafe 互換のソース管理システムがサポートされます。この機能の使用方法については、マニュアル内の[対応するユーザーリファレンスのセクション](1171)を参照ください。

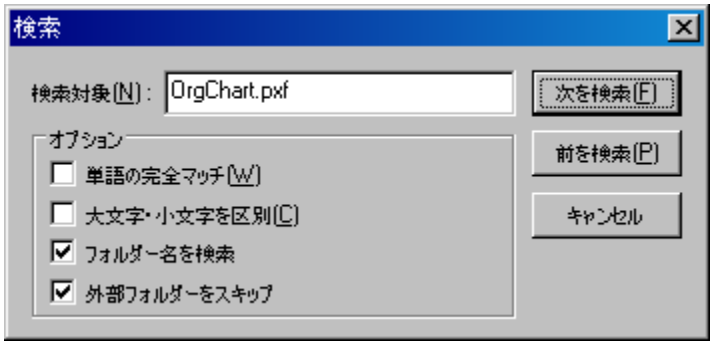
プロジェクトの保存

ファイルの追加や削除、プロジェクトプロパティの修正といったプロジェクトの変更は、**「プロジェクトの保存」** コマンドにより保存されなければなりません。

プロジェクト内の検索

ファイルの名前、またはファイル名の一部からプロジェクト内部にあるファイルやフォルダーを検索することができます。検索が行われると、該当するファイルやフォルダーが1つずつハイライトされた状態で表示されます。

検索を行うには、検索を行うプロジェクトフォルダーをプロジェクトウィンドウにて選択し、**「編集 | 検索」** メニューコマンドを選択するか、「Ctrl」+ F キーを押下します。検索ダイアログが表示されるので、検索を行う文字列を入力し、目的に従うかたちで検索オプションを指定します(以下のスクリーンショットを参照)。検索オプションについては以下を参照ください。



以下の検索オプションが利用できます:

- 単語の完全マッチ: 入力された検索語とファイルまたはフォルダー名が完全に一致するかチェックされ、厳格な検索を行うことができます。ドット (".") の前後に入力された文字列も単語として認識され、例えばファイルの拡張子だけを入力して検索を行うことができます。
- 大文字小文字を区別: 検索語として入力された文字列の大文字と小文字を区別するかどうか指定することができます。
- フォルダー名を検索: フォルダー名を検索の対象に含めます。選択されていない場合、ファイル名だけが検索されます。
- 外部フォルダーをスキップ[1186]: 外部フォルダーを検索の対象に加える、または対象から外す事ができます。外部フォルダーはシステムまたはネットワーク上に実在しているフォルダーで、プロジェクト内部で論理的に作成されたプロジェクトフォルダーとは異なるものです。

検索が正常に行われた場合、最初にマッチしたアイテムがプロジェクトウィンドウにてハイライトされます。検索ダイアログにある「次を検索」ならびに「前を検索」ボタンをクリックすることで、その他のマッチを参照することができます。

プロジェクトの更新

外部フォルダーに対して変更がなされた場合、プロジェクトが再ロードされるまでその変更はプロジェクトウィンドウに反映されません。

(C) 2015-2021 Altova GmbH

21.2　プロジェクトを使用する

プロジェクトを使うことで同時に複数のファイルに対して設定を行ったり、バッチコマンドを設定し、ワークスペースの管理を効率的に行うことができます。プロジェクトを使用することで、作業のスピードが飛躍的に向上します。

プロジェクトを使う利点

プロジェクトを使用する利点を以下に示します:

- ファイルの拡張子などの条件により、複数のファイルやフォルダーを1つのフォルダーにグループ化することができます。
- スキーマやXSLT ファイルをフォルダーに割り当てることができます。複数のスキーマやファイルを使って XML ファイルの検証を素早く行うことができます。XML ファイルを複数のフォルダーに追加して、各フォルダーにて異なるスキーマやXSLT を定義してください。
- 各フォルダーに対して、バッチ処理を適用することができます。バッチ処理で使用することのできるコマンドは以下に記されます。
- 変換の出力先となるフォルダーを指定することができます。

素早いアクセスを実現するリソース管理

フォルダーやファイルリソースをツリー構造で管理することができ、プロジェクト内にある様々なフォルダーやファイルの全容を確認し、目的のファイルへ素早く辿りつけることができます。プロジェクトウィンドウで表示されているファイルをダブルクリックして開いてください。プロジェクトへのファイルやフォルダーの追加は、必要に応じて素早く行うことができ、必要でないファイルやフォルダーを削除することもできます。他のプロジェクトで作業を行ないたい場合は、現在開かれているプロジェクトを閉じて、目的のプロジェクトを開いてください。

バッチ処理

フォルダー内にあるファイルのバッチ処理は、それがトップレベルのプロジェクトフォルダーであるかどうかに関わらず、**そのフォルダーのコンテキストメニューから利用できます**（フォルダーにて右クリック）。バッチ処理のステップは以下のとおりです:

1. フォルダーのプロパティダイアログにて検証や変換に使用するファイルを定義します。
2. 変換の出力先となるフォルダーを指定します。出力先のフォルダーが指定されなかった場合、プロジェクトツリー上の親フォルダーにて指定されている出力先フォルダーが使用されます。
3. コンテキストメニューからバッチ処理コマンドを使用します。メニューオプションのXML、DTD/スキーマ、または**XSL/XQuery** から対応するコマンドを選択すると、メインウィンドウにてアクティブになっているドキュメントに対してのみコマンドが実行され、プロジェクトウィンドウのプロジェクトフォルダーには影響を与えません。

（トップレベルまたはその他の）プロジェクトフォルダーのコンテキストメニューでは、以下のコマンドがバッチ処理で利用できます:

- *整形式をチェック*：バッチ処理中に発見したエラーは、全てメッセージウィンドウに表示されます。
- *検証*：バッチ処理中に発見したエラーは、全てメッセージウィンドウに表示されます。
- *変換*：フォルダーのプロパティダイアログにて指定した出力フォルダーに変換の出力が保存されます。フォルダーが指定されなかった場合、プロジェクトツリー上の親フォルダーにて指定されている出力先フォルダーが使用されます。親（または祖先）フォルダーにて出力先フォルダーが定義されていない場合、ドキュメントウィンドウが開かれ、各変換の結果が表示されます。XSL-FO 変換ではXML またはFO ドキュメントをPDF へ変換します。
- *DTD/スキーマの生成*：スキーマが生成される前に、出力先フォルダーを指定するよう促されます。生成されたスキーマファイルはそのフォルダーに保存され、個別のウィンドウにて表示されます。

メモ:　バッチ処理を実行するには、プロジェクトウィンドウの対応するフォルダーにてコンテキストメニューを使用してください。メニューオプションのXML、DTD/スキーマ、またはXSL/XQuery から対応するコマンドを使用しないでください。これらのコマンドは、メインウィンドウにてアクティブになっているドキュメントに対して実行されます。

RaptorXML Server を使用しての検証とXSLT/XQuery

プロジェクトフォルダーのコンテキストメニューコマンドを使用して、RaptorXML Server をハイパフォーマンスXML 検証とXSLT/XQuery 変換に使用することができます。詳細に関してはRaptorXML Server 957 のセクションを参照してください。

(C) 2015-2021 Altova GmbH

22 RaptorXML Server

Altova RaptorXML Server が、インストールされ、ネットワーク上でライセンス与えられており、XMLSpy のインストールにアクセスが与えられている場合、XML および XBRL*ドキュメントを検証し、XSLT と XQuery の変換[975]、の実行をするために、RaptorXML Server (RaptorXML または Raptor と以下省略されます) を使用することができます。XMLSpy プロジェクト フォルダー内のアクティブなドキュメント または すべてのドキュメントを検証することができます。検証の結果 は、メッセージウィンドウの GUI に表示されます。

XMLSpy を使用して以下を行うことができます (i) ドキュメントの検証 (ii) XMLSpy のエンジンまたは RaptorXML Server を使用して、XSLT/XQuery の変換の実行。Raptor を使用する利点の一つは、広範囲に及ぶ検証の個別の検証を構成できることです。更に、Raptor オプション のセットを XMLSpy 内の"構成"として、保管することができ、特定の Raptor 検証のための構成を選択します。Raptor の使用は、大きなデータコレクションを検証する際にとても役に立ちます。

メモ 実際のパフォーマンスは、検証のために RaptorXML Server により使用される PC プロセッサーコアの数によります。使用するコア数が多いほど、処理は速くなります。

***メモ** Raptor には、2 つのエディションがあります。: (XML 検証のための) *RaptorXML Server* および (XML および XBRL 検証のための) *RaptorXML+XBRL* 。XBRL ドキュメントを検証する場合は、RaptorXML+XBRL Server を使用します。RaptorXML(+XBRL) Server の詳細に関しては、 Altova Web サイト および ユーザーマニュアル: *RaptorXML Server* および *RaptorXML+XBRL Server* を参照してください。

メモ RaptorXML Server は Web ソケットの更新をサポートしないため HTTP プロキシと一緒に使用することはできません。この問題が発生すると RaptorXML Server ホストをプロキシを無視するリストに追加してください。

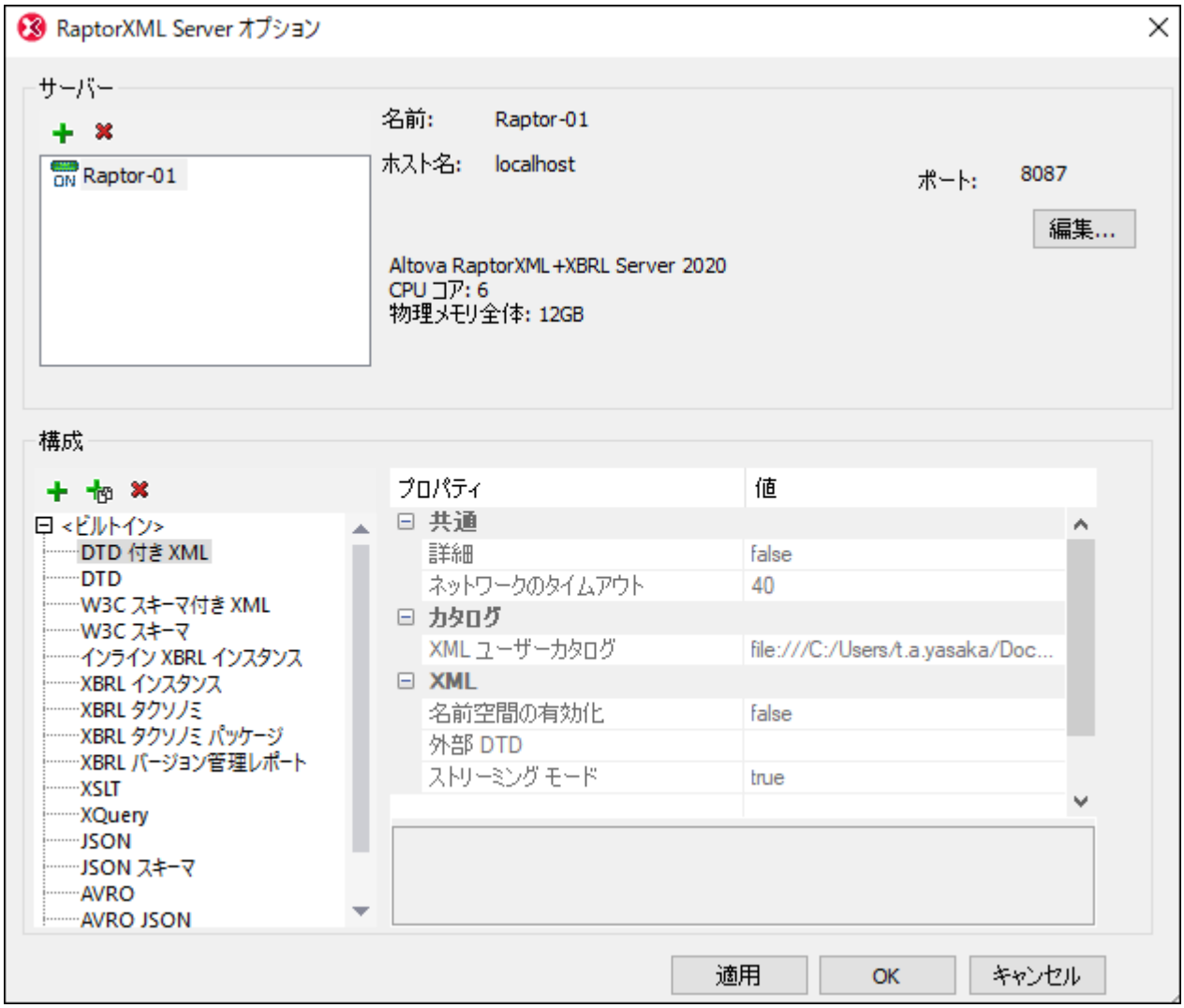
RaptorXML Server を使用して検証または変換する方法

RaptorXML Server を使用して、XML または XBRL ドキュメントを検証、または、XSLT や XQuery 変換を実行するには、XMLSpy は、サーバーへのアクセス方法、検証のため Raptor にパスするオプションなどの使用する RaptorXML Server についての情報を必要とします。XMLSpy 内でこの情報は以下のとおり管理されます:

1. *Raptor サーバーのプールにサーバーを追加する*[958]。このステップでは、, RaptorXML Servers が、プールに追加され、XMLSpy 内に補完されている各サーバーの情報にアクセスすることができます。各サーバーは、名前により識別されています。
2. *各 サーバーのための構成の定義*[959]。構成は、Raptor 検証 オプションのセットです。各サーバーは、複数の構成を持つことができます。検証に関しては、構成を1つ選択すると、選択された構成がアクティブな構成になります。
3. *検証に使用する サーバー構成 の選択*[962]。サーバーとその構成をアクティブな 構成に選択します。アクティブな構成は、後のすべての検証で使用されます。
4. *検証*[962] または Raptorを使用して *XSLT/XQuery 変換の実行*[975]。

22.1 サーバーとサーバー構成の追加

RaptorXML Server オプションダイアログ（下のスクリーンショット、「ツール | Raptor Servers の管理」1434）内で、Raptor Servers で使用することのできるプールに複数の Raptor Servers の追加 958 し、各サーバーのための複数の構成の定義 959 に追加することができます。各サーバーのために定義された構成と共にサーバーを追加するには、「ツール | Raptor Servers および構成」1438 サブメニューを使用します。このサブメニューで、Raptor 検証のために使用するを選択します。



Raptor Server を追加する

ダイアログの *サーバーペイン*（*下のスクリーンショット*）内から、「**サーバーの追加**」アイコンをクリックします。アイコンをクリックして、Raptor Server、がインストールされているマシンの名前（ホスト名）、および Raptor Server のポートを認識するための名前を入力します。「OK」をクリックして設定を保存します。

- *名前:* 選択されるすべての文字列。XMLSpy 内では、特定のRaptorXML Server を認識するために使用されています。
- *ホスト名:* Raptor Server がインストールされているマシンのネットワークの名前 またはIP アドレス。処理はホスト名よりもIP アドレスを使用した場合のほうが速くなります。`localhost`（ローカルマシン）に対応するIP アドレスは`127.0.0.1` です。
- *ポート:* Raptor Server がアクセスされるポート。このポートは、（`server_config.xml` という名前の）Raptor の構成ファイル内で指定されています。ポートは、固定され、サービスへのリクエストが対応できるように知らされている必要があります。Raptor 構成ファイルの詳細は、*RaptorXML Server* および *RaptorXML+XBRL Server* のユーザーマニュアルを参照してください。

サーバー情報を入力すると、「OK」をクリックします。入力したサーバー名が（ペインの左側の）サーバーリストに表示されます。サーバーの名前の横に緑のアイコンが表示され、サーバーが開始され、実行中であることを示しています。サーバーの詳細はペインに表示されています（*上のスクリーンショット参照*）。赤いアイコンはサーバーがオフラインであることを示します。サーバーを検出することができない場合、エラーメッセージが表示されます。

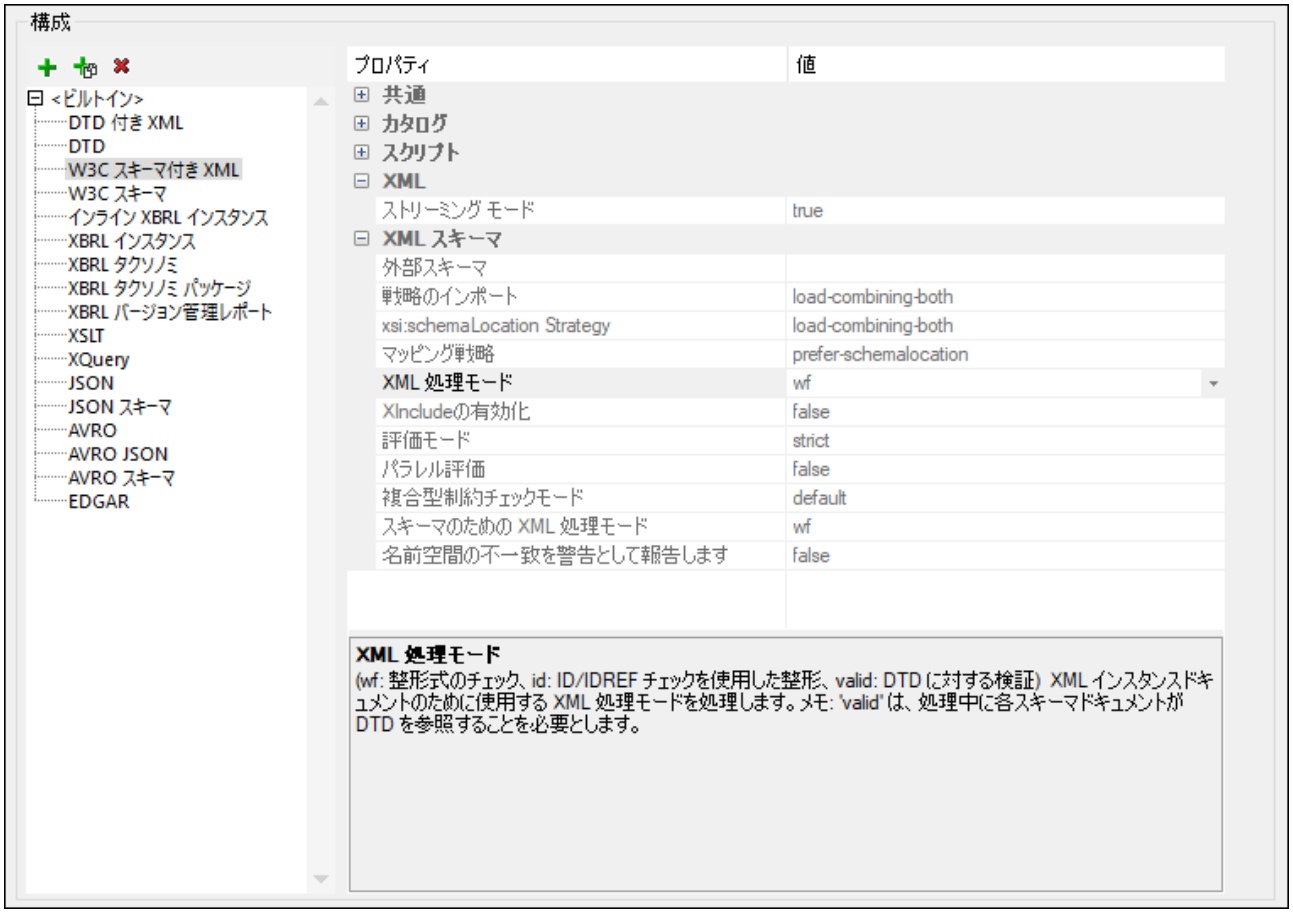
メモ　Raptor Server は、サーバーが追加される際に実行中である必要があります。これはXMLSpy がサーバーの情報を取得し、保管するために必要です。サーバーが追加された後、サーバーがオフラインまたは検出されない場合、これらのシナリオは赤いアイコンまたはエラーメッセージで通知されます。

サーバーの名前、ホスト名、またはポートを編集する場合、左側のペインのサーバーを選択して、「**編集**」ボタンをクリックし、表示されるダイアログ内の情報を変更します。プールからサーバーを削除する場合は、サーバーを選択して「**選択されたサーバーを削除**」アイコンをクリックします。

Server 構成

構成とは、RaptorXML 検証オプションのセットです。サーバーが追加されると、構成を持ち、`default` と名前が付けられます。これはRaptorXML オプションのセットに設定されたデフォルトの値です。他のオプションの値を含む新しい構成を追加することもできます。複数のサーバー構成を追加した後、アクティブな構成になる1 つの構成を選択します。これが「**サーバー上で検証**」コマンドが実行される時に使用される構成です。

構成ペインは2 つの部分により構成されています: (i) それぞれが変更可能なドキュメント型のリストを含む構成を表示する左側のペイン、(ii) 左側のペインで選択されたドキュメント型のための検証オプションを表示する右側のペイン。右側のペインの下には選択されたオプションの説明が表示されています（*上のスクリーンショット参照*）。



構成の追加

RaptorXML Server オプションダイアログの構成ペイン内から(*上のスクリーンショット*)「**構成の追加**」をクリックします。デフォルトのオプション値に新規構成が追加されます。「**選択された構成をコピー**」をクリックすることにより新規構成を作成することもできます。これによりコピーされた構成のオプション値と同じ値を持つ新しい構成が作成されます。新たしい構成は、型 `config<X>` のデフォルトの名前と共に作成されます。構成の名前をダブルクリックして新しい名前を入力して編集します。構成のオプション値のすべてを編集することができます。

構成のオプション値の編集

左側のペインからドキュメント型を選択します。これにより右側のペイン内に選択されたドキュメント型の検証オプションが表示されます。オプションの値を編集するには、(オプションの値の型に従い)以下を行います:

- 値が定義済みの値のセットの1つの場合、そのオプションの値の列のコンボボックスから値を選択します。
- 値に制約がある場合、オプションの値フィールドをクリックして、値を入力します。
- 値がファイルパスの場合、値を入力できるだけでなく、オプションの値の列の「**参照**」ボタンを押して、使用するファイルを参照します。

オプションを選択すると、右側のペインの下のボックスに詳細が表示されます。各オプションの詳細は、RaptorXML Server および RaptorXML(+XBRL) Server ユーザーマニュアルのコマンドラインインターフェイスのセクションを参照してください。

構成の削除

左側のペインから、削除する構成を選択して、「**選択された構成を削除**」をクリックします。

Visual Studio および Eclipse 内の XMLSpy

Visual Studio[1005] および Eclipse[1010], に XMLSpy が統合されると、IDE のアクティブの構成が XMLSpy のスタンドアロンバージョン内のアクティブな構成として設定されます。

22.2 RaptorXML Server を使用して検証

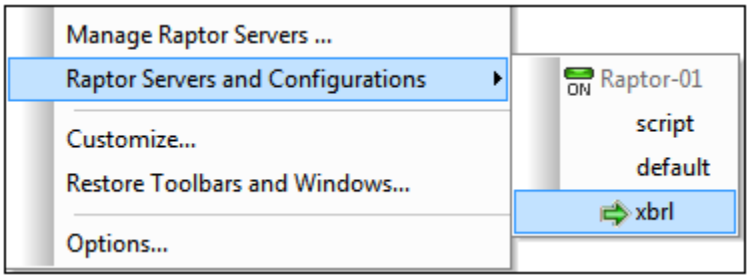
RaptorXML Server を使用して、XML および XBRL* ドキュメント を検証することができます。検証には 2 つの ステップ が含まれます:

- 検証のために使用されるサーバーとサーバー構成を選択します。
- (**サーバー上での検証** コマンドを使用した *以下を参照*) 検証の実行

***メモ** Raptor には、2 つのエディションがあります。: (XML 検証のための) *RaptorXML Server* および (XML および XBRL 検証のための) *RaptorXML+XBRL* 。XBRL ドキュメントを検証する場合は、RaptorXML+XBRL Server を使用します。RaptorXML(+XBRL) Server の詳細に関しては、 Altova W eb サイト および ユーザーマニュアル: *RaptorXML Server* および *RaptorXML+XBRL Server* を参照してください。

使用するサーバー構成の選択

複数のサーバーで複数の構成を定義した場合、サーバーとそのアクティブな構成として 1 つの構成を選択することができます。アクティブな構成は、以降の検証に使用されます。カーソルを「**ツール | Raptor Servers および 構成**」コマンドにポイントして、追加されたすべてのサーバーとその詳細が含まれるサブメニューから、を含む表示されるサブメニューが表示されます (*下のスクリーンショット参照*) 。アクティブな構成と設定するサーバー構成を選択します。したのスクリーンショットでは、`Raptor-01` という名前の サーバーの (緑色の矢印で示されている) `xbrl` 構成がアクティブな構成として選択されています。



RaptorXML Server を使用して検証する

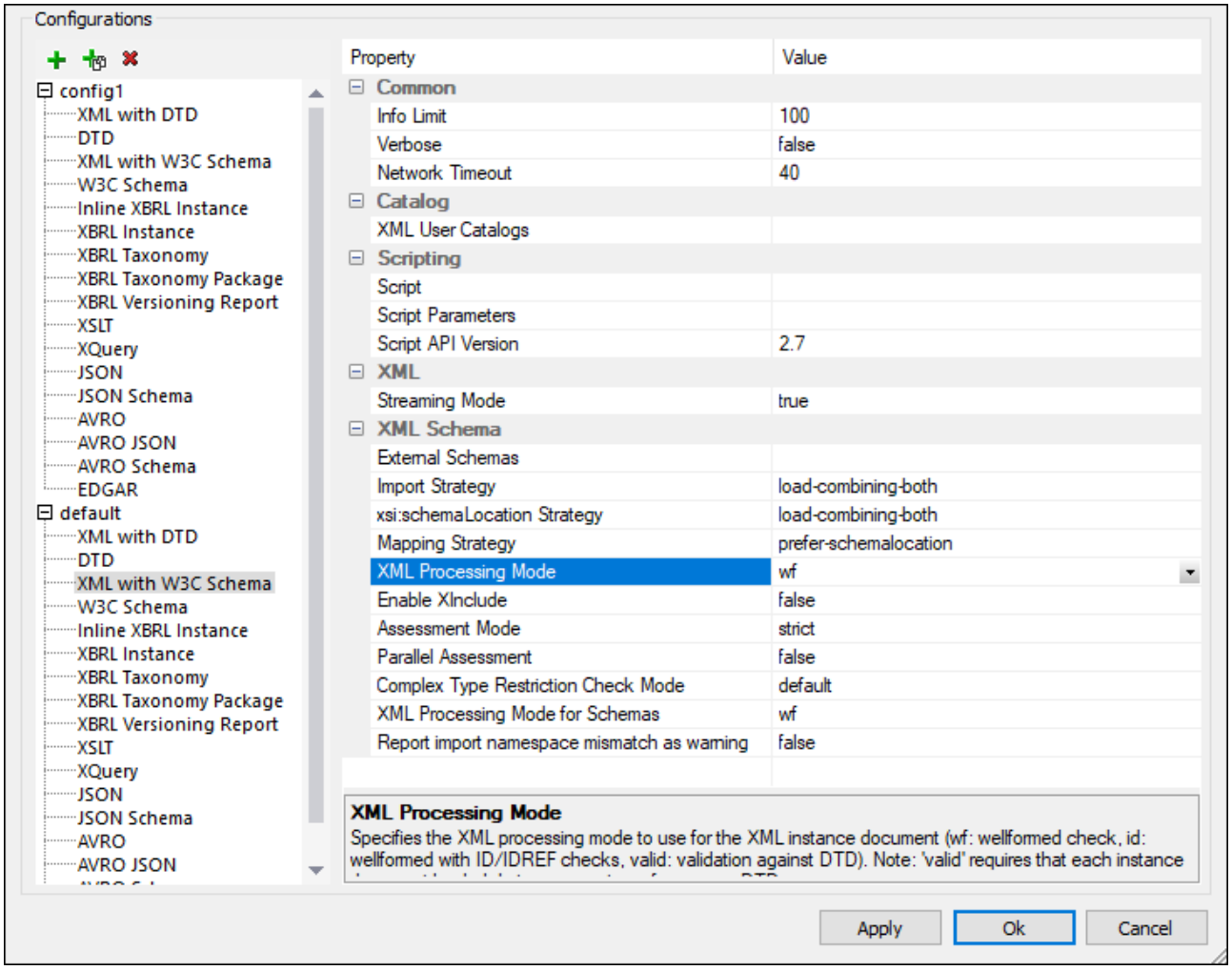
XMLSpy の検証エンジンまたは、RaptorXML Server を使用して XML および XBRL ドキュメントを検証することができます。以下の方法を使用してください:

- ツールバーアイコン **サーバー上で検証する**をクリックします。
- コマンド「**XML | サーバー上で XML の検証する**」(**ハイパフォーマンス**)(Ctrl+F8) を選択します。
- プロジェクト、フォルダーまたはファイル内の入力ヘルパーのプロジェクトを右クリックし、「**サーバー上の XML を検証する**」(**ハイパフォーマンス**)を選択して、選択されたオブジェクト内の XML または XBRL データを検証します。

メモ テキストビュー、スキーマビュー、XBRL ビュー、および グリッドビュー内で Raptor 検証を使用することができます。

22.3 検証オプション

このセクションは、検証されるドキュメントの型により整理されています（下のスクリーンショットの左側のペインを参照してください）。例えば、*W3C スキーマを持つ XML* は、XML ドキュメントを W3C XML スキーマに対して検証します。左側のペインで検証が選択されると、検証の型で使用することのできる RaptorXML Server 検証 オプションが右側のペインに表示されます。これらのオプションは、*スクリプト* および *XML スキーマ*などのグループ別に整理されています（*下のスクリーンショット参照*）。



このセクションのサブセクションは、それぞれの RaptorXML Server 検証 オプションの説明へのリンクを含みます。

22.3.1 共通のオプション

オプションはすべての型の検証で共通です。

- 共通

  情報の制限
  詳細

ネットワークタイムアウト

カタログ

XML ユーザーカタログ

22.3.2 DTD に対して XML

DTD に対して XML データを検証するためのオプション

共通

情報の制限
詳細
ネットワークのタイムアウト

カタログ

XML ユーザーカタログ

XML

名前空間を有効化する
外部 DTD
ストリーミングモード

22.3.3 DTD

DTD の検証のためのオプション

共通

情報の制限
詳細
ネットワークタイムアウト

カタログ

XML ユーザーカタログ

スクリプト

スクリプト

(C) 2015-2021 Altova GmbH

スクリプト パラメーター
スクリプト API バージョン

22.3.4 W3C スキーマに対して XML

XML スキーマに対して XML データを検証するためのオプション

- 共通

  情報の制限
  詳細
  ネットワークのタイムアウト

- カタログ

  XML ユーザーカタログ

- スクリプト

  スクリプト
  スクリプト パラメーター
  スクリプト API バージョン

- XML

  ストリーミングモード

- XML スキーマ

  外部 スキーマ(xsd)
  インポートストラテジー(schema-imports)
  xsi:schemaLocation Strategy (schemalocation-hints)
  ストラテジーのマッピング(schema-mapping)
  XML 検証モード(xml-mode)
  XInclude の有効化 (xinclude)
  評価モード
  パラレル評価

22.3.5 W3C スキーマ

XML スキーマの検証のためのオプション

- 共通

  情報の制限

詳細
ネットワークのタイムアウト

- カタログ

XML ユーザーカタログ

- スクリプト

スクリプト
スクリプト パラメーター
スクリプト API バージョン

- XML スキーマ

インポートストラテジー(schema-imports)
xsi:schemaLocation Strategy (schemalocation-hints)
ストラテジーのマッピング(schema-mapping)
XML 検証モード(xml-mode)
XInclude の有効化(xinclude)

22.3.6 XBRL インスタンス

XBRL インスタントドキュメントの検証のためのオプション

- 共通

情報の制限
詳細
ネットワークのタイムアウト

- カタログ

XML ユーザーカタログ

- スクリプト

スクリプト
スクリプト パラメーター
スクリプト API バージョン

- XML スキーマ

インポートストラテジー(schema-imports)
xsi:schemaLocation Strategy (schemalocation-hints)
ストラテジーのマッピング(schema-mapping)

(C) 2015-2021 Altova GmbH

XInclude の有効化 (xinclude)
パラレル評価

- XBRL

ディメンション拡張子の有効化 (dimensions)
拡張可能な列挙 拡張子の有効化 (extensible-enumerations)
ユニットレジストリ名前空間の有効化 (utr)
XBRL スペックスキーマのプリロード (preload-xbrl-schemas)
タクソノミ パッケージ
参照されているDTS のみを検証(validate-dts-only)
XBRL 矛盾をエラーとして扱う(treat-inconsistencies-as-errors)

- XBRL フォーミュラ

フォーミュラ拡張子の有効化 (formula)
アサーション重要度の拡張の有効化 (assertion-severity)
フォーミュラスペックスキーマのプリロード (preload-formula-schemas)
条件を満たさないアサーションのレポート
検証メッセージ言語 (message-lang)
検証メッセージの役割 (message-role)

- XBRL テーブル

テーブル拡張子の有効化 (table)
テーブルスペックスキーマのプリロード (preload-table-schemas)
テーブルリンクベース名前空間

22.3.7 XBRL タクソノミ

XBRL タクソノミの検証のためのオプション

- 共通

情報の制限
詳細
ネットワークのタイムアウト

- カタログ

XML ユーザーカタログ

- スクリプト

スクリプト
スクリプト パラメーター
スクリプト API バージョン

- XML スキーマ

  インポートストラテジー(schema-imports)
  xsi:schemaLocation Strategy (schemalocation-hints)
  ストラテジーのマッピング(schema-mapping)
  XInclude の有効化(xinclude)

- XBRL

  ディメンション拡張子の有効化(dimensions)
  Enable 拡張可能な列挙 拡張子(extensible-enumerations)
  XBRL スペックスキーマのプリロード(preload-xbrl-schemas)
  タクソノミパッケージ
  XBRL 矛盾をエラーとして扱う(treat-inconsistencies-as-errors)

- XBRL フォーミュラ

  フォーミュラ拡張子の有効化(formula)
  フォーミュラスペックスキーマのプリロード(preload-formula-schemas)

- XBRL Table

  テーブル拡張子の有効化(table)
  テーブルスペックスキーマのプリロード(preload-table-schemas)
  テーブルリンクベース名前空間

22.3.8 XBRL タクソノミパッケージ

XBRL タクソノミパッケージの検証のためのオプション

- 共通

  情報の制限
  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

- Scripting

  スクリプト
  スクリプト パラメーター
  スクリプト API バージョン

- XML スキーマ

  インポートストラテジー(schema-imports)
  xsi:schemaLocation Strategy (schemalocation-hints)
  ストラテジーのマッピング(schema-mapping)

22.3.9 XBRL バージョンレポート

XBRL バージョンの検証のためのオプション

- 共通

  情報の制限
  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

- XMLスキーマ

  インポートストラテジー(schema-imports)
  xsi:schemaLocation Strategy (schemalocation-hints)
  ストラテジーのマッピング(schema-mapping)
  XInclude の有効化(xinclude)

22.3.10 XSLT

XSLT ドキュメントの検証のためのオプション

- 共通

  情報の制限
  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

- XML スキーマ

  インポートストラテジー(schema-imports)

xsi:schemaLocation ストラテジー(schemalocation-hints)
マッピングストラテジー(schema-mapping)
XML 検証モード (xml-mode)
XInclude の有効化 (xinclude)

- Java 拡張子

  Java 拡張子の無効化 (javaext-disable)
  バーコード拡張子のロケーション(javaext-barcode-location)

- チャート拡張子

  チャート拡張子の無効化(chartext-disable)

- .NET 拡張子

  .NET 拡張子の無効化 (dotnetext-disable)

- XEngines 共通

  XML をPSVI と共にロード(load-xml-with-psvi)

- XSLT

  XSLT エンジン バージョン (xslt-バージョン)
  テンプレートモード
  テンプレートエントリポイント

22.3.11 XQuery

XQuery ドキュメントの検証のためのオプション

- 共通

  情報の制限
  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

- XML スキーマ

  インポートストラテジー(schema-imports)

xsi:schemaLocation ストラテジー(schemalocation-hints)
マッピングストラテジー(schema-mapping)
XML 検証モード (xml-mode)
XInclude の有効化 (xinclude)

- Java 拡張子

  Java 拡張子の無効化 (javaext-disable)
  バーコード拡張子のロケーション(javaext-barcode-location)

- チャート拡張子

  チャート拡張子の無効化(chartext-disable)

- .NET 拡張子

  .NET 拡張子の無効化 (dotnetext-disable)

- XEngines 共通

  XML をPSVI と共にロード(load-xml-with-psvi)

- XQuery

  XQuery エンジン バージョン(xquery-version)
  XML 宣言の削除

22.3.12 JSON

JSON (インスタンス) ドキュメントの検証のためのオプション

- 共通

  情報の制限
  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

- JSON 検証

  フォーマットチェックの無効化
  JSON ライン

コメント付きのJSON

22.3.13 JSON スキーマ

JSON スキーマドキュメントの検証のためのオプション

- 共通

  情報の制限
  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

- JSON 検証

  フォーマットチェックの無効化

22.3.14 AVRO

Avro バイナリファイル内の一つ以上のデータブロックを各バイナリファイル内の対応するスキーマに対して検証するためのオプション

- 共通

  情報の制限
  詳細
  ネットワークタイムアウト

- カタログ

  XML ユーザーカタログ

22.3.15 AVRO JSON

JSON ドキュメントを AVRO スキーマに対して検証するためのオプション

- 共通

  情報の制限
  詳細

(C) 2015-2021 Altova GmbH

ネットワークタイムアウト

- カタログ

XML ユーザーカタログ

22.3.16 AVRO スキーマ

Avro スキーマ仕様に対して1つ以上のAvro スキーマドキュメントを検証するためのオプション

- 共通

情報の制限
詳細
ネットワークタイムアウト

- カタログ

XML ユーザーカタログ

22.3.17 EDGAR

EDGAR (エドガー、Electronic Data Gathering, Analysis, and Retrieval) は、米国証券取引委員会SEC (Securities and Exchange Commission) に企業により提出されたファイルを自動的に収集、検証、および分類するシステムです。EDGAR を介して検証する場合、Raptor は、内部 EDGAR スクリプトを使用してXBRL インスタンスドキュメントを検証します。以下の追加オプションの設定することとができます。

- EDGAR スクリプト パラメーター

EDGAR スクリプトは、EDGAR Filing Manual Volume II: EDGAR Filingで説明されているとおり、追加のチェックを行います。スクリプトは、以下のスクリプト パラメーターを指定することを許可します:

CIK	登録者名のCIK
submissionType	EDGAR 送信の型、例えば '10-K'
cikList	カンマで区切られたCIK のリスト: ','
cikNameList	'|Edgar|' により区切られる、cikList 内の各 CIK のための公式の登録者名のリスト
forceUtrValidation	UTR 検証を強制的に有効化するためにtrue に設定
edbody-url	埋め込まれたHTML フラグメントの検証に使用されるedbody.dtd へのパス
edgar-taxonomies-url	カンパニー拡張タクソノミタクソノミからの参照を許可するファイルのリストを含むedgartaxonomies.xml へのパス

- 共通

情報の制限
詳細
ネットワークタイムアウト

カタログ

XML ユーザーカタログ

XML スキーマ

インポートストラテジー(schema-imports)
xsi:schemaLocation ストラテジー(schemalocation-hints)
マッピングストラテジー(schema-mapping)
XInclude の有効化 (xinclude)
パラレル評価

XBRL

ディメンション拡張子の有効化 (dimensions)
拡張可能な列挙 拡張子の有効化 (extensible-enumerations)
XBRL スペックスキーマのプリロード (preload-xbrl-schemas)
タクソノミパッケージ
XBRL 非整合性をエラーとして扱う(treat-inconsistencies-as-エラー)

XBRL フォーミュラ

フォーミュラ 拡張子の有効化(formula)
フォーミュラ スペックスキーマのプリロード (preload-formula-schemas)

XBRL テーブル

テーブル拡張子の有効化 (table)
テーブルスペックスキーマのプリロード(preload-table-schemas)
テーブルリンクベース名前空間

(C) 2015-2021 Altova GmbH

22.4 RaptorXML Server使用しての XSLT と XQuery

RaptorXML Server を使用して以下を行うことができます (i) XSLT 変換 (ii) XML ドキュメント上のXQuery アップデートの実行。これらのアクションはプロジェクト[950]のみで使用することができ、以下の3 つのステップを含みます:

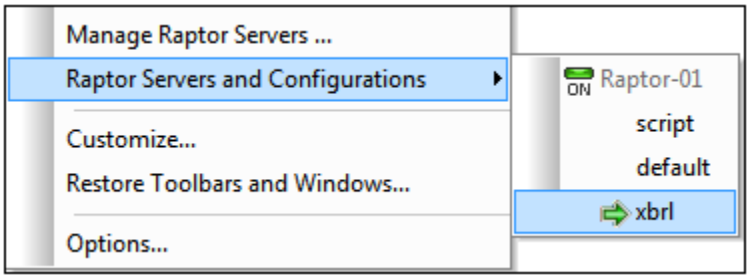
- ジョブに使用されるサーバーとサーバーの構成の選択。
- プロジェクト フォルダー[951]の設定と(プロジェクト プロパティダイアログ内で[951])使用するXSLT/XQuery ファイルの指定。XSLT/XQuery ファイルは、プロジェクト フォルダー内のすべてのXML ファイルのXSLT とXQuery 変換に使用されたファイルであるフォルダーのプロジェクト プロパティダイアログ[951]内に割り当てられます。プロジェクト フォルダー内の個々のXML ファイルにXSLT/XQuery ファイルを割り当てることはできません。XSLT/XQuery ファイルはフォルダー全体のみに割り当てることができます。
- XSLT 変換またはXQuery 更新および実行

メモ XSLT またはXQuery ドキュメントがJava 拡張関数 または.NET 拡張関数を使用する場合、ファイル パスを使用して、JAR ファイル(Java) または外部(登録されていない) アセンブリ ファイル(.NET)を検索します。これは、XMLSpy および RaptorXML Serverを使用して、同じ XSLT/XQuery ドキュメントが変換および実行に使用される場合、JAR ファイルへの または アセンブリ ファイルへのファイル パスがこれらのファイルを正確に検索する必要があります。

メモ RaptorXML Server がXMLSpy と同じマシン上にある場合、パフォーマンス向上のために、サーバー設定を `true` の値を持つ **`server.unrestricted-filesystem-access`** と設定します。更に詳しい情報は、RaptorXML Server 構成ファイルのドキュメントを参照してください。

使用するサーバー構成の選択

複数のサーバーで複数の構成を定義した場合、サーバーとそのアクティブな構成として1 つの構成を選択することができます。アクティブな構成は、以降の検証に使用されます。カーソルを「**ツール | Raptor Servers および 構成**」コマンドにポイントして、追加されたすべてのサーバーとその詳細が含まれるサブメニューから、を含む表示されるサブメニューが表示されます (*下のスクリーンショット参照*)。アクティブな構成と設定するサーバー構成を選択します。したのスクリーンショットでは、`Raptor-01` という名前の サーバーの(緑色の矢印で示されている) `xbrl` 構成がアクティブな構成として選択されています。



XSLT 変換の実行

XMLSpy またはRaptorXML Server のXSLT エンジンを使用してXSLT 変換 の実行をするには以下を行います:

- 変換されるXML ファイルが存在するプロジェクト フォルダーを右クリックします。このフォルダーはプロジェクト フォルダー全体またはプロジェクトの階層内の個別のフォルダーであることができます。
- 表示されるメニューから、「**サーバー上でのXSL 変換 (ハイパフォーマンス)**」を選択します。

メモ プロジェクト フォルダー内の個々のXML ファイルにXSLT/XQuery ファイルを割り当てることはできません。XSLT/XQuery ファイルはフォルダー全体のみに割り当てることができます。セクションの始め[975]を参照してください。

詳細に関しては、XSLT[428] およびXSLT 変換[1279]のセクションを参照してください。

XQuery の更新/実行の実行

XMLSpy またはRaptorXML Server のXSLT エンジンを使用してXQuery 更新/変換 の実行をするには以下を行います:

- 更新および実行するXQuery またはXML ファイルが存在するプロジェクトフォルダーを右クリックします。このフォルダーはプロジェクトフォルダー全体またはプロジェクトの階層内の個別のフォルダーであることができます。
- 表示されるメニューから、「**サーバー上でのXQuery 更新/実行 (ハイパフォーマンス)**」を選択します。

メモ プロジェクトフォルダー内の個々のXML ファイルにXSLT/XQuery ファイルを割り当てることはできません。XSLT/XQuery ファイルはフォルダー全体のみに割り当てることができます。セクションの始め 975 を参照してください。

詳細に関しては、XQuery 442 および XQuery/Update 実行 1285 のセクションを参照してください。

23 ファイル/ディレクトリの比較

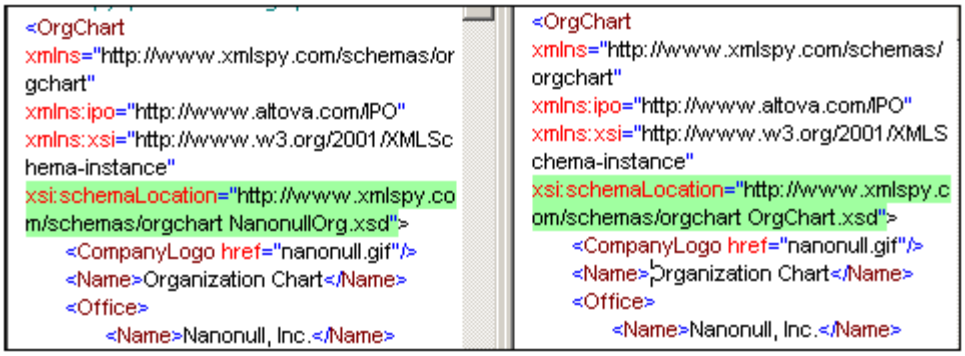
XMLSpy にはファイル比較ならびにディレクトリ比較機能が搭載されています。ファイル比較とディレクトリ比較は、「**ツール**」メニューの「**開いているファイルと比較**」ならびに「**ディレクトリを比較**」コマンドからそれぞれアクセスできます。ファイルの比較オプションは、メニューオプションの「**ツール | 比較オプション**」コマンドによりアクセスすることのできる設定ダイアログにて設定することができます。

各コマンドの詳細はユーザーリファレンスのセクション[1423]にて記述されます。このセクションのサブセクションではファイル比較[978]とディレクトリ比較[979]メカニズムの概要について説明します。

23.1 ファイルの比較

ファイル比較機能[1423]により、ファイルを開くダイアログやグローバルリソース[934]から選択されたファイルと、アクティブなファイルとの比較を行うことができます。以下ではそのメカニズムの概要を説明します。詳細についてはユーザーリファレンスのセクション[1423]を参照ください。

- ファイル比較セッションが開始されると、比較オプション[1430]ダイアログにて指定された設定が、そのセッションにて有効になります。
- ファイルはXML ファイル(ドキュメントの構造も比較の対象になります)、またはテキストファイルとして比較することができます。設定ダイアログ[1430]にて、(i) グリッドビューまたはテキストビューを選択し、テキストの比較だけを行うのチェックを外すことでXML 比較を、(ii) テキストビューを選択し、テキストの比較だけを行うにチェックを入れることでテキストの比較を行うことができます。
- 選択されたビュー(グリッドビューまたはテキストビュー)にて2つのファイルが隣接して開かれ、両方のファイル間の差異がハイライトされます(以下のスクリーンショットを参照)。



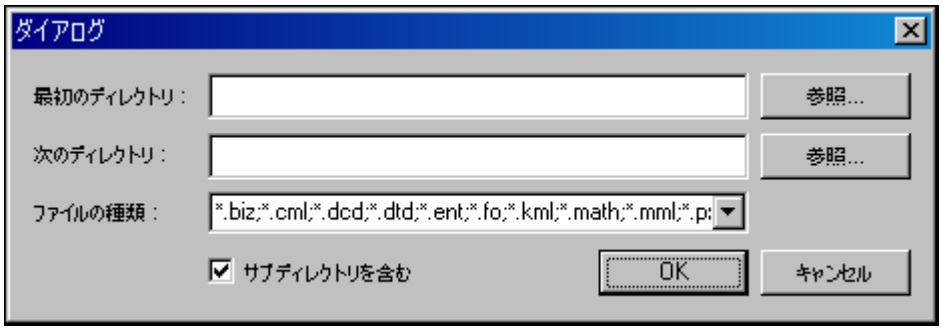
ファイルの比較コントロールウィンドウが表示され、差異のナビゲーションやマージを行うことができます。

設定ダイアログ[1430]では、どのXML コンポーネントを比較の対象とし、何を無視するか設定することができます。詳細については、ユーザーリファレンス[1423]の比較オプション[1430]セクションを参照ください。

(C) 2015-2021 Altova GmbH

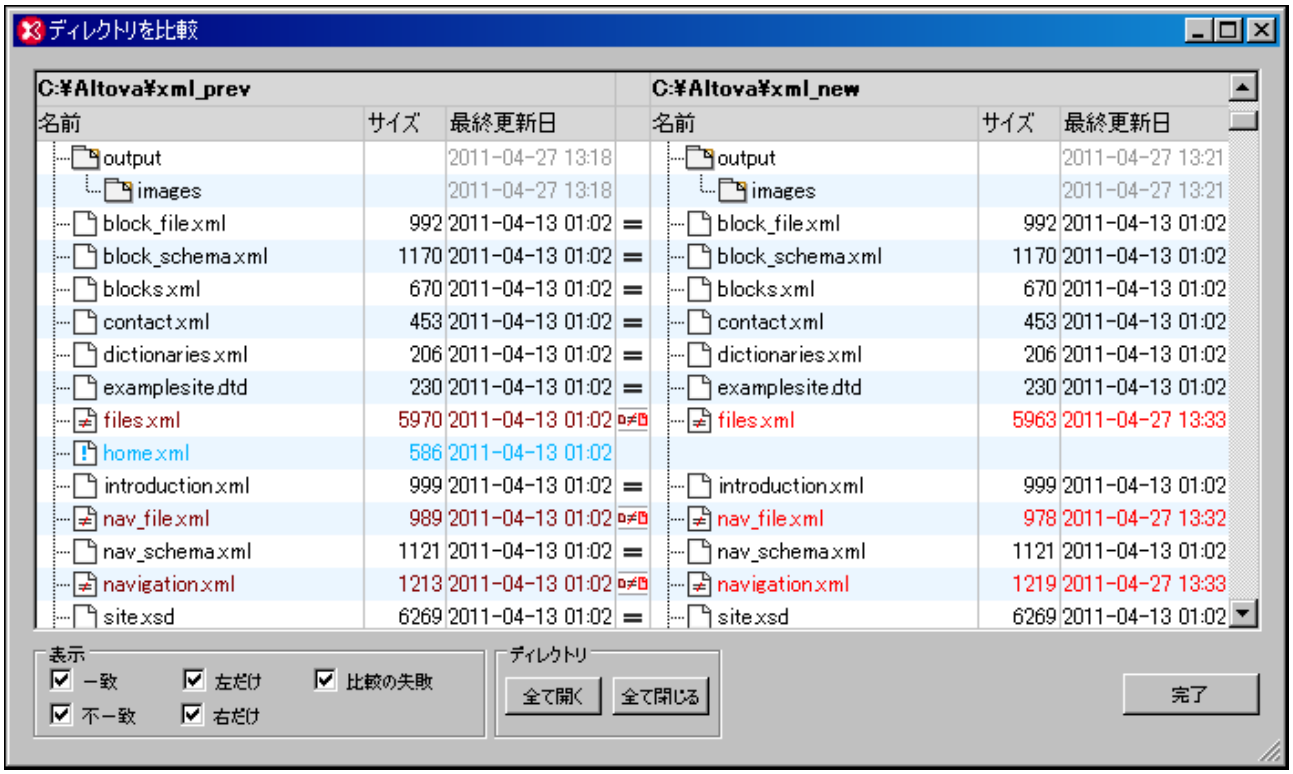
23.2　ディレクトリの比較

ディレクトリの比較機能[1427]を使うことで、フォルダーダイアログにて指定したディレクトリ同士を比較することができます。サブディレクトリも比較の対象とするか選択することができ、比較の対象となるファイルの種類を指定することができます。



ファイルの種類のリストの中にZIP ファイル拡張子を加えることで、ZIP アーカイブも比較の対象とすることができます。

ディレクトリの比較では、ファイルが欠けているか、同名のファイルが同じものであるかが比較されます。ファイル同士の比較は設定ダイアログ[1430]にて指定された設定をベースに行われます。ディレクトリ比較の結果は別のウィンドウにて表示されます（以下のスクリーンショットを参照）。



ディレクトリを比較ウィンドウ内のシンボルの意味や、管理の方法については、ユーザーリファレンス[1423]の「ディレクトリを比較」コマンドに関する記述を参照ください。ファイルの行をダブルクリックすると、ファイルの比較を開始することができます。

24 Source Control

The source control support in XMLSpy is available through the Microsoft Source Control Plug-in API (formerly known as the MSSCCI API), versions 1.1, 1.2 and 1.3. This enables you to run source control commands such as "Check in" or "Check out" directly from XMLSpy to virtually any source control system that lets native or third-party clients connect to it through the Microsoft Source Control Plug-in API.

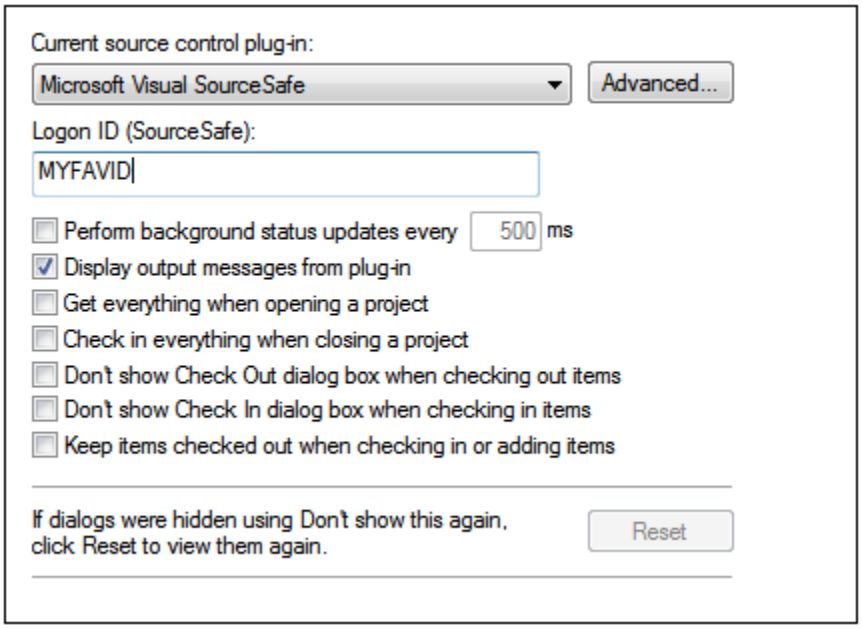
You can use as your source control provider any commercial or non-commercial plug-in that supports the Microsoft Source Control Plug-in API, and can connect to a compatible version control system. For the list of source control systems and plug-ins tested by Altova, see Supported Source Control Systems 983.

Installing and configuring the source control provider

To view the source control providers available on your system, do the following:

1. On the **Tools** menu, click **Options**.
2. Click the **Source Control** tab.

Any source control plug-ins compatible with the Microsoft Source Code Control Plug-in API are displayed in the **Current source control plug-in** drop-down list.



If a compatible plug-in cannot be found on your system, the following message is displayed:

```
"Registration of installed source control providers could not be found or is incomplete."
```

Some source control systems might not install the source control plug-in automatically, in which case you will need to install it separately. For further instructions, refer to the documentation of the respective source control system. A plug-in (provider) compatible with the Microsoft Source Code Control Plug-in API is expected to be registered under the following registry entry on your operating system:

```
HKEY_LOCAL_MACHINE\SOFTWARE\SourceCodeControlProvider\InstalledSCCProviders
```


(C) 2015-2021 Altova GmbH

Upon correct installation, the plug-in becomes available automatically in the list of plug-ins available to XMLSpy.

Accessing the source control commands

The commands related to source control are available in the **Project | Source Control** menu.

Resource / Speed issues

Very large source control databases might be introducing a speed/resource penalty when automatically performing background status updates.

You might be able to speed up your system by disabling (or increasing the interval of) the **Perform background status updates every ... seconds** option in the **Source Control** tab accessed through **Tools | Options**.

Note: The **64-bit** version of your Altova application automatically supports any of the supported 32-bit source control programs listed in this documentation. When using a 64-bit Altova application with a 32-bit source control program, the **Perform background status updates every ... seconds** option is automatically grayed-out and cannot be selected.

Differencing with Altova DiffDog

You can configure many source control systems (including Git and TortoiseSVN) so that they use Altova DiffDog as their differencing tool. For more information about DiffDog, see https://www.altova.com/diffdog. For DiffDog documentation, see https://www.altova.com/documentation.html.

24.1 Setting Up Source Control

The mechanism for setting up source control and placing files in a XMLSpy project under source control is as follows:

1. If this hasn't been done already, install the source control system (see Supported Source Control Systems 983) and set up the source control database (repository) to which you wish to save your work.
2. Create a local workspace folder that will contain the working files that you wish to place under source control. The folder that contains all your workspace folders and files is called the local folder, and the path to the local folder is referred to as the local path. This local folder will be bound to a particular folder in the repository.
3. In your Altova application, create an application project folder to which you must add the files you wish to place under source control. This organization of files in an application project is abstract. The files in a project reference physical files saved locally, preferably in one folder (with sub-folders if required) for each project.
4. In the source control system's database (also referred to as source control or repository), a folder is created that is bound to the local folder. This folder (called the bound folder) will replicate the structure of the local folder so that all files to be placed under source control are correctly located hierarchically within the bound folder. The bound folder is usually created when you add a file or an application project to source control for the first time. See the section, Application Project 986, for information about the repository's folder structure.
5. Project files are added to source control using the command **Project | Source Control | Add to Source Control**. When you add a project or a file in a project for the first time to source control, the correct bindings and folder structure will be created in the repository.
6. Source control actions, such as the checking in and out of files, and the removing of files from source control, can be carried out via commands in the **Project | Source Control** submenu. These commands are described in the Project menu subsection 1171 of the User Reference.

Note: If you wish to change the current source control provider, this can be done in one of two ways: (i) via the Source Control options (**Tools | Options | Source Control** 1489), or (ii) in the Change Source Control dialog (**Project | Source Control | Change Source Control**).

(C) 2015-2021 Altova GmbH

24.2 Supported Source Control Systems

The list below shows the Source Control Servers (SCSs) supported by XMLSpy, together with their respective Source Control Clients (SCCs). The list is organized alphabetically by SCS. Note the following:

- Altova has implemented the Microsoft Source Control Plug-in API (versions 1.1, 1.2, and 1.3) in XMLSpy, and has tested support for the listed drivers and revision control systems. It is expected that XMLSpy will continue to support these products if, and when, they are updated.
- Source Code Control clients not listed below, but which implement the Microsoft Source Control Plug-in API, should also work with XMLSpy.

Source Control System	Source Code Control Clients
AccuRev 4.7.0 Windows	AccuBridge for Microsoft SCC 2008.2
Bazaar 1.9 Windows	Aigenta Unified SCC 1.0.6
Borland StarTeam 2008	Borland StarTeam Cross-Platform Client 2008 R2
Codice Software Plastic SCM Professional 2.7.127.10 (Server)	Codice Software Plastic SCM Professional 2.7.127.10 (SCC Plugin)
Collabnet Subversion 1.5.4	• Aigenta Unified SCC 1.0.6 • PushOK SVN SCC 1.5.1.1 • PushOK SVN SCC x64 version 1.6.3.1 • TamTam SVN SCC 1.2.24
ComponentSoftware CS-RCS (PRO) 5.1	ComponentSoftware CS-RCS (PRO) 5.1
Dynamsoft SourceAnywhere for VSS 5.3.2 Standard/Professional Server	Dynamsoft SourceAnywhere for VSS 5.3.2 Client
Dynamsoft SourceAnywhere Hosted	Dynamsoft SourceAnywhere Hosted Client (22252)
Dynamsoft SourceAnywhere Standalone 2.2 Server	Dynamsoft SourceAnywhere Standalone 2.2 Client
Git	PushOK GIT SCC plug-in (see Source Control with Git 1000)
IBM Rational ClearCase 7.0.1 (LT)	IBM Rational ClearCase 7.0.1 (LT)
March-Hare CVSNT 2.5 (2.5.03.2382)	Aigenta Unified SCC 1.0.6
March-Hare CVS Suite 2008	• Jalindi Igloo 1.0.3 • March-Hare CVS Suite Client 2008 (3321) • PushOK CVS SCC NT 2.1.2.5 • PushOK CVS SCC x64 version 2.2.0.4 • TamTam CVS SCC 1.2.40
Mercurial 1.0.2 for Windows	Sergey Antonov HgSCC 1.0.1
Microsoft SourceSafe 2005 with CTP	Microsoft SourceSafe 2005 with CTP

Source Control System	Source Code Control Clients
Microsoft Visual Studio Team System 2008/2010 Team Foundation Server	Microsoft Team Foundation Server 2008/2010 MSSCCI Provider
Perforce 2008 P4S 2008.1	Perforce P4V 2008.1
PureCM Server 2008/3a	PureCM Client 2008/3a
QSC Team Coherence Server 7.2.1.35	QSC Team Coherence Client 7.2.1.35
Reliable Software Code Co-Op 5.1a	Reliable Software Code Co-Op 5.1a
Seapine Surround SCM Client/Server for Windows 2009.0.0	Seapine Surround SCM Client 2009.0.0
Serena Dimensions Express/CM 10.1.3 for Win32 Server	Serena Dimensions 10.1.3 for Win32 Client
Softimage Alienbrain Server 8.1.0.7300	Softimage Alienbrain Essentials/Advanced Client 8.1.0.7300
SourceGear Fortress 1.1.4 Server	SourceGear Fortress 1.1.4 Client
SourceGear SourceOffsite Server 4.2.0	SourceGear SourceOffsite Client 4.2.0 (Windows)
SourceGear Vault 4.1.4 Server	SourceGear Vault 4.1.4 Client
VisualSVN Server 1.6	• Aigenta Unified SCC 1.0.6 • PushOK SVN SCC 1.5.1.1 • PushOK SVN SCC x64 version 1.6.3.1 • TamTam SVN SCC 1.2.24

(C) 2015-2021 Altova GmbH

24.3 Local Workspace Folder

The files you will be working with should be saved in a hierarchy inside a local workspace folder (*see diagram below*).

```
Local Workspace Folder
|
|-- MyProject.spp
|-- QuickStart
|   |-- QuickStart.css
|   |-- QuickStart.xml
|   |-- QuickStart.xsd
|-- Grouping
|   |-- Persons
|   |   |-- Persons.xml
```

The application project file (`.spp` file) typically will be located directly inside the local workspace folder (*see diagram above*).

When one or more files in this (workspace) folder are placed under source control, the local workspace folder's structure is partly or wholly reproduced in the repository. For example, if the file `Persons.xml` from the local folder shown above is placed under source control, then the path to it in the repository will be:

```
[RepositoryFolder]/MyProject/Grouping/Persons/Persons.xml
```

The `MyProject` folder in the repository folder is bound to the local folder. Typically it would be the name of the project, but you could give it any name.

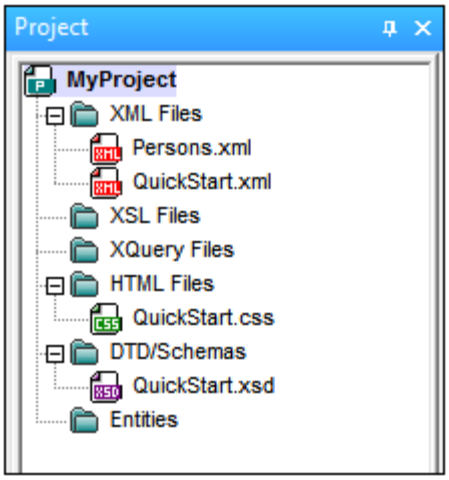
If the entire application project is placed under source control (by selecting the project name in the Projects window and placing it under source control), then the entire local folder structure is recreated in the repository.

Note: Files from outside the local workspace folder can be added to the application project. But whether you can place such a file under source control depends upon the source control system you are using. Some source control systems could have a problem placing a file from outside the local folder into the repository. We therefore recommend that all project files you wish to place under source control be located in the local workspace folder.

24.4 Application Project

Create or load the Altova application project you wish to place under source control. If you wish to place a single file under source control, this file must be included in a project—since source control can only be accessed via a project.

For example, consider a project in Altova's XMLSpy application. The project's properties are saved in a `.spp` file. In the application, the project is displayed in the application's Project window (*see screenshot below*). The project in the screenshot below is named `MyProject` and the project's properties are saved in the file `MyProject.spp`.



You can place the entire project (all files in the project) or only some project files under source control. **Only files that are in the project can be placed under source control.** So you will need to add files to the project before you can place them under source control. The project file (`.spp` file) will automatically be placed under source control as soon as a file from within the project is placed under source control.

The entire project, or one or more project files, is placed under source control via the command **Project | Source Control | Add to Source Control** (*see next section below*).

Note, however, that the folder structure of the repository corresponds not to the project's folder structure (*screenshot above*) but to the structure of the local workspace folder 985 (*see folder diagram below*). In the diagram below, notice that the `MyProject` folder in the repository has a folder structure corresponding to that of the local workspace folder. Note that the bound folder occurs within the repository folder.

```
Local Workspace Folder
|
|-- MyProject.spp
|-- QuickStart
|   |-- QuickStart.css
|   |-- QuickStart.xml
|   |-- QuickStart.xsd
|-- Grouping
|   |-- Persons
|   |   |-- Persons.xml
```

```
Repository
|
|-- MyProject (bound to Local Workspace)
||-- MyProject.spp
||-- QuickStart
||   |-- QuickStart.css
||   |-- QuickStart.xml
||   |-- QuickStart.xsd
||-- Grouping
||   |-- Persons
||   |   |-- Persons.xml
```

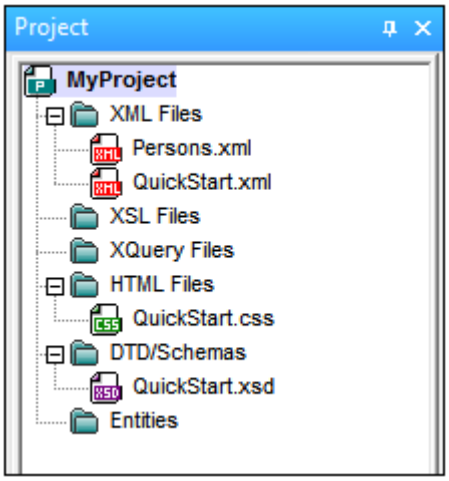
Note: An application project can contain project folders (green) and external folders (yellow). Only files in (green) project folders can be placed under source control. Files in (yellow) external folders cannot be placed under source control.

Note: Files from outside the local workspace folder can be added to the application project. But whether you can place such a file under source control depends upon the source control system you are using. Some source control systems could have a problem placing a file from outside the local folder into the repository. We therefore recommend that all project files you wish to place under source control be located in the local workspace folder.

24.5 Add to Source Control

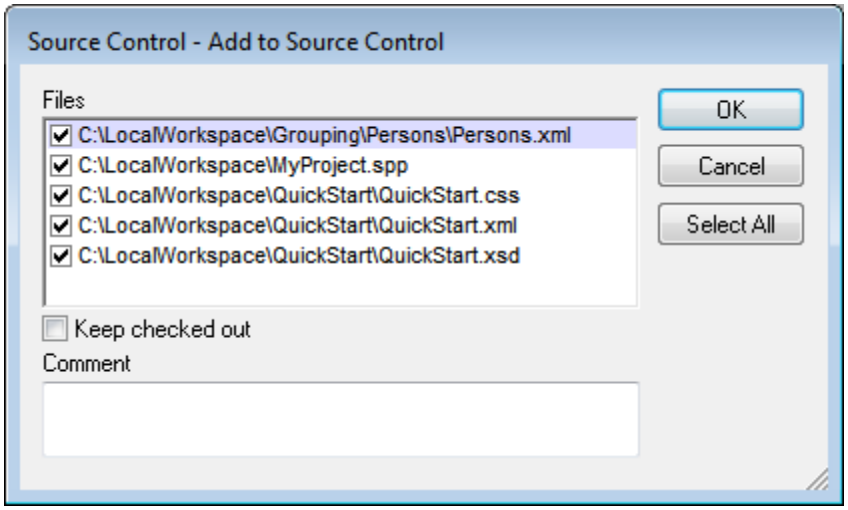
Adding the project to source control will automatically create the correct bindings and repository structure before adding the project file (`.spp` file) or individual files to source control. Add the project to source control as follows.

Select the project in the Project window (`MyProject` in the screenshot below) so that it is highlighted (*as in the screenshot below*). Alternatively select a single file, or select multiple files by clicking them with the **Ctrl** key pressed. Adding a single file to source control will automatically add the project file (`.spp` file) to source control as well.



Next, select the menu command **Project | Source Control | Add to Source Control**. This pops up the connection and configuration dialogs of the currently selected source control system. (You can change the source control system via the Change Source Control dialog (**Project | Source Control | Change Source Control**).)

Follow the source control system's instructions to make the connection and configuration. After this has been completed, all the files selected for addition plus the project file (`.spp` file) are displayed in an Add to Source Control dialog (*screenshot below*). Select the files you wish to add and click **OK**.



(C) 2015-2021 Altova GmbH

The files will be added to the repository and be either checked in or checked out[991] depending on whether the *Keep Checked Out* check box has been checked or not.

Configuration notes

You might be prompted to create a folder in the repository for the project if it has not already been created. If you are, go ahead and create it. The local workspace folder[985] will be bound to this folder created in the repository (*see diagrams below*).

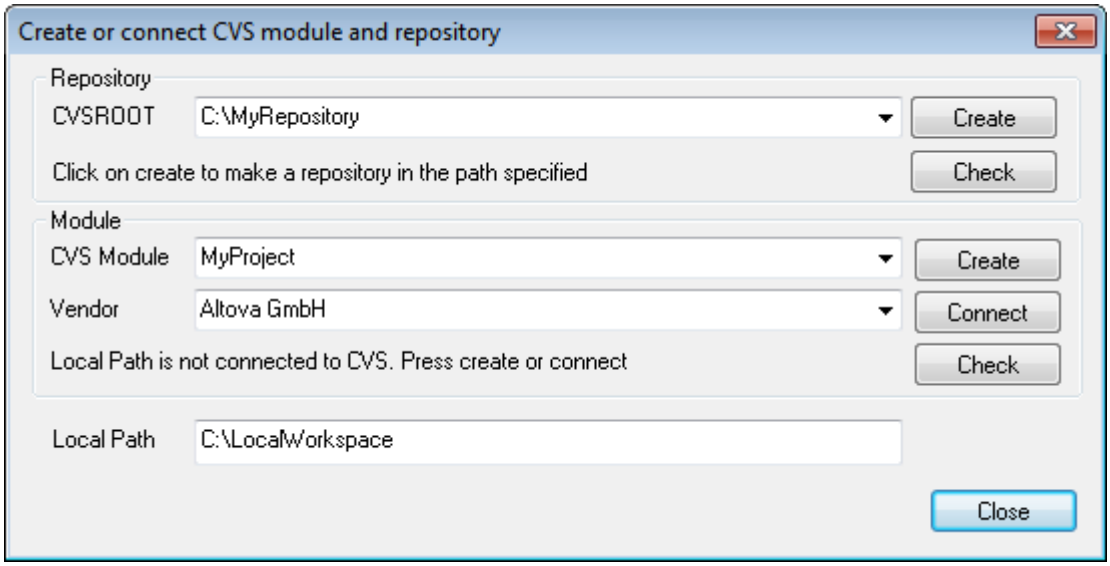
Local Workspace Folder

```
|
|-- MyProject.spp
|-- QuickStart
|   |-- QuickStart.css
|   |-- QuickStart.xml
|   |-- QuickStart.xsd
|-- Grouping
|   |-- Persons
|   |   |-- Persons.xml
```

Repository

```
|
|-- MyProject (bound to Local Workspace)
||-- MyProject.spp
||-- QuickStart
||   |-- QuickStart.css
||   |-- QuickStart.xml
||   |-- QuickStart.xsd
||-- Grouping
||   |-- Persons
||   |   |-- Persons.xml
```

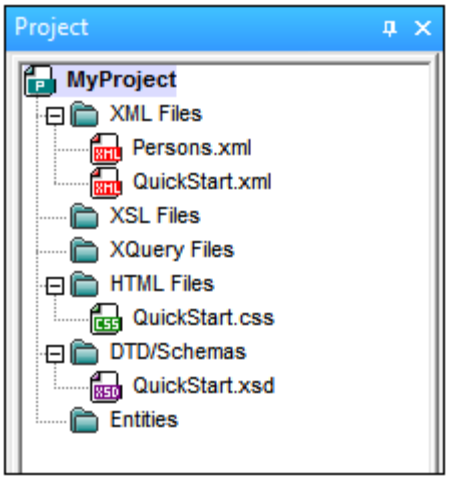
The configuration dialog of Jalindi Igloo is show below. The CVSROOT field is the path to the repository folder.



In the screenshot above, the local path locates the local workspace folder, which corresponds to the CVS module, `MyProject`, and is bound to it.

24.6 Working with Source Control

To work with source control, select the project, a project folder, or a project file in the Project window (*screenshot below*) and then select the command you want in the **Project | Source Control** menu. The **Check In** and **Check Out** commands are available as context menu commands of Project window items.



In this section, we describe the main source control features in detail:

- Add to, Remove from Source Control 990
- Check Out, Check In 991
- Getting Files as Read-Only 993
- Copying and Sharing from Source Control 995
- Changing Source Control 998

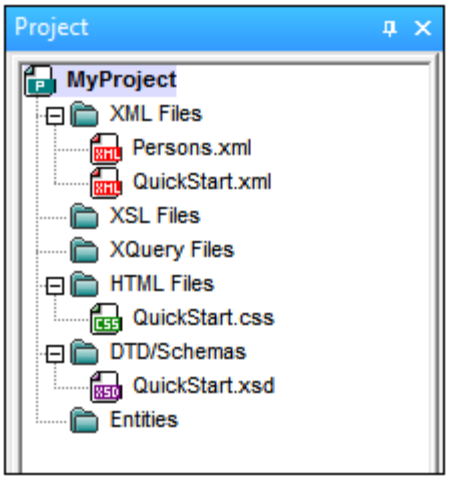
Additional commands in the **Project | Source Control** menu are described in the User Reference section 1171 of the manual. For information specific to a particular source control system, please see the user documentation of that system.

24.6.1 Add to, Remove from Source Control

Adding

After a project has been added to source control, you can place files either singly or in groups under source control. This is also known as adding the files to source control. Select the file in the Project window and then click the command **Project | Source Control | Add to Source Control**. To select multiple files, keep the **Ctrl** key pressed while clicking on the files you wish to add. Running the command on a (green) project folder (*see screenshot below*) adds all files in the folder and its sub-folders to source control.

(C) 2015-2021 Altova GmbH



When files are added to source control, the local folder hierarchy is replicated in the repository 986 (it is not the project folder hierarchy that is replicated). So, if a file is in a sub-folder X levels deep in the local folder, then the file's parent folder and all other ancestor folders are automatically created in the repository.

When the first file from a project is added to source control, the correct bindings are created in the repository and the project file (`.spp` file) is added automatically. For more details, see the section Add to Source Control 988.

Source control symbols

Files and the project folder display certain symbols, the meanings of which are given below.

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

Removing

To remove a file from source control, select the file and click the command **Project | Source Control | Remove from Source Control**. You can also remove: (i) files in a project folder by executing the command on the folder, and (ii) the entire project by executing the command on the project.

24.6.2 Check Out, Check In

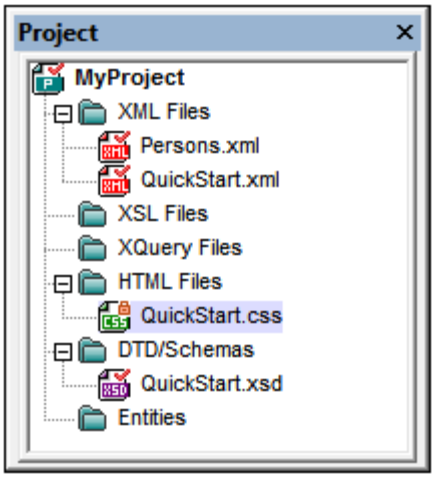
After a project file has been placed under source control, it can be checked out or checked in by selecting the file (in the Project window) and clicking the respective command in the **Project | Source Control** menu: **Check Out** and **Check In**.

When a file is checked out, a copy from the repository is placed in the local folder. A file that is checked out can be edited. If a file that is under source control is not checked out, it cannot be edited. After a file has been edited, the changes can be saved to the repository by checking in the file. Even if the file is not saved in the application, checking it in will save the changes to the repository. Whether a file is checked out or not is indicated with a tick or lock symbol in its Project window icon.

Files and the project folder display certain symbols, the meanings of which are given below.

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	Checked out locally. Can be edited and checked-in.

Selecting the project or a folder within the project selects all files in the selected object. To select multiple objects (files and folders), press the **Ctrl** key while clicking the objects. The screenshot below shows a project that has been checked out. The file `QuickStart.css` has subsequently been checked in.



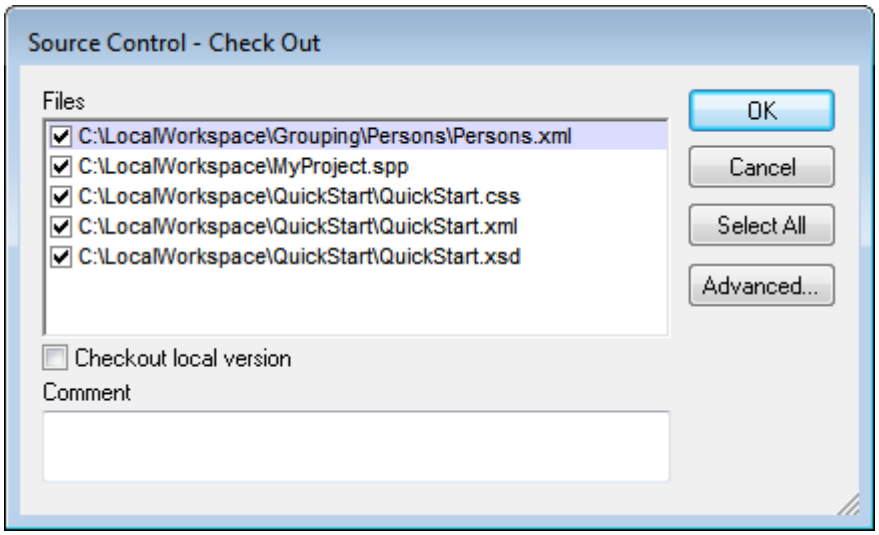
Saving and rejecting editing changes

Note that, when checking in a file, you can choose to leave the file checked out. What this does is save editing changes to the repository while continuing to keep the file checked out, which is useful if you wish to periodically save editing changes to the repository and then continue editing.

If you have checked out a file and made editing changes, and then wish to reject these changes, you can revert to the document version saved in the repository by selecting the command **Project | Source Control | Undo Check Out**.

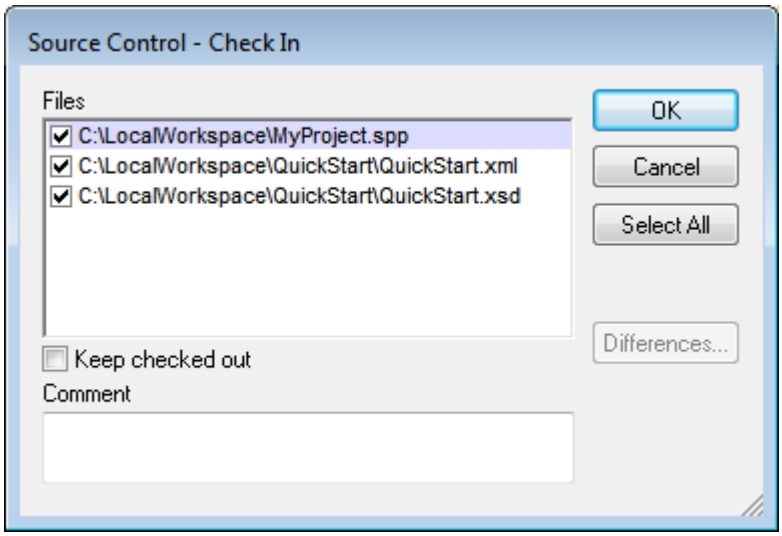
Checking out

The Check Out dialog (*screenshot below*) allows you: (i) to select the files to check out, and (ii) to select whether the repository version or the local version should be checked out.



Checking in

The Check In dialog (*screenshot below*) allows you: (i) to select the files to check in, and (ii) if you wish, to keep the file checked out.

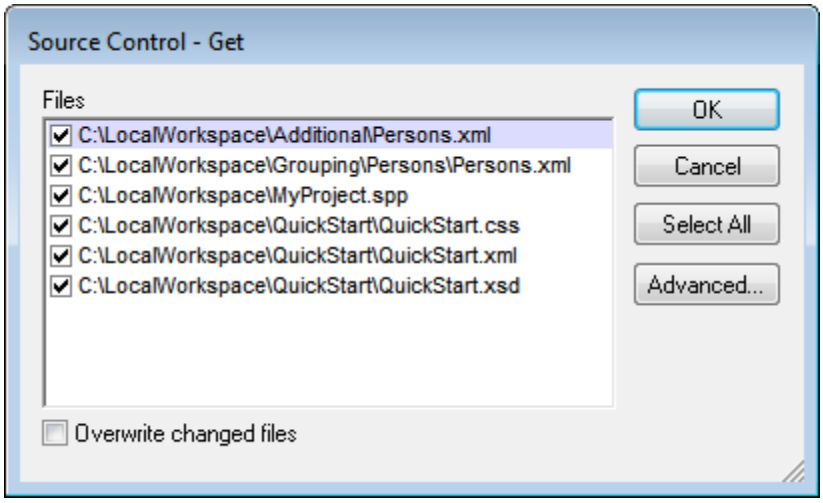


Note: In both dialogs (Check Out and Check In), multiple files appear if the selected object (project or project folder/s) contain multiple files.

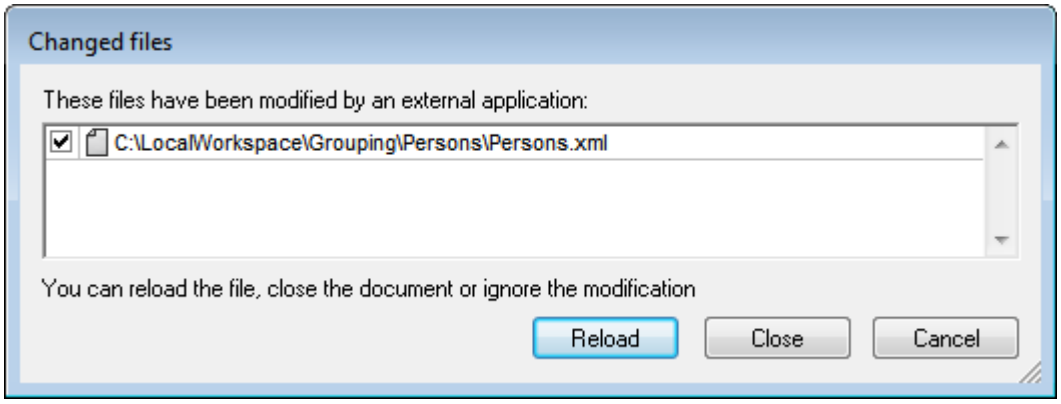
24.6.3 Getting Files as Read-Only

The **Get** command (in the **Project | Source Control** menu) retrieves files from the repository as read-only files. (To be able to edit a file, you must check it out 991.) The Get dialog lists the files in the object (project or folder) on which the **Get** command was executed (*see screenshot below*). You can select the files to retrieve by checking them in the Get dialog list.

Note: The **Get Folders** command allows you to select individual sub-folders in the repository if this is allowed by your source control system, .

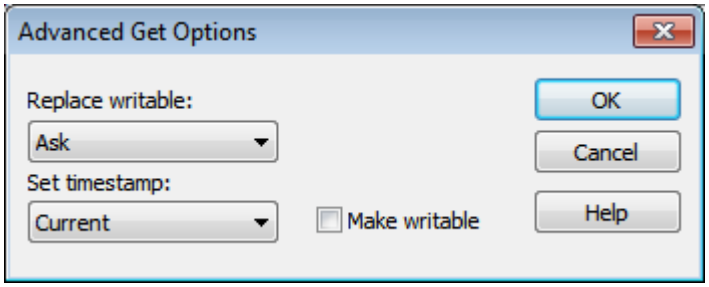


You can choose to overwrite changed checked-out files by checking this option at the bottom of the Get dialog. On clicking **OK**, the files will be overwritten. If any of the overwritten files is currently open, a dialog pops up (*screenshot below*) asking whether you wish to reload the file/s (**Reload** button), close the file/s (**Close**), or retain the current view of the file (**Cancel**).



Advanced Get Options

The Advanced Get Options dialog (*screenshot below*) is accessed via the **Advanced** button in the Get dialog (*see first screenshot in this section*).



(C) 2015-2021 Altova GmbH

Here you can set options for (i) replacing writable files that are checked out, (ii) the timestamp, and (iii) whether the read-only property of the retrieved file should be changed so that it will be writable.

Get latest version

The **Get Latest Version** command (in the **Project | Source Control** menu) retrieves and places the latest source control version of the selected file(s) in the working directory. The files are retrieved as read-only and are not checked out. This command works like the **Get** command (*see above*), but does not display the Get dialog.

If the selected files are currently checked out, then the action taken will depend on how your source control system handles such a situation. Typically, the source control system will ask whether you wish to replace, merge with, or leave the checked-out file as it is.

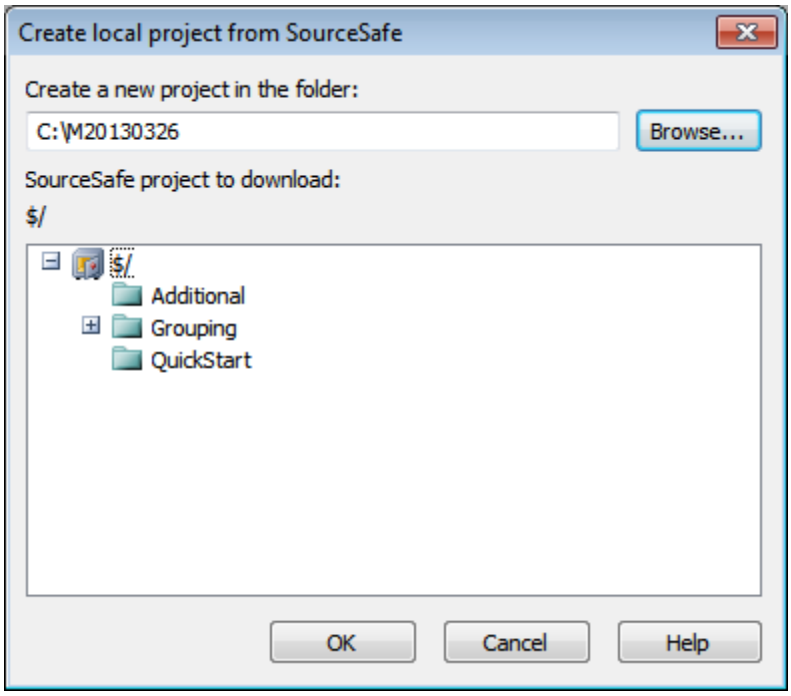
Note: This command is recursive when performed on a folder, that is, it affects all files below the current one in the folder hierarchy.

24.6.4 Copying and Sharing from Source Control

The **Open from Source Control** command creates a new application project from a project under source control.

Create the new project as follows:

1. Depending on the source control system used, it might be necessary, before you create a new project from source control, to make sure that no file from the source-controlled project is checked out.
2. No project need be open in the application, but can be.
3. Select the command **Project | Source Control | Open from Source Control**.
4. The source control system that is currently set will pop up its verification and connection dialogs. Make the connection to the bound folder in the repository 986 that you want to copy.
5. In the dialog that pops up (*screenshot below*), browse for the local folder to which the contents of the bound folder in the repository (that you have just connected to) must be copied. In the screenshot below the bound folder is called `MyProject` and is represented by the `$` sign; the local folder is `C:\M20130326`.



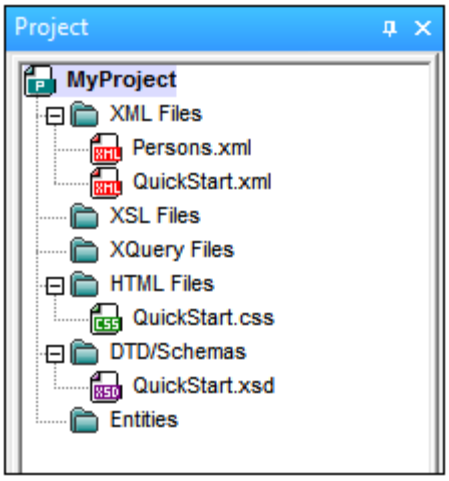
6. Click **OK**. The contents of the bound folder (`MyProject`) will be copied to the local folder `C:\M20130326`., and a dialog pops up asking you to select the project file (`.spp` file) that is to be created as the new project.
7. Select the `.spp` file that will have been copied to the local folder. In our example, this will be `MyProject.spp` located in the `C:\M20130326` folder. A new project named `MyProject` will be created in the application and will be displayed in the Project window. The project's files will be in the folder `C:\M20130326`.

Sharing from source control

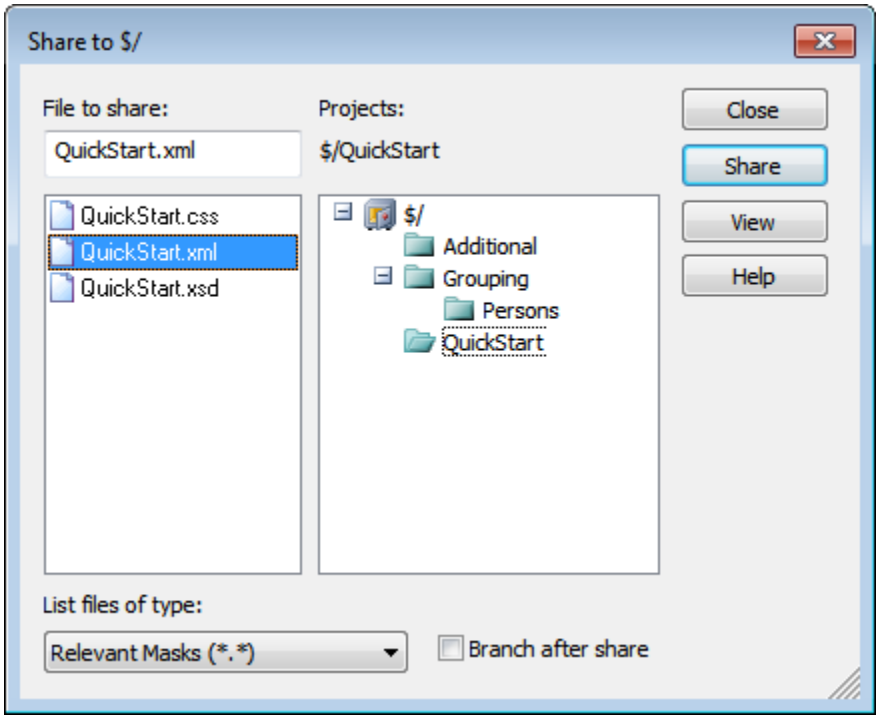
The **Share from Source Control** command is supported when the source control system being used supports shares. You can share a file, so that it is available at multiple local locations. A change made to one of these local files will be reflected in all the other "shared" versions.

In the application's Project window first select the project (*highlighted in the screenshot below*). Then click the **Share from Source Control**.

(C) 2015-2021 Altova GmbH

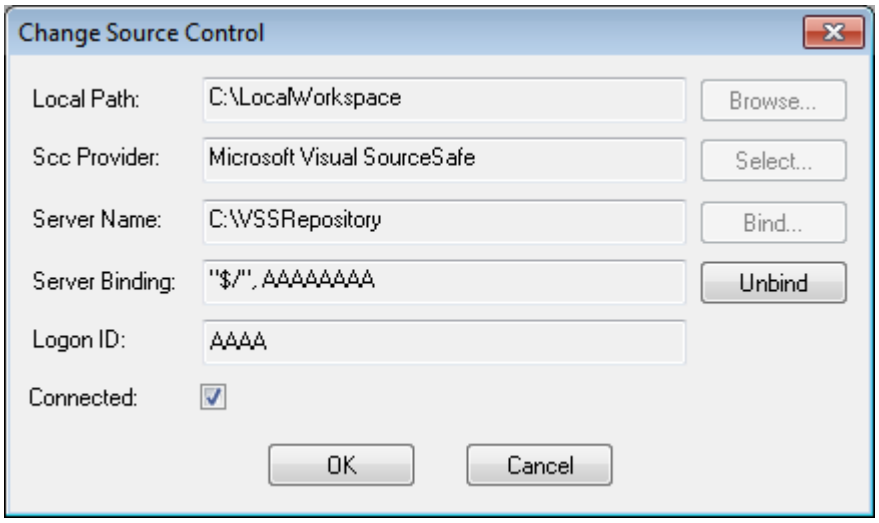


The Share To [Folder] dialog (*screenshot below*) pops up.



To select the files to share, first choose, in the project tree in the right-hand pane of the dialog (*see screenshot above*), the folder in which the files are. The files in the chosen folder are displayed in the left-hand pane. Select the file you wish to share (multiple files by pressing the **Ctrl** key and clicking the files you want to share). The selected file/s will be displayed in the *Files to Share* text box (*at top left*). The files disappear from the left hand pane. Click **Share** and then **Close** to copy the selected file/s to the local share folder. When you click **Close**, the files to share will be copied to the selected local location.

The share folder is noted in the name of the Share to [Folder] dialog. In the screenshot above it is the local folder (since the `$` sign is the folder in the repository to which the local folder is bound). You can see and set the share folder in the Change Source Control dialog (*screenshot below*, **Change Source Control**) by changing the local path and server binding.



For more details about sharing using your source control system, see the source control system's user documentation.

24.6.5 Changing Source Control

Source control settings can be changed via two commands in the **Project | Source Control** menu:

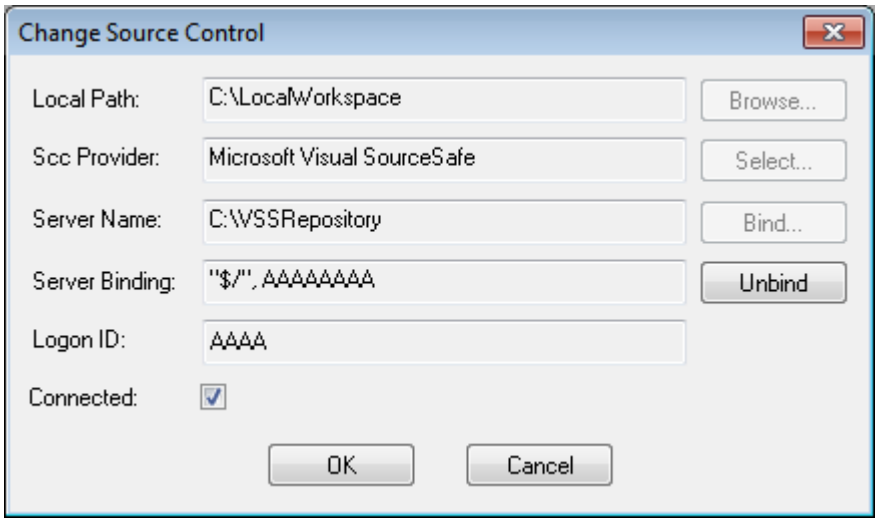
- **Source Control Manager**, which opens the source control system application and allows you to set up databases and configure bindings.
- **Change Source Control**, which pops up the Change Source Control dialog, in which you can change the source control system being used by the Altova application and the current binding. This dialog is described below.

The current binding is what the active application project will use to connect to the source control database. The current binding is correct when the application project file (`.spp` file) is in the local folder and the bound folder in the repository is where this project's files are stored. Typically the bound folder and its sub-structure will correspond with the local workspace folder and its sub-structure.

In the Change Source Control dialog (*screenshot below*), you can change the source control system (*SCC Provider*), the local folder (*Local Path*), and the repository binding (*Server Name* and *Server Binding*).

Only after undoing the current binding can the settings be changed. Undo the current binding with the **Unbind** button. All the settings are now editable.

(C) 2015-2021 Altova GmbH



Change source control settings as follows:

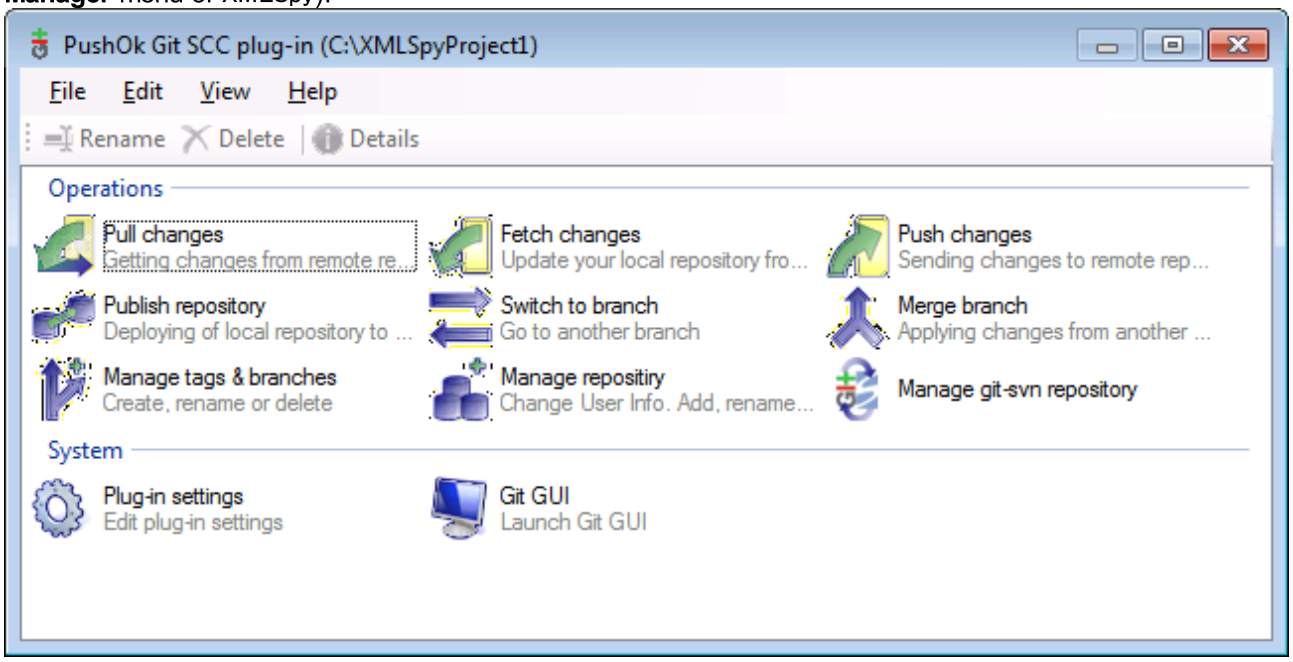
1. Use the **Browse** button to browse for the local folder and the **Select** button to select from among the installed source control systems.
2. After doing this you can bind the local folder to a repository database. Click the **Bind** button to do this. This pops up the connection dialog of your source control system.
3. If you have entered a *Logon ID*, this will be passed to the source control system; otherwise you might have to enter your logon details in the connection dialog.
4. Select the database in the repository that you wish to bind to this local folder. This setting might be spread over more than one dialog.
5. After the setting has been created, click **OK** in the Change Source Control dialog.

24.7 Source Control with Git

Support for Git as a source control system in XMLSpy is available through a third-party plug-in called **GIT SCC plug-in** (http://www.pushok.com/software/git.html).

> At the time when this documentation is written, the **GIT SCC plug-in** is available for experimental use. Registration with the plug-in publisher is required in order to use the plug-in.

The GIT SCC plug-in enables you to work with a Git repository using the commands available in the **Project | Source Control** menu of XMLSpy. Note that the commands in the **Project | Source Control** menu of XMLSpy are provided by the Microsoft Source Control Plug-in API (MSSCCI API), which uses a design philosophy different from Git. As a result, the plug-in essentially mediates between "Visual Source Safe"-like functionality and Git functionality. On one hand, this means that a command such as **Get latest version** may not be applicable with Git. On the other hand, there are new Git-specific actions, which are available in the "Source Control Manager" dialog box provided by the plug-in (under the **Project | Source Control | Source Control Manager** menu of XMLSpy).



The Source Control Manager dialog box

Other commands that you will likely need to use frequently are available directly under the **Project | Source Control** menu.

The following sections describe the initial configuration of the plug-in, as well as the basic workflow:

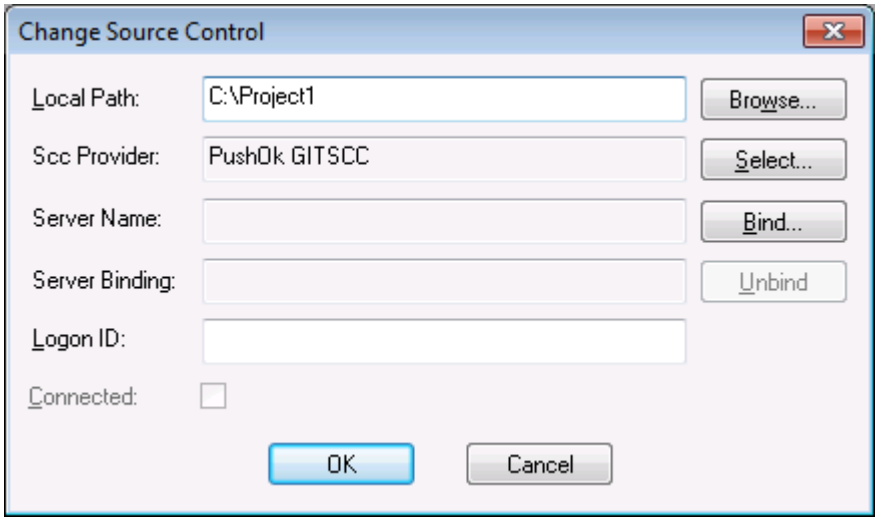
- Enabling Git Source Control with GIT SCC Plug-in 1001
- Adding a Project to Git Source Control 1001
- Cloning a Project from Git Source Control 1002

(C) 2015-2021 Altova GmbH

24.7.1 Enabling Git Source Control with GIT SCC Plug-in

To enable Git source control with XMLSpy, the third-party **PushOK GIT SCC plug-in** must be installed, registered, and selected as source control provider, as follows:

1. Download the plug-in installation file from the publisher's website (http://www.pushok.com), run it, and follow the installation steps.
2. On the **Project** menu of XMLSpy, click **Change Source Control**, and make sure **PushOk GITSCC** is selected as source control provider. If you do not see **Push Ok GITSCC** in the list of providers, it is likely that the installation of the plug-in was not successful. In this case, check the publisher's documentation for a solution.



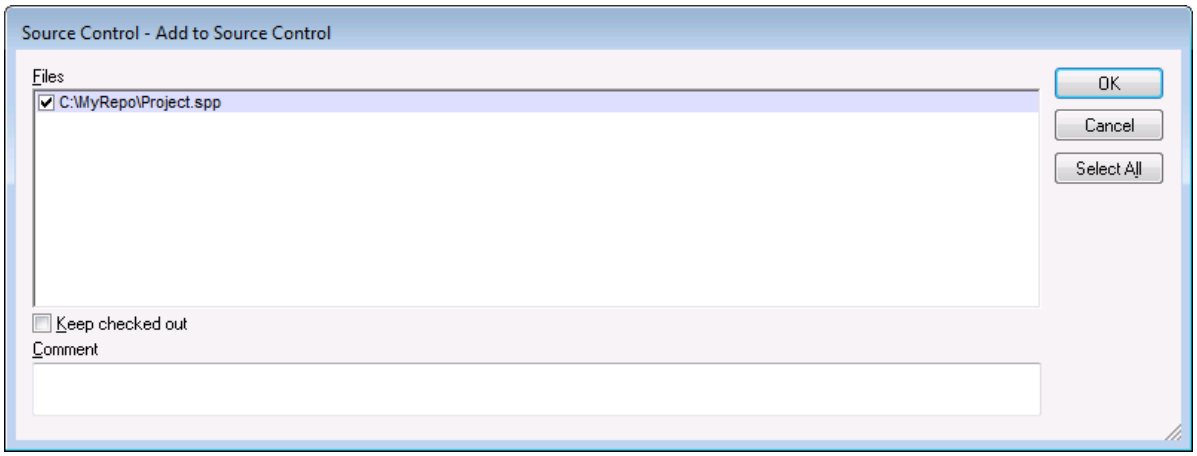
3. When a dialog box prompts you to register the plug-in, click **Registration** and follow the wizard steps to complete the registration process.

24.7.2 Adding a Project to Git Source Control

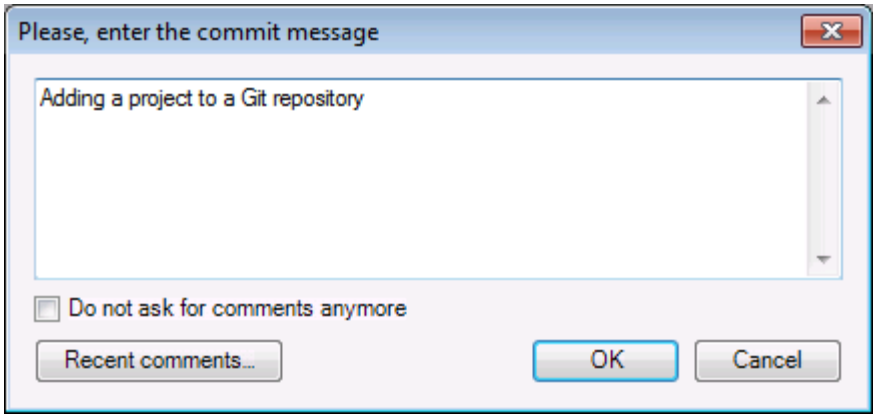
You can save XMLSpy projects as Git repositories. The structure of files or folders that you add to the project would then correspond to the structure of the Git repository.

To add a project to Git source control:

1. Make sure that **PushOK GIT SCC Plug-in** is set as source control provider (see Enabling Git Source Control with GIT SCC Plug-in 1001).
2. Create a new project using the menu command **Project | Create Project**.
3. Save the project to a local folder, for example `C:\MyRepo\Project.spp`
4. On the **Project** menu, under **Source Control**, click **Add to Source Control**.

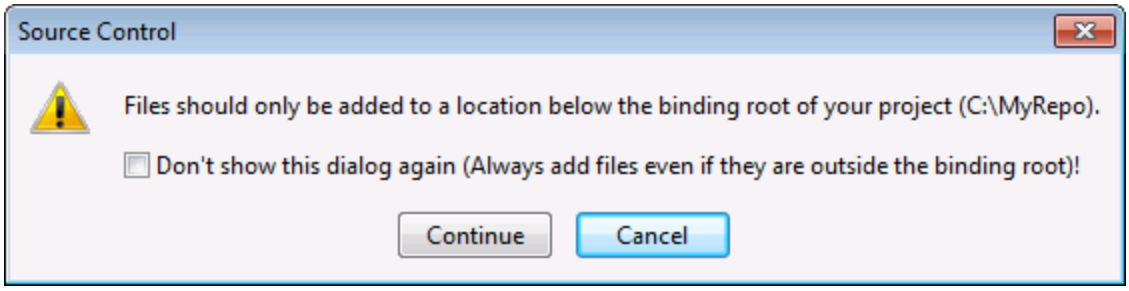


5. Click **OK**.



6. Enter the text of your commit message, and click **OK**.

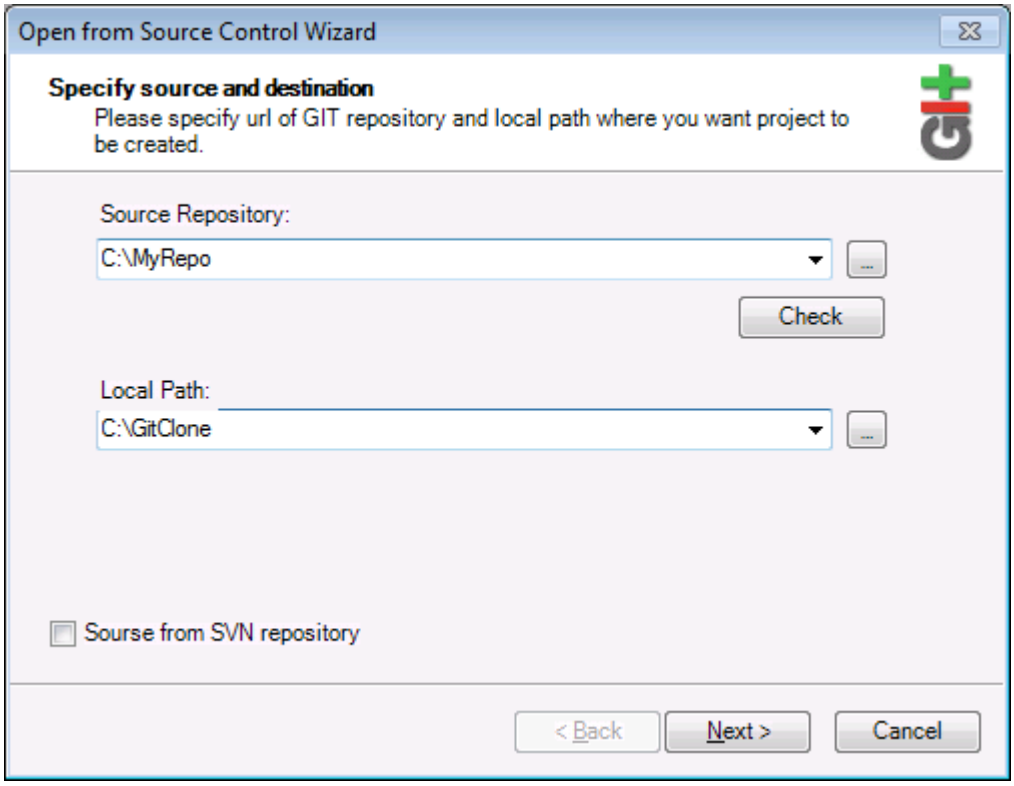
You can now start adding files and folders to your project. Note that all project files and folders must be under the root folder of the project. For example, if the project was created in the `C:\MyRepo` folder , then only files under `C:\MyRepo` should be added to the project. Otherwise, if you attempt to add to your project files that are outside the project root folder, a warning message is displayed:



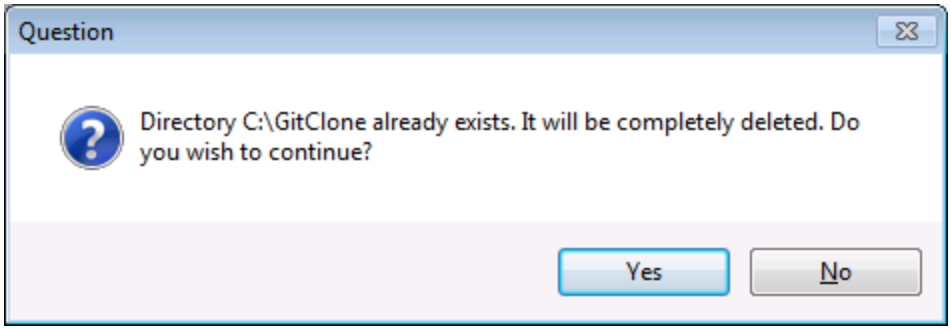
24.7.3 Cloning a Project from Git Source Control

Projects that have been previously added to Git source control (see Adding a Project to Git Source Control 1001) can be opened from the Git repository as follows:

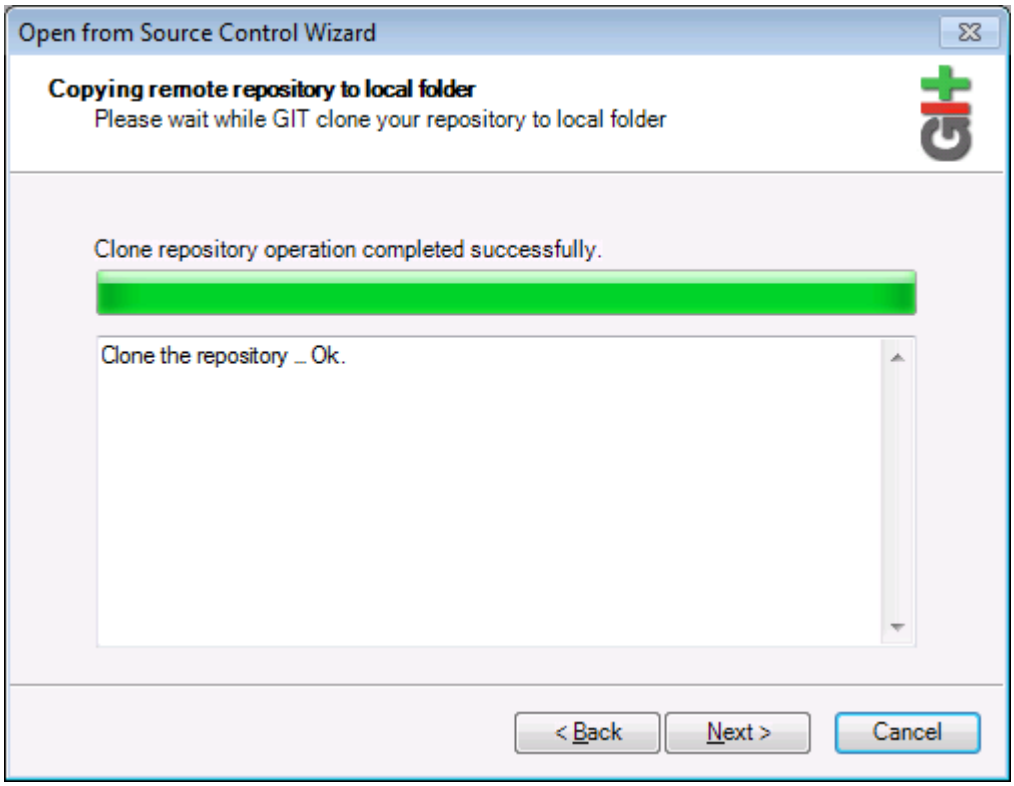
1. Make sure that **PushOK GIT SCC Plug-in** is set as source control provider (see Enabling Git Source Control with GIT SCC Plug-in 1001).
2. On the **Project** menu, click **Source Control | Open from Source Control**.
3. Enter the path or the URL of the source repository. Click **Check** to verify the validity of the path or URL.



4. Under **Local Path**, enter the path to local folder where you want the project to be created, and click **Next**. If the local folder exists (even if it is empty), the following dialog box opens:



5. Click **Yes** to confirm, and then click **Next**.



6. Follow the remaining wizard steps, as required by your specific case.
7. When the wizard completes, a Browse dialog box appears, asking you to open the XMLSpy Project (*.spp) file. Select the project file to load the project contents into XMLSpy.

(C) 2015-2021 Altova GmbH

25 Visual Studio 内の XMLSpy

XMLSpy はMicrosoft Visual Studio IDE のバージョン2010/2012/2013/2015/2017/2019 に統合することができます。この機能により、高度な XML 編集機能を、Visual Studio の高度な開発環境内で使用することができるようになります。

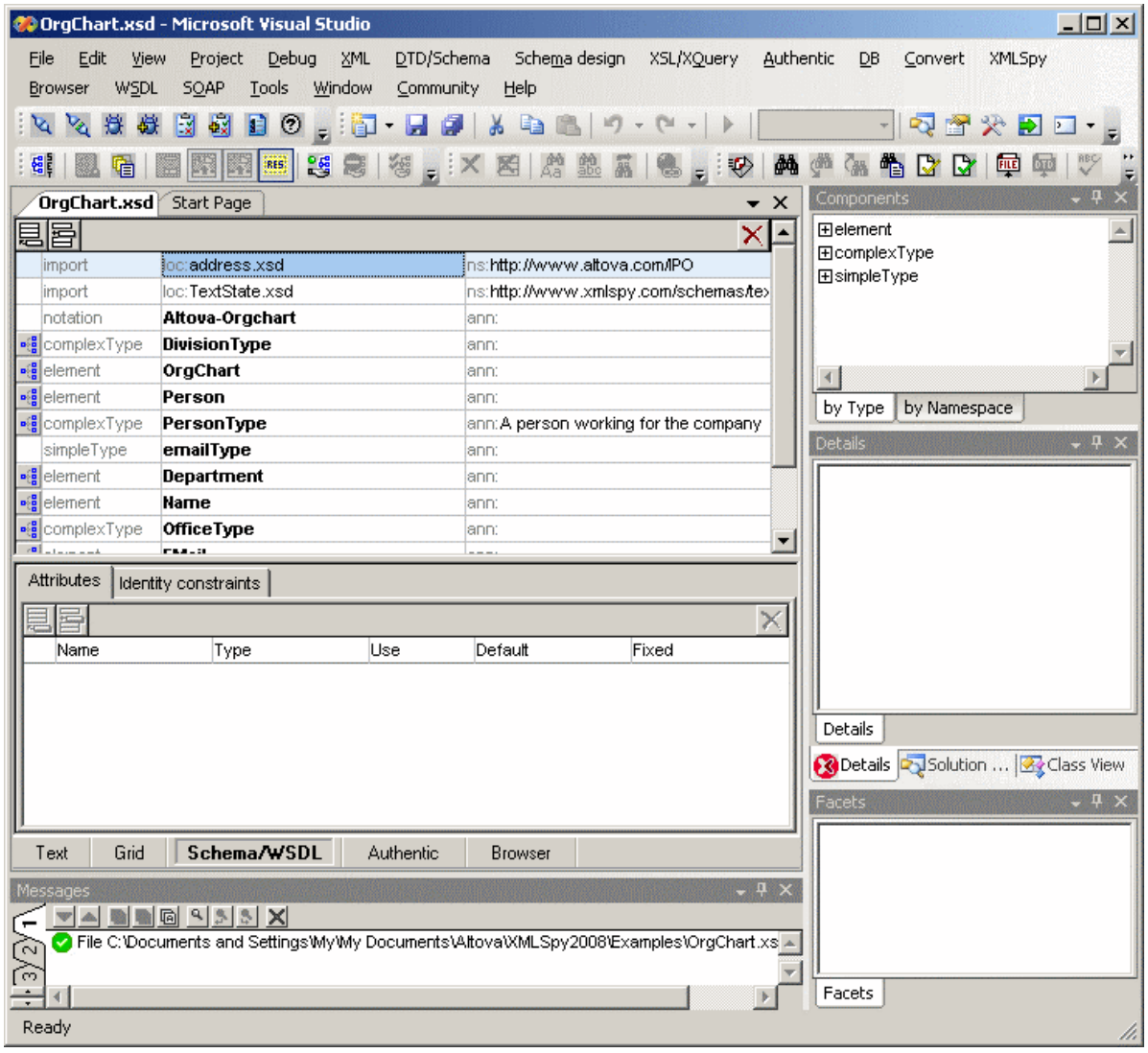
このセクションでは、以下のように構成されます:

- 大まかなインストールの流れ[1006]と、XMLSpy プラグインを Visual Studio へ統合
- Visual Studio バージョンとスタンドアロンバージョンの違い[1007]
- Visual Studio 内のXMLSpy デバッガー[1009]

25.1 XMLSpy プラグインのインストール

Visual Studio で使えるXMLSpy プラグインをインストールするには、以下の手順を踏みます:

1. Microsoft Visual Studio 2010/2012/2013/2015/2017/2019 のインストール。
2. XMLSpy (Enterprise またはProfessional Edition) のインストール
3. XMLSpy のMicrosoft Visual Studio Integration Package をダウンロードして実行。パッケージはwww.altova.com にあるXMLSpy (Enterprise またはProfessional Edition)ダウンロードページにてご利用になれます。(注意: XMLSpy バージョンに対応するIntegration Package を使用する必要があります(現在のバージョンは2021 です)。)



Integration Package をインストールすると、Visual Studio 環境からXMLSpy を使用することができるようになります。

25.2 XMLSpy スタンドアロンとの違い

このセクションではVisual Studio バージョンとスタンドアロンバージョンのXMLSpy の違いについて説明します。まずはVisual Studio バージョンでサポートされていない機能について、そしてスタンドアロンバージョンのVisual Studio 間で動作が異なる点について説明します。

- Visual Studio ではサポートされない機能 1007
- Visual Studio で追加されたXMLSpy メニュー 1007
- 入力ヘルパー(Visual Studio のツールウィンドウ) 1007
- 異なるコマンドによる同等の機能 1007
- Visual Studio コマンドとして使えるXMLSpy コマンド 1008

Visual Studio ではサポートされない機能

Visual Studio では以下のXMLSpy の機能がご利用になれません:

- スクリプティング環境(「**ツール**| XMLSpy **オプション**」メニューオプションの「**スクリプト**」タブ)は現在サポートされていません。
- メニューオプションの「**ツール**| **ツール**」の「**ビュー**」タブにある、ブラウザービューを別のウィンドウに表示するオプションがサポートされていません。つまりテキストビューとテキストビューは常に同じウィンドウで表示されます。
- Authentic View 526 の状態アイコンがサポートされていません。
- 全てのソース管理機能がサポートされていません。
- (スタンドアロンバージョンでは「**ツール**」メニューオプションから利用できる)全ての比較機能がサポートされていません。

Visual Studio で追加されたXMLSpy メニュー

以下のコマンドはVisual Studio バージョンのXMLSpy だけでご利用になれます:

- **表示**| XMLSpy **ツールウィンドウ**
- **表示**| XMLSpy **表示**
- **XMLSpy** (グローバルリソースメニューアイテムが含まれます)
- ツール| XMLSPY **オプション**

入力ヘルパー(Visual Studio のツールウィンドウ)

XMLSpy の入力ヘルパーはVisual Studio のツールウィンドウとして使用することができます。入力ヘルパーに関して、以下の点に注意してください。入力ヘルパーとXMLSpy のGUI に関しては、イントロダクション 113 を参照ください。

- 入力ヘルパーウィンドウは開発環境内のどのような場所にでもドラッグすることができます。
- 入力ヘルパータブを右クリックすることで、インターフェースを更にカスタマイズすることができます。入力ヘルパーのオプションは: フローティング、ドッキング可能、タブ付きドキュメント、自動的に隠す、です。

異なるコマンドによる同等の機能

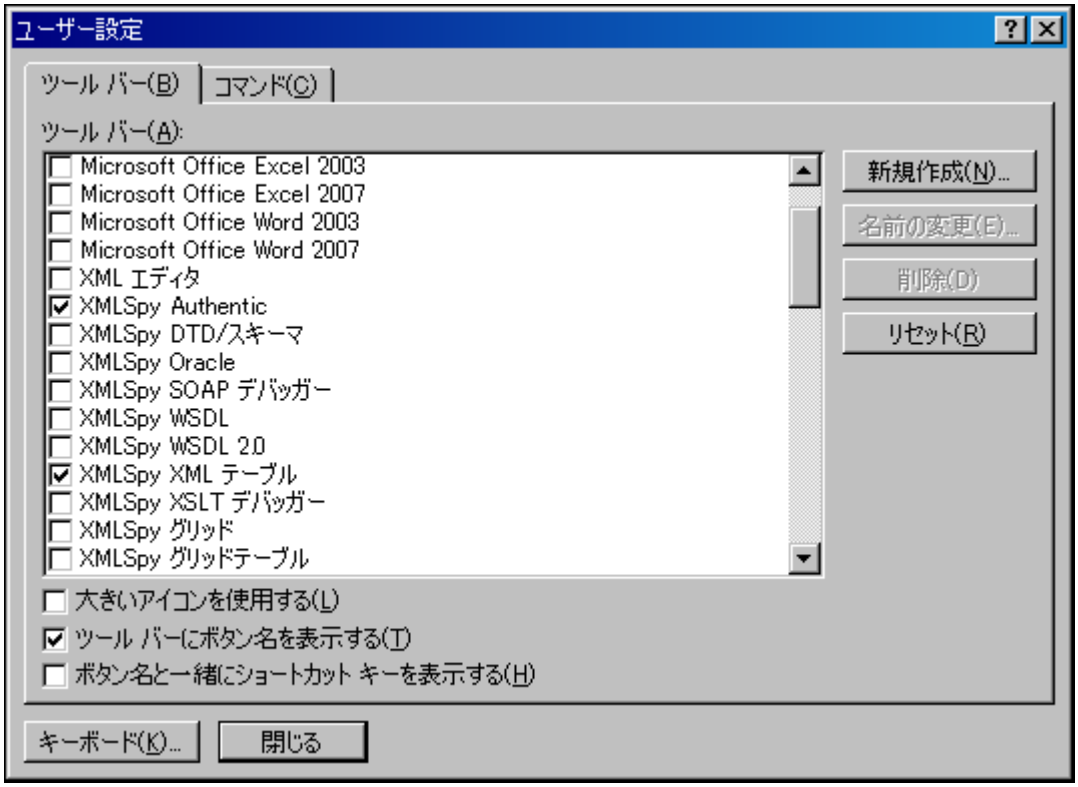
以下の表に示されるように、XMLSpy の機能によっては、Visual Studio にて異なるコマンド名で表示されます。

XMLSpy	*Visual Studio*	*機能*
ファイル| 開く| URL にスイッチ	ファイル| 開く| W eb サイト	URL からファイルを開く
ファイル| 名前を付けて保存| URL にスイッチ	ファイル| XMLSpy ファイルを URL に保存	ファイルを URL に保存

Visual Studio コマンドとして使える XMLSpy コマンド

いくつかのXMLSpy コマンドはVisual Studio コマンドとしてVisual Studio GUI に表示されます:

- *元に戻す、やり直し* : これらのVisual Studio コマンドは、Visual Studio 開発環境全てに対して適用されます。
- *プロジェクト:* XMLSpy プロジェクトはVisual Studio プロジェクトとして扱われます。
- *ツールバーのカスタマイズ、コマンドのカスタマイズ*: (以下のスクリーンショットにある) ユーザー設定ダイアログ(「**ツール | ユーザー設定**」) 内のツールバーならびにコマンドタブにはVisual Studio とXMLSpy のコマンドが含まれます。

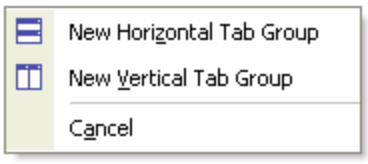


- *表示* : 表示メニュー以下のXMLSpy ツールウィンドウとXMLSpy ビューの2つのコマンドには、入力ヘルパーや他のサイドバー、編集ビュー間の切り替え、そして特定の入力ガイドの有効/無効を切り替えるオプションが含まれています。
- *XMLSpy ヘルプ:* このXMLSpy メニューはVisual Studio 「**ヘルプ**」 メニューのサブメニューに表示されます。

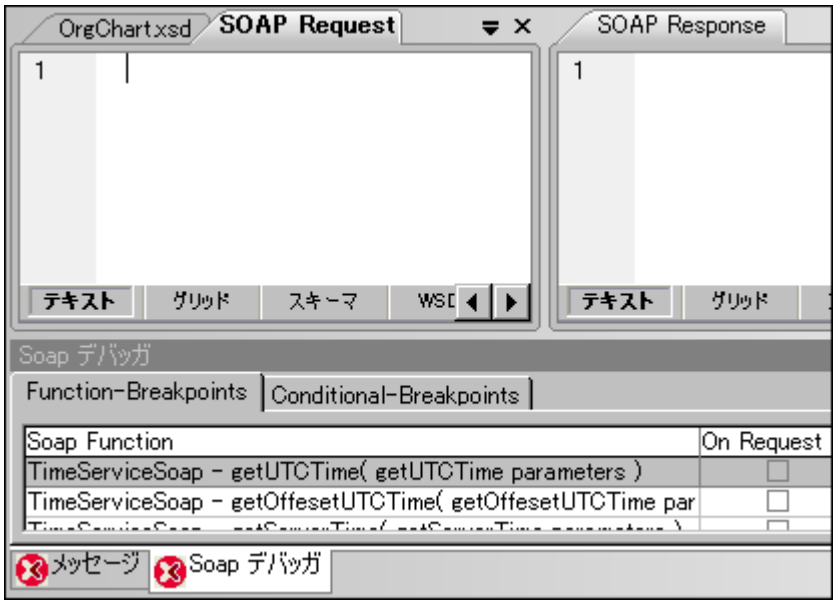
25.3 Visual Studio での XMLSpy デバッガー

XMLSpy にはXSLT/XQuery デバッガー(Enterprise ならびにProfessional Edition)と、SOAP デバッガー(Enterprise Edition)が搭載されています。デバッガーでは(例えばXML、XSLT、XSLT 出力ファイルなど)複数のファイルが表示されますが、これらのファイルはVisual Studio にて1つにタブグループ化されます。Visual Studio では新たにタブグループを作成して、デバッグを見やすく、処理の経過をより簡単にたどることができます:

1. 単一のタブグループから分離したいタブをクリックします。そのタブを、現在アクティブなタブのどこかにドラッグします。この操作により、新たに作成するタブグループの種類を決定するポップアップメニューが表示されます。



2. 「垂直タブグループの新規作成」を選択すると、選択されたタブのみが含まれるタブグループが新たに作成されます(以下のスクリーンショットを参照)。



26 Eclipse 内の XMLSpy

Eclipse はオープンソースのフレームワークで、様々なアプリケーション開発環境をプラグインという形で統合します。

Eclipse 用のXMLSpy プラグインを使えば、Eclipse 2020-12, 2020-09, 2020-06, 2020-03 プラットフォームからXMLSpy の機能にアクセスすることができるようになります。プラグインはW indows プラットフォームにてご利用になれます。このセクションではEclipse 用のXMLSpy プラグインのインストール方法[1011]と、XMLSpy パースペクティブ[1013]ならびにXMLSpy デバッガーパースペクティブ[1016]の設定方法について説明します。これらの操作を行うと、XMLSpy GUI とXMLSpy メニューコマンドをEclipse GUI から使用することができるようになります。

注意: スタンドアロンバージョンで使用することのできるソース管理機能はEclipse バージョンにてご利用になれません。

(C) 2015-2021 Altova GmbH

26.1 Eclipse 用の XMLSpy プラグインをインストール

必要条件

- Eclipse 2020-12, 2020-09, 2020-06, 2020-03 (http://www.eclipse.org)。
- the 64-ビットプラットフォームのための Java Runtime Environment (JRE) または Java Development Kit (JDK)
- XMLSpy Enterprise または Professional Edition 64-ビット
- XMLSpy Integration Package 64-ビット を以下でダウンロードすることができます https://www.altova.com/ja/components/download。下記の通り Eclipse 統合をインストールの一部として行うことができます。

メモ　上記の必要条件は 64-ビットプラットフォームを必要としています。古い Eclipse 32-ビットプラットフォームとの統合は作動する場合がありますが、サポートされていません。

Eclipse のための XMLSpy プラグインをインストールする方法

Eclipse のための XMLSpy プラグインの統合を以下の方法の一つで行うことができます:

1. 自動的に XMLSpy Integration Package のインストール中に統合することができます (これは奨励されるオプションです)。
2. XMLSpy Integration Package を最初にインストールし、Eclipse からプラグインを手動で統合する。

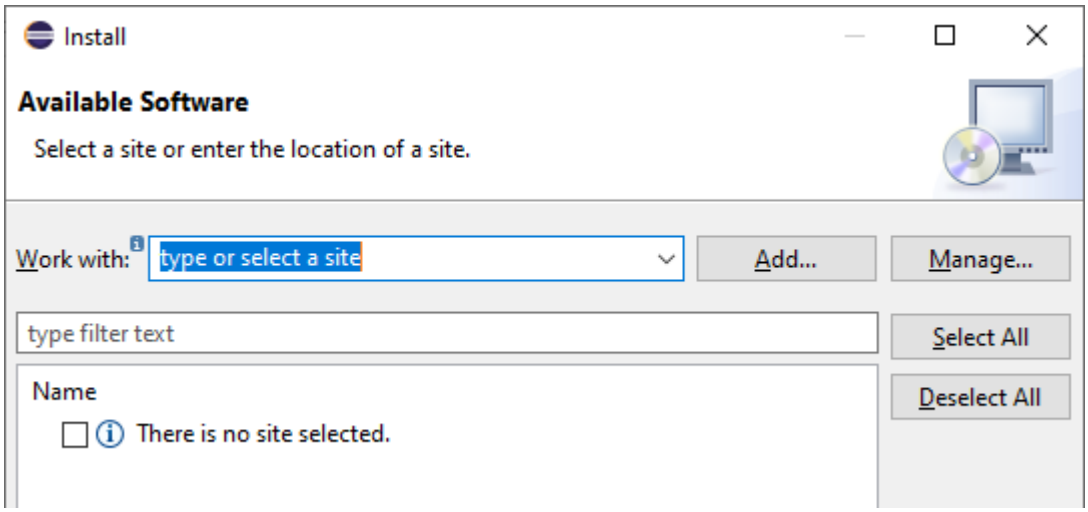
Eclipse のための XMLSpy プラグインを統合する方法:

1. インストールウィザードを開始するために XMLSpy Integration Package を実行します。
2. プロンプトされると *Eclipse プラグインのインストール* オプションを選択し [次へ] をクリックします。
3. プロンプトされると XMLSpy プラグインの Eclipse への統合方法を選択します。以下の 1 つを行ってください:
   - プラグインインストールを自動的に完了する場合 *このウィザードに Altova XMLSpy プラグインを Eclipse に統合する* を選択し、Eclipse 実行可能ファイル (`eclipse.exe`) がロードされている場所を参照します。
   - Eclipse 内で後にプラグインのインストールを完了する場合、clear the *このウィザード...* チェックボックスをクリアします。
4. [次] をクリックすることでインストールを完了します。自動統合を選択する場合 XMLSpy パースペクティブとメニューが Eclipse を次回起動する際に Eclipse 内で使用可能になります。

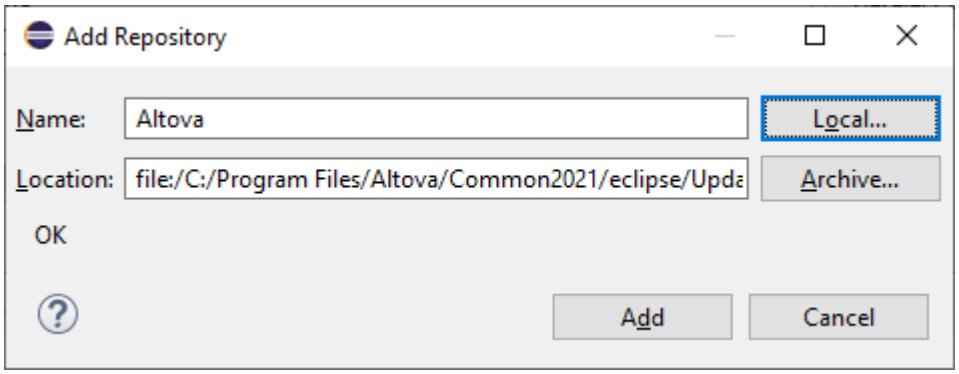
メモ　XMLSpy Integration Package をインストールまたはアンインストールする場合 Eclipse を閉じる必要があります。

Eclipse のための XMLSpy プラグインを手動で統合する方法

1. Eclipse にてメニューオプションから [**ヘルプ | 新規ソフトウェアのインストール**] を選択します。
2. [インストール] ダイアログボックス内で [**追加**] をクリックします。



3. [レポジトリの追加] ダイアログボックス 内で [ローカル] をクリックします。フォルダー `C:\Program Files\Altova\Common2021\eclipse\UpdateSite` を参照して選択します。("Altova" などの) サイトの名前を与えます。



4. ステップ 2 から 3 を繰り返します、今度は、フォルダー `C:\Program Files\Altova\XMLSpy2021\eclipse\UpdateSite` を選択し、XMLSpy などの名前を与えます。
5. インストールダイアログボックスで *ローカルのサイトのみ* を選択します。次に [Altova category] フォルダーを選択し [次へ] をクリックします。
6. インストールされるアイテムをレビューし [次へ] をクリックして継続します。
7. 使用許諾契約書に合意するには対応するチェックボックスを選択します。
8. [完了] をクリックすることでインストールを完了します。

メモ (不足するアイコンなど) プラグインの使用に関する問題が発生した場合 `-clean` フラグを使用してコマンドラインから Eclipse を起動します。

26.2 Eclipse 内の XMLSpy エントリポイント

Eclipse 内にある以下のエントリポイントを使って XMLSpy の機能にアクセスすることができます:

- XMLSpy パースペクティブ[1013]から XMLSpy の GUI 機能に Eclipse GUI からアクセスすることができます。
- XMLSpy ツールバーボタン[1015]から XMLSpy のヘルプならびにドキュメントの新規作成機能にアクセスすることができます。

XMLSpy パースペクティブ

Eclipse における*パースペクティブ*とは、特定の機能に結び付けられた GUI ビューのことです。Eclipse 用の XMLSpy プラグインが Eclipse に統合されると、XMLSpy パースペクティブも自動的に作成されます。この*パースペクティブ*は、編集ビュー、メニュー、入力ヘルパー、そしてサイドバーなど、XMLSpy の GUI 要素を含む*パースペクティブ*となります。

XMLSpy と関連付けられている種類のファイル(例えば .xml) が開かれた時、このファイルを XMLSpy *パースペクティブ*で編集することができます。同様に、他の種類のファイルも Eclipse の別の*パースペクティブ*で開くことができます。更に、*アクティブ*なファイルに対しては、*パースペクティブ*を変更することができ、異なる環境で同じファイルの編集を行うことができます。従って、*パースペクティブ*には2つの利点があります:

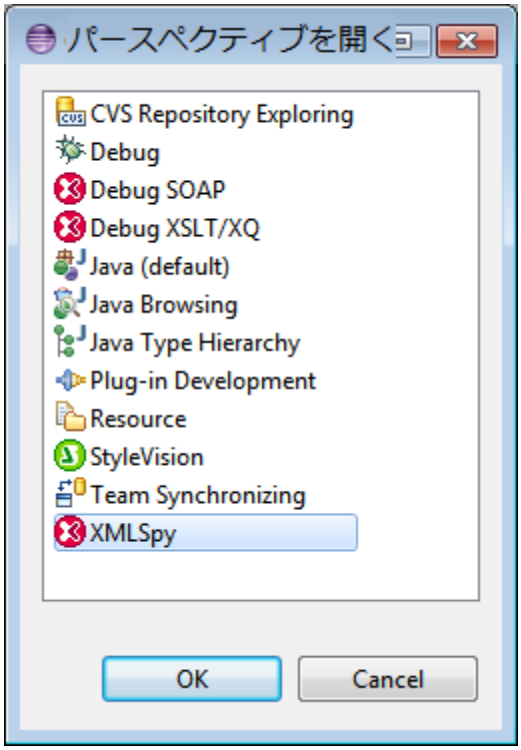
1. *アクティブ*なファイルに対して、作業環境を素早く変更することができる。
2. 新たな開発環境を開くこと無くファイルの切り替えを行うことができる(*パースペクティブ*に結び付けられた環境も有効になります)。

XMLSpy *パースペクティブ*を使った作業には以下のようなものがあります:

- XMLSpy *パースペクティブ*への切り替え
- XMLSpy *パースペクティブ*の環境設定
- XMLSpy *パースペクティブ*のカスタマイズ

XMLSpy パースペクティブへの切り替え

Eclipse にて、**メニューコマンド**から「**ウィンドウ | パースペクティブ | パースペクティブを開く | その他**」を選択します。表示されるダイアログボックス内で(*下のスクリーンショット*) select **XMLSpy**を選択して[開く]をクリックします。



空のドキュメント、またはアクティブなドキュメントに対して XMLSpy パースペクティブが開かれます。この方法でユーザーはパースペクティブの切り替えを行うことができます。目的のパースペクティブをパースペクティブを開くサブメニュー（のその他アイテムの上）に表示させることで、パースペクティブにより素早くアクセスすることができます。

この設定はカスタマイズダイアログにて行うことができます。ファイルが開かれた時、またはアクティブになったときにパースペクティブを自動的に切り替えることもできます。そのファイルが最初に開かれたときには、ファイルの種類に関連付けられているアプリケーションのパースペクティブが自動的に開かれ、このファイルの種類に対して、このパースペクティブをデフォルトのパースペクティブとして割り当てるか尋ねられます。この種類のファイルを開くたびに尋ねられたくない場合は、*今後このメッセージを表示しない* にチェックを入れ「OK」をクリックします。

XMLSpy パースペクティブの環境設定

パースペクティブの優先順位には以下が含まれています: XMLSpy パースペクティブの環境設定パースペクティブの設定には、(i) 関連付けられた種類のファイルが開かれたときに、自動的にパースペクティブを切り替える、（*上の参照*）(ii) 各 XMLSpy ツールバーを含めるオプションが含まれます。設定ダイアログにアクセスするには、[**ウィンドウ | 設定**]メニューコマンドを選択します。左のペインにあるパースペクティブのリストから XMLSpy を選択し、必要な設定を選択します。「**OK**」ボタンをクリックして完了します。

XMLSpy パースペクティブのカスタマイズ

カスタマイズオプションを使うことで、ショートカットやコマンドをパースペクティブに加えることができます。（*以下のスクリーンショットにある*）パースペクティブのカスタマイズダイアログにアクセスするには、（*XMLSpy パースペクティブ* の場合は）パースペクティブをアクティブにして、メニューコマンドから[**ウィンドウ | パースペクティブ | パースペクティブのカスタマイズ**]を選択します。
ツールバー可視性とメニュー可視性タブ内では、度のツールバーとメニューが表示されるかを指定することができます。アクションセットの使用可能タブでは、親メニューとツールバーにアクションセットを追加することができます。アクショングループ可用性タブでは、アクションセットを親メニューやツールバーへ追加することができます。アクショングループを有効にするには、対応するチェックボックスにチェックを入れてください。カスタマイズパースペクティブダイアログのショートカットタブでは、サブメニューのショートカットをセットすることができます。サブメニューコンボボックスから目的のサブメニューを選択し、ショートカットのカテゴリを選択し、パースペクティブに追加したいショートカットにチェックを入れます。[**適用 & 閉じる**]をクリックしてカスタマイズを完了すると変更が反映されます。

(C) 2015-2021 Altova GmbH

XMLSpy ツールバーボタン

The **XMLSpy** メニューにはXMLSpy により認識されるドキュメントの型が現在 Eclipse 内で開かれていない場合でも関連性のあるコマンドが含まれています。XMLSpy のスタンダードバージョンではコマンドの一部は「ファイル」メニュー内に存在しています。

XMLSpy ツールバーには次のボタンが含まれています（*下のスクリーンショット参照*）。

これらは左側から: (i) XMLSpy ヘルプを開く (ii) (**XMLSpy** メニューからアクセスすることの代替として) XMLSpy コマンドにアクセス

26.3 XMLSpy のデバッガーパースペクティブ

Enterprise Edition のXMLSpy には2つのデバッガー(XSLT/XQuery ならびにSOAP)が搭載されており、Professional Edition のXMLSpy には1つのデバッガー(XSLT/XQuery)が搭載されています。これらデバッガーのパースペクティブも、現在インストールされているXMLSpy のエディションに応じてEclipse で利用することができます。

デバッガーのパースペクティブに切り替えるには、メニューコマンドから「**ウィンドウ | パースペクティブを開く | その他**」を選択します。表示されるダイアログ(以下のスクリーンショット)にて、デバッガーを選択(例えば、XSLT/XQをデバッグ)し、「**OK**」をクリックします。

入力ヘルパーまたはアクティブなドキュメントに対して選択されたデバッガーのパースペクティブが適用されます。この方法でユーザーはパースペクティブの切り替えを行うことができます。目的のパースペクティブを「**パースペクティブを開く**」サブメニュー(の「**その他**」アイテムの上)に表示させることで、パースペクティブにより素早くアクセスすることができます。この設定はカスタマイズダイアログにて行うことができます。

デバッガーを使用する方法については、このドキュメンテーション内の対応するセクションを参照ください: XSLT とXQuery、W SDL とSOAP。

(C) 2015-2021 Altova GmbH

27 Code Generator

XMLSpy includes a built-in code generator which can generate Java, C++ or C# class files from XML schemas. The generated code consists of strongly-typed schema wrapper libraries that enable you to create software applications that process XML data. The schema wrapper libraries enable you to work with XML data programmatically, using types generated from the schema, without too much concern for the underlying XML API.

XMLSpy's schema editor is well suited as a software modeling and prototyping tool, allowing XML applications to be rapidly prototyped at a high level in XML Schema and then automatically generated. Changes to an application's XML Schema content model can be immediately reconciled with a software implementation simply by re-running the code generator.

The generated code supports the following operations:

- Read XML files into a Document Object Model (DOM) in-memory representation
- Write XML files from a DOM representation back to a system file
- Convert strings to XML DOM trees and vice versa.

The generated code is expressed in C++, Java or C# programming languages.

Target Language	C++	C#	Java
Development environments	Microsoft Visual Studio 2010, 2013, 2015, 2017, 2019	Microsoft Visual Studio 2010, 2013, 2015, 2017, 2019 Target frameworks: • .NET Framework • .NET Core 3.1 • .NET 5.0	(OpenJDK を含む) Java SE JDK 7、8、または11 Eclipse 4.4 or later Apache Ant (build.xml file)
XML DOM implementations	MSXML 6.0 Apache Xerces 3	System.Xml	JAXP

C++

You can configure whether the C++ generated output should use MSXML 6.0 or Apache Xerces 3. XMLSpy generates complete project (.vcproj) and solution (.sln) files for all supported versions of Visual Studio (see table above). The generated code optionally supports MFC.

Prerequisites:

1. To compile the generated C++ code, Windows SDK must be installed on your computer.
2. To use Xerces 3 for C++, you will need to install and build it by following instructions from http://xerces.apache.org/xerces-c/. Make sure to add the XERCES3 environment variable that points to the directory where Xerces was installed, for example: `C:\xerces-c-3.2.2`. Also, the PATH environment variable must include the path where the Xerces binaries are, for example: `%XERCES3%\bin`.
3. When building C++ code for Visual Studio and using a Xerces library precompiled for Visual C++, a compiler setting has to be changed in all projects of the solution:

a) Select all projects in the Solution Explorer.
b) On the **Project** menu, click **Properties**.
c) Click **Configuration Properties | C/C++ | Language**.
d) In the list of configurations, select *All Configurations*.
e) Change **Treat wchar_t as Built-in Type** to **No (/Zc:wchar_t-)**.

C#

The generated C# code can be used from any .NET capable programming language, such as VB.NET, Managed C++, or J#. Project files can be generated for all supported versions of Visual Studio (see table above).

Java

The generated Java output is written against the Java API for XML Processing (JAXP) and includes an Ant build file and project files for supported versions of Java and Eclipse (see table above).

Generated output

The designated destination folder for the generated code includes all the libraries and files required to manipulate XML files programmatically, namely:

- Standard Altova libraries
- Schema wrapper libraries
- An empty test application with sample source code. The test application skeleton is a compilable application that calls an empty Example() method. You can add your test code into this method for easy and quick testing of your new generated library.

Code generator templates

The generated code supports customization via a template language 1110 called SPL (Spy Programming Language). SPL gives full control in mapping XML Schema built-in data-types to the primitive datatypes of a particular programming language. This makes it possible to build your own templates to automate the generation of virtually any other format, for example: EJB's, WSDL files, SQL scripts, ASP or WML code.

(C) 2015-2021 Altova GmbH

27.1 What's new ...

Version 2021 R2

- When generating C# code, you can select .NET Core 3.1 and .NET 5.0 as target frameworks from code generation options[1108] (this adds to existing support for .NET Framework projects).

Version 2020 R2

- Code generated for XML schema wrapper libraries now provides more control over element namespaces and prefixes. New methods are available to declare or override namespaces for an element, or to append an element with a prefixed namespace. See Example: Purchase Order[1055].

Version 2020

- Added support for Visual Studio 2019
- End of support for Visual Studio 2008

Version 2018

- Added support for Microsoft Visual Studio 2013, 2015, 2017
- End of support for Visual Studio 2005 and Xerces 2.x

Version 2014

- Removal of compatibility mode option for code generation

Version 2011

- Contains bug fixes and enhancements

Version 2010 R3

- Support for Microsoft Visual Studio 2010
- Support for MSXML 6.0 in generated C++ code
- Support for 64-bit targets for C++ and C# projects

Version 2010 R2

- Code generation for C++ for the Linux platform

Version 2010

- Enumeration facets from XML schemas are now available as symbolic constants in the generated classes (using 2007r3 templates)

Version 2009 sp1

- Apache Xerces version 3.x support added (older versions starting from Xerces 2.6.x are still supported)

Version 2008 R2

- Support for generation of Visual Studio 2008 project files for C# and C++ has been added

Version 2008

- The new 2007 R3-style SPL templates have been further enhanced:
  - It is now possible to remove single elements
  - Access to schema metadata (e.g. element names, facets, enumerations, occurrence, etc.) is provided
  - Complex types derived by extension are now generated as derived classes

Version 2007 R3

Code Generator has been redesigned for version 2007 release 3 to simplify usage of the generated code, reduce code volume and increase performance.

- Handling of XML documents and nodes with explicit ownership, to avoid memory leaks and to enable multi-threading
- New syntax to avoid name collisions
- New data types for simpler usage and higher performance (native types where possible, new null handling, ...)
- Attributes are no longer generated as collections
- Simple element content is now also treated like a special attribute, for consistency
- New internal object model (important for customized SPL templates)
- Compatibility mode to generate code in the style of older releases

(C) 2015-2021 Altova GmbH

27.2 Generating Code from XML Schemas or DTDs

With XMLSpy code generator, you can generate C#, C++, or Java program code from XML schemas or DTDs. The generated schema wrapper libraries can then be integrated in your custom application in order to read, modify, or write XML documents programmatically.

Generating program code

1. Open the schema for which you want to generate source code.
2. Select the menu item **DTD/Schema | Generate Program Code**.
3. In the **Choose Template** pane of the dialog that pops up, set the code generator options 1108.
4. Click **OK**. The *Browse for Folder* dialog appears.
5. Select the target folder and click **OK**.
6. You are prompted to open the newly created project in Microsoft Visual Studio. Click **Yes**. If Java code is produced, you are prompted to open the corresponding output directory.

When XMLSpy generates code from an XML Schema or DTD, the following libraries are created:

C++ or C#	Java	Purpose
Altova	`com.altova`	Base library containing common runtime support, identical for every schema.
AltovaXML	`com.altova.xml`	Base library containing runtime support for XML, identical for every schema.
[YourSchema]	`com.YourSchema`	A library containing declarations generated from the input schema, named as the schema file or DTD. This library is a DOM (W3C Document Object Model) wrapper that allows you to read, modify and create XML documents easily and safely. All data is held inside the DOM, and there are methods for extracting data from the DOM, and to update and create data into the DOM. The generated C++ code supports either Microsoft MSXML or Apache Xerces 3. The syntax for using the generated code is generally similar for both DOM implementations, except for a few slight differences (for example, Xerces supports more overloaded functions). The generated C# code uses the .NET standard System.XML library as the underlying DOM implementation. The generated Java code uses JAXP (Java API for XML Processing) as the underlying DOM interface.
[YourSchemaTest]	`com.YourSchemaTest`	The generated code also includes a test application skeleton named after your schema (for example, *YourSchemaTest*). This is a compilable application that calls an empty Example() method. You can add your test code into this method for easy and quick testing of your new generated library.

While prototyping an application from a frequently changing XML schema, you may need to frequently generate code to the same directory, so that the schema changes are immediately reflected in the code. Note that the generated test application and the Altova libraries are overwritten every time when you generate code into the same target directory. Therefore, do not add code to the generated test application. Instead, integrate the Altova libraries into your project (see Integrating Schema Wrapper Libraries[1030]).

Name generation and namespaces

XMLSpy generates classes corresponding to all declared elements or complex types which redefine any complex type in your XML Schema, preserving the class derivation as defined by extensions of complex types in your XML Schema. In the case of complex schemas which import schema components from multiple namespaces, XMLSpy preserves this information by generating the appropriate C# or C++ namespaces or Java packages.

Generally, the code generator tries to preserve the names for generated namespaces, classes and members from the original XML Schema. Characters that are not valid in identifiers in the target language are replaced by a "_". Names that would collide with other names or reserved words are made unique by appending a number. Name generation can be influenced by changing the default settings in the SPL[1110] template.

The namespaces from the XML Schema are converted to packages in Java or namespaces in C# or C++ code, using the namespace prefix from the schema as code namespace. The complete library is enclosed in a package or namespace derived from the schema file name, so you can use multiple generated libraries in one program without name conflicts.

Data Types

XML Schema has a more elaborate data type model than Java, C# or C++. Code Generator converts the built-in XML Schema types to language-specific primitive types, or to classes delivered with the Altova library. Complex types and derived types defined in the schema are converted to classes in the generated library. Enumeration facets from simple types are converted to symbolic constants.

The mapping of simple types can be configured in the SPL template, see SPL Reference[1110].

If your XML instance files use schema types related to time and duration, these are converted to Altova native classes in the generated code. For information about the Altova library classes, see:

- Reference to Generated Classes (C++)[1062]
- Reference to Generated Classes (C#)[1077]
- Reference to Generated Classes (Java)[1092]

For information about type conversion and other details applicable to each language, see:

- About Schema Wrapper Libraries (C++)[1023]
- About Schema Wrapper Libraries (C#)[1026]
- About Schema Wrapper Libraries (Java)[1028]

Memory management

A DOM tree is comprised of nodes, which are always owned by a specific DOM document - even if the node is not currently part of the document's content. All generated classes are references to the DOM nodes they

(C) 2015-2021 Altova GmbH

represent, not values. This means that assigning an instance of a generated class does not copy the value, it only creates an additional reference to the same data.

XML Schema support

The following XML Schema constructs are translated into code:

a) XML namespaces

b) Simple types:

- Built-in XML schema types
- Simple types derived by extension
- Simple types derived by restriction
- Facets
- Enumerations
- Patterns

c) Complex types:

- Built-in anyType node
- User-defined complex types
- Derived by extension: Mapped to derived classes
- Derived by restriction
- Complex content
- Simple content
- Mixed content

The following advanced XML Schema features are not supported (or not fully supported) in generated wrapper classes:

- Wildcards: `xs:any` and `xs:anyAttribute`
- Content models (sequence, choice, all). Top-level compositor is available in SPL[1110], but is not enforced by generated classes.
- Default and fixed values for attributes. These are available in SPL[1110], but are not set or enforced by generated classes.
- The attributes `xsi:type`, abstract types. When you need to write the xsi:type attribute, use the `SetXsiType()` method of the generated classes.
- Union types: not all combinations are supported.
- Substitution groups are partially supported (resolved like "choice").
- Attribute `nillable="true"` and `xsi:nil`
- Uniqueness constraints
- Identity constraints (`key` and `keyref`)

27.2.1 About Schema Wrapper Libraries (C++)

Character Types

The generated C++ code can be compiled with or without Unicode support. Depending on this setting, the types `string_type` and `tstring` will both be defined as `std::string` or `std::wstring`, consisting of narrow or wide characters. To use Unicode characters in your XML file that are not representable with the current 8-bit character set, Unicode support must be enabled. Pay special attention to the `_T()` macros. This macro

ensures that string constants are stored correctly, whether you're compiling for Unicode or non-Unicode programs.

Data Types

The default mapping of XML Schema types to C++ data types is:

XML Schema	C++	Remarks
xs:string	string_type	string_type is defined as std::string or std:wstring
xs:boolean	bool	
xs:decimal	double	C++ does not have a decimal type, so double is used.
xs:float, xs:double	double	
xs:integer	__int64	xs:integer has unlimited range, mapped to __int64 for efficiency reasons.
xs:nonNegativeInteger	unsigned __int64	see above
xs:int	int	
xs:unsignedInt	unsigned int	
xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth	altova::DateTime 1062	
xs:duration	altova::Duration 1065	
xs:hexBinary and xs:base64Binary	std::vector<unsigned char>	Encoding and decoding of binary data is done automatically.
xs:anySimpleType	string_type	

All XML Schema types not contained in this list are derived types, and mapped to the same C++ type as their respective base type.

Generated Classes

For each type in the schema, a class is generated that contains a member for each attribute and element of the type. The members are named the same as the attributes or elements in the original schema (in case of possible collisions, a number is appended). For simple types, assignment and conversion operators are generated. For simple types with enumeration facets, the methods `GetEnumerationValue()` and `SetEnumerationValue(int)` can be used together with generated constants for each enumeration value. In addition, the method `StaticInfo()` allows accessing schema information as one of the following types:

altova::meta::SimpleType 1070
altova::meta::ComplexType 1069

(C) 2015-2021 Altova GmbH

Classes generated from complex types include the method `SetXsiType()`, which enables you to set the `xsi:type` attribute of the type. This method is useful when you want to create XML instance elements of a derived type.

In addition to the classes for the types declared in the XML Schema, a document class (identified with "CDoc" below) is generated. It contains all possible root elements as members, and various other methods. For more information about the class, see `[YourSchema]::[CDoc]` 1071.

Note: The actual class name depends on the name of the .xsd schema.

For each member attribute or element of a schema type, a new class is generated. For more information about such classes, see:

`[YourSchema]::MemberAttribute` 1075
`[YourSchema]::MemberElement` 1075

Note: The actual class names depend on the name of the schema attribute or element.

See also Example: Using the Schema Wrapper Libraries 1033.

Error Handling

Errors are reported by exceptions. The following exception classes are defined in the namespace `altova`:

Class	Base Class	Description
`Error`	`std::logic_error`	Internal program logic error (independent of input data).
`Exception`	`std::runtime_error`	Base class for runtime errors.
`InvalidArgumentsException`	`Exception`	A method was called with invalid argument values.
`ConversionException`	`Exception`	Exception thrown when a type conversion fails.
`StringParseException`	`ConversionException`	A value in the lexical space cannot be converted to value space.
`ValueNotRepresentableException`	`ConversionException`	A value in the value space cannot be converted to lexical space.
`OutOfRangeException`	`ConversionException`	A source value cannot be represented in target domain.
`InvalidOperationException`	`Exception`	An operation was attempted that is not valid in the given context.
`DataSourceUnavailableException`	`Exception`	A problem occurred while loading an XML instance.
`DataTargetUnavailableException`	`Exception`	A problem occurred while saving an XML instance.

All exception classes contain a message text and a pointer to a possible inner exception.

Method	Purpose
`string_type message()`	Returns a textual description of the exception.
`std::exception inner()`	Returns the exception that caused this exception, if available, or NULL.

Accessing schema information

The generated library allows accessing static schema information via the following classes. All methods are declared as `const`. The methods that return one of the metadata classes return a NULL object if the respective property does not exist.

altova::meta::Attribute 1068
altova::meta::ComplexType 1069
altova::meta::Element 1070
altova::meta::SimpleType 1070

27.2.2 About Schema Wrapper Libraries (C#)

The default mapping of XML Schema types to C# data types is as follows.

XML Schema	C#	Remarks
`xs:string`	`string`	
`xs:boolean`	`bool`	
`xs:decimal`	`decimal`	xs:decimal has unlimited range and precision, mapped to decimal for efficiency reasons.
`xs:float, xs:double`	`double`	
`xs:long`	`long`	
`xs:unsignedLong`	`ulong`	
`xs:int`	`int`	
`xs:unsignedInt`	`uint`	
`xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth`	Altova.Types.DateTime 1077	
`xs:duration`	Altova.Types.Duration 1082	
`xs:hexBinary and xs:base64Binary`	`byte[]`	Encoding and decoding of binary data is done automatically.

(C) 2015-2021 Altova GmbH

XML Schema	C#	Remarks
xs:anySimpleType	string	

All XML Schema types not contained in this list are derived types, and mapped to the same C# type as their respective base type.

Generated Classes

For each type in the schema, a class is generated that contains a member for each attribute and element of the type. The members are named the same as the attributes or elements in the original schema (in case of possible collisions, a number is appended). For simple types, assignment and conversion operators are generated. For simple types with enumeration facets, the methods `GetEnumerationValue()` and `SetEnumerationValue(int)` can be used together with generated constants for each enumeration value. In addition, the method `StaticInfo()` allows accessing schema information as one of the following types:

Altova.Xml.Meta.SimpleType 1086
Altova.Xml.Meta.ComplexType 1085

Classes generated from complex types include the method `SetXsiType()`, which enables you to set the `xsi:type` attribute of the type. This method is useful when you want to create XML instance elements of a derived type.

In addition to the classes for the types declared in the XML Schema, a document class (identified with "Doc" below) is generated. It contains all possible root elements as members, and various other methods. For more information about the class, see [YourSchema].[Doc] 1087.

Note: The actual class name depends on the name of the .xsd schema.

For each member attribute or element of a schema type, a new class is generated. For more information about such classes, see:

[YourSchemaType].MemberAttribute 1090
[YourSchemaType].MemberElement 1090

Note: The actual class names depend on the name of the schema attribute or element.

Error Handling

Errors are reported by exceptions. The following exception classes are defined in the namespace Altova:

Class	Base Class	Description
ConversionException	Exception	Exception thrown when a type conversion fails
StringParseException	ConversionException	A value in the lexical space cannot be converted to value space.
DataSourceUnavailableException	System.Exception	A problem occurred while loading an XML instance.
DataTargetUnavailableException	System.Exception	A problem occurred while saving an XML instance.

In addition, the following .NET exceptions are commonly used:

Class	Description
System.Exception	Base class for runtime errors
System.ArgumentException	A method was called with invalid argument values, or a type conversion failed.
System.FormatException	A value in the lexical space cannot be converted to value space.
System.InvalidCastException	A value cannot be converted to another type.
System.OverflowException	A source value cannot be represented in target domain.

Accessing schema information

The generated library allows accessing static schema information via the following classes:

Altova.Xml.Meta.Attribute 1084
Altova.Xml.Meta.ComplexType 1085
Altova.Xml.Meta.Element 1085
Altova.Xml.Meta.SimpleType 1086

The properties that return one of the metadata classes return null if the respective property does not exist.

27.2.3 About Schema Wrapper Libraries (Java)

The default mapping of XML Schema types to Java data types is as follows:

XML Schema	Java	Remarks
xs:string	String	
xs:boolean	boolean	
xs:decimal	java.math.BigDecimal	
xs:float, xs:double	double	
xs:integer	java.math.BigInteger	
xs:long	long	
xs:unsignedLong	java.math.BigInteger	Java does not have unsigned types.
xs:int	int	
xs:unsignedInt	long	Java does not have unsigned types.

(C) 2015-2021 Altova GmbH

XML Schema	Java	Remarks
xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth	com.altova.types.DateTime 1092	
xs:duration	com.altova.types.Duration 1096	
xs:hexBinary and xs:base64Binary	byte[]	Encoding and decoding of binary data is done automatically.
xs:anySimpleType	string	

All XML Schema types not contained in this list are derived types, and mapped to the same Java type as their respective base type.

Generated Classes

For each type in the schema, a class is generated that contains a member for each attribute and element of the type. The members are named the same as the attributes or elements in the original schema (in case of possible collisions, a number is appended). For simple types, assignment and conversion operators are generated. For simple types with enumeration facets, the methods `GetEnumerationValue()` and `SetEnumerationValue(int)` can be used together with generated constants for each enumeration value. In addition, the method `StaticInfo()` allows accessing schema information as one of the following types:

com.altova.xml.meta.SimpleType 1101
com.altova.xml.meta.ComplexType 1100

Classes generated from complex types include the method `SetXsiType()`, which enables you to set the `xsi:type` attribute of the type. This method is useful when you want to create XML instance elements of a derived type.

In addition to the classes for the types declared in the XML Schema, a document class (identified with "Doc" below) is generated. It contains all possible root elements as members, and various other methods. For more information about the class, see com.[YourSchema].[Doc] 1102.

Note: The actual class name depends on the name of the .xsd schema.

For each member attribute or element of a schema type, a new class is generated. For more information about such classes, see:

com.[YourSchema].[YourSchemaType].MemberAttribute 1105
com.[YourSchema].[YourSchemaType].MemberElement 1105

Note: The actual class names depend on the name of the schema attribute or element.

Error Handling

Errors are reported by exceptions. The following exception classes are defined in the namespace com.altova:

Class	Base Class	Description
SourceInstanceUnvailableException	Exception	A problem occurred while loading

Class	Base Class	Description
		an XML instance.
`TargetInstanceUnavailableException`	Exception	A problem occurred while saving an XML instance.

In addition, the following Java exceptions are commonly used:

Class	Description
`java.lang.Error`	Internal program logic error (independent of input data)
`java.lang.Exception`	Base class for runtime errors
`java.lang.IllegalArgumentsException`	A method was called with invalid argument values, or a type conversion failed.
`java.lang.ArithmeticException`	Exception thrown when a numeric type conversion fails.

Accessing schema information

The generated library allows accessing static schema information via the following classes:

`com.altova.xml.meta.Attribute` 1099
`com.altova.xml.meta.ComplexType` 1100
`com.altova.xml.meta.Element` 1100
`com.altova.xml.meta.SimpleType` 1101

The properties that return one of the metadata classes return null if the respective property does not exist.

27.2.4 Integrating Schema Wrapper Libraries

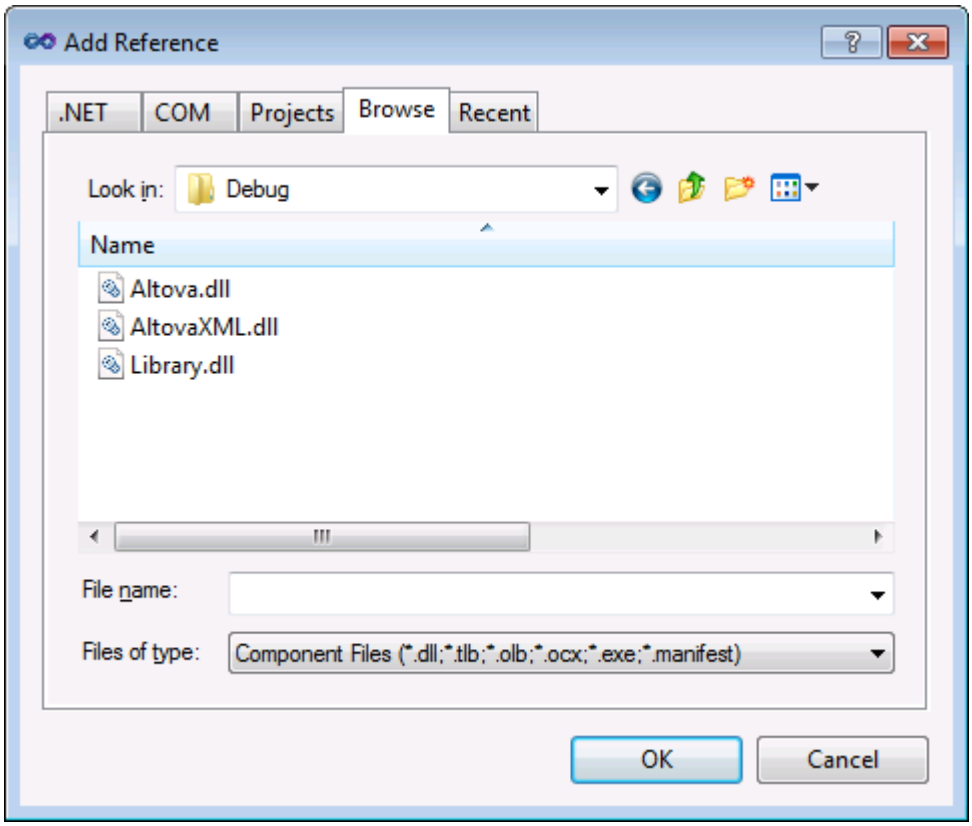
To use the Altova libraries in your custom project, refer to the libraries from your project (or include them into your project), as shown below for each language.

C#

To integrate the Altova libraries into an existing C# project:

1. After XMLSpy generates code from a schema (for example, **YourSchema.xsd**), build the generated **YourSchema.sln** solution in Visual Studio. This solution is in a project folder with the same name as the schema.
2. Right-click your existing project in Visual Studio, and select **Add Reference**.
3. On the Browse tab, browse for the following libraries: **Altova.dll**, **AltovaXML.dll**, and **YourSchema.dll** located in the output directory of the generated projects (for example, **bin\Debug**).

(C) 2015-2021 Altova GmbH



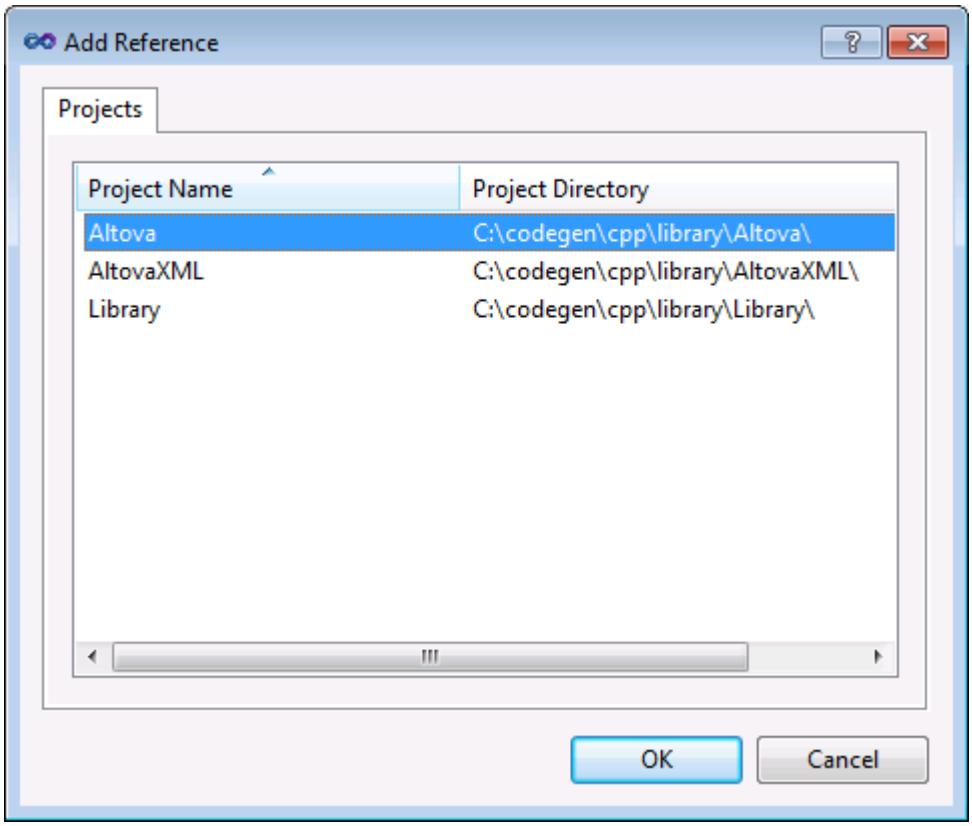
C++

The easiest way to integrate the libraries into an existing C++ project is to add the generated project files to your solution. For example, let's assume that you generated code from a schema called **Library.xsd** and selected **c:\codegen\cpp\library** as target directory. The generated libraries in this case are available at:

- c:\codegen\cpp\library\Altova.vcxproj
- c:\codegen\cpp\library\AltovaXML\AltovaXML.vcxproj
- c:\codegen\cpp\library\Library.vcxproj

First, open the generated **c:\codegen\cpp\library\Library.sln** solution and build it in Visual Studio.

Next, open your existing Visual Studio solution (in Visual Studio 2010, in this example), right-click it, select **Add | Existing Project**, and add the project files listed above, one by one. Be patient while Visual Studio parses the files. Next, right-click your project and select **Properties**. In the Property Pages dialog box, select **Common Properties | Framework and References**, and then click **Add New Reference**. Next, select and add each of the following projects: *Altova*, *AltovaXML*, and *Library*.



See also the MSDN documentation for using functionality from a custom library, as applicable to your version of Visual Studio, for example:

- If you chose to generate static libraries, see https://msdn.microsoft.com/en-us/library/ms235627(v=vs.100).aspx
- If you chose to generate dynamic libraries, see https://msdn.microsoft.com/en-us/library/ms235636(v=vs.100).aspx

The option to generate static or dynamic libraries is available in code generation options (see Code generator options 1108).

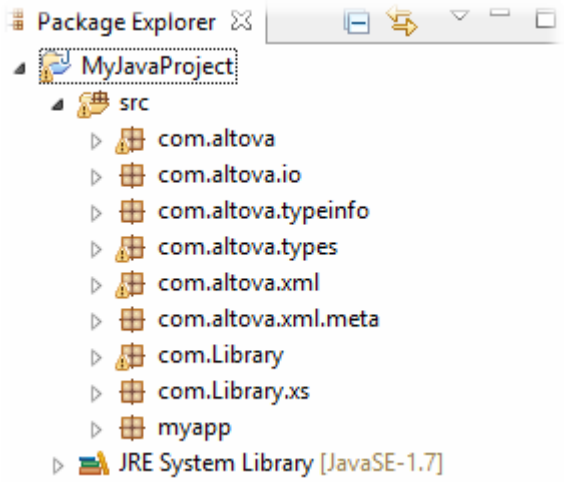
Java

One of the ways to integrate the Altova packages into your Java project is to copy the **com** directory of the generated code to the directory which stores the source packages of your Java project (for example, **C:\Workspace\MyJavaProject\src**). For example, let's assume that you generated code in **c:\codegen\java\library**. The generated Altova classes in this case are available at **c:\codegen\java\library\com**.

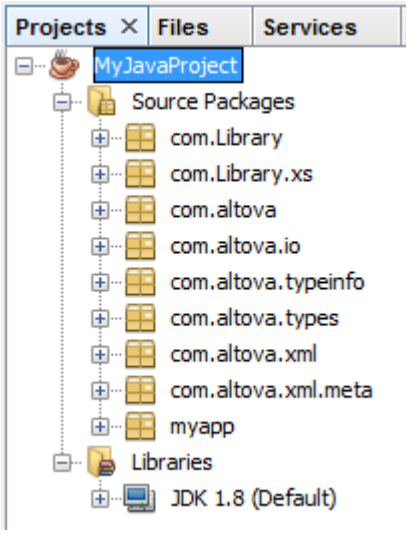
After copying the libraries, refresh the project. To refresh the project in Eclipse, select it in the Package Explorer, and press **F5**. To refresh the project in NetBeans IDE 8.0, select the menu command **Source | Scan for External Changes**.

Once you perform the copy operation, the Altova packages are available in the Package Explorer (in case of Eclipse), or under "Source Packages" in the Projects pane (in case of NetBeans IDE).

(C) 2015-2021 Altova GmbH



Altova packages in Eclipse 4.4



Altova packages in NetBeans IDE 8.0.2

27.2.5 Example: Book Library

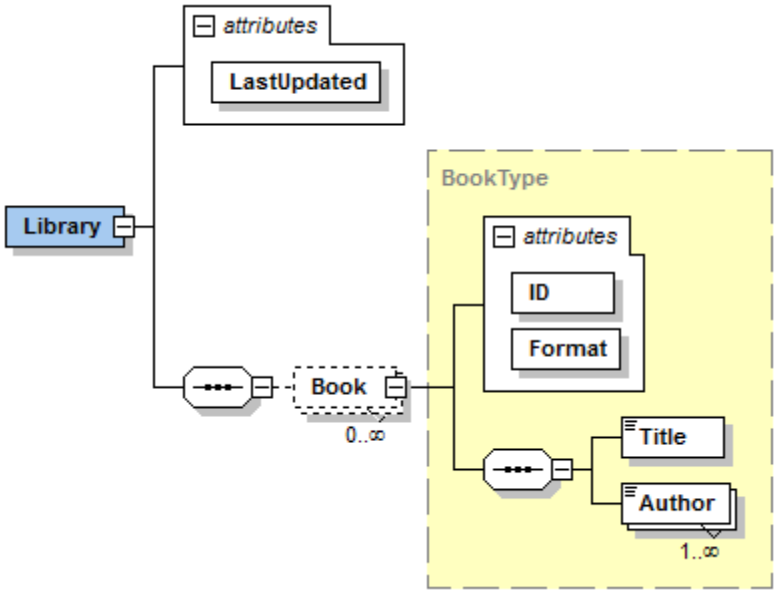
This example illustrates how to use the generated schema wrapper libraries in order to write or read programmatically XML documents conformant to the schema. Before using the sample code, take some time to understand the structure of the schema below.

The schema used in this example describes a library of books. The complete definition of the schema is shown below. Save this code listing as **Doc.xsd** if you want to get the same results as this example. You will need this schema to generate the code libraries used in this example.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.nanonull.com/LibrarySample"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.nanonull.com/LibrarySample" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xs:element name="Library">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Book" type="BookType" minOccurs="0"
maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="LastUpdated" type="xs:dateTime"/>
      </xs:complexType>
   </xs:element>
   <xs:complexType name="BookType">
      <xs:sequence>
         <xs:element name="Title" type="xs:string"/>
         <xs:element name="Author" type="xs:string" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="ID" type="xs:integer" use="required"/>
      <xs:attribute name="Format" type="BookFormatType" use="required"/>
   </xs:complexType>
   <xs:complexType name="DictionaryType">
      <xs:complexContent>
         <xs:extension base="BookType">
            <xs:sequence>
               <xs:element name="FromLang" type="xs:string"/>
               <xs:element name="ToLang" type="xs:string"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:simpleType name="BookFormatType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="Hardcover"/>
         <xs:enumeration value="Paperback"/>
         <xs:enumeration value="Audiobook"/>
         <xs:enumeration value="E-book"/>
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

Library is a root element of a `complexType` which can be graphically represented as follows in the schema view of XMLSpy:

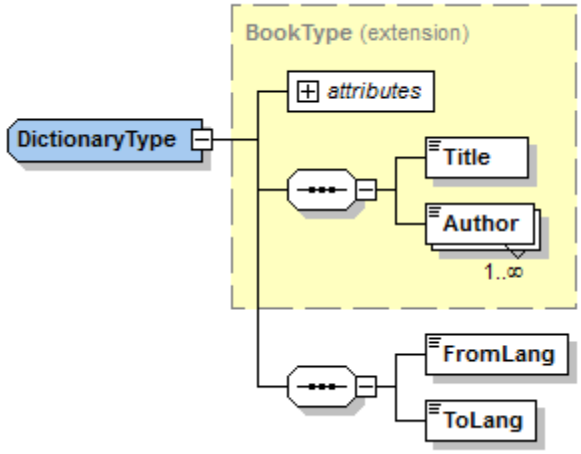
(C) 2015-2021 Altova GmbH



As shown above, the library has a **LastUpdated** attribute (defined as `xs:dateTime`), and stores a sequence of books. Each book is an `xs:complexType` and has two attributes: an **ID** (defined as `xs:integer`), and a **Format**. The format of any book can be hardcover, paperback, audiobook, or e-book. In the schema, **Format** is defined as `xs:simpleType` which uses an enumeration of the above-mentioned values.

Each book also has a **Title** element (defined as `xs:string`), as well as one or several **Author** elements (defined as `xs:string`).

The library may also contain books that are dictionaries. Dictionaries have the type `DictionaryType`, which is derived by extension from the `BookType`. In other words, a dictionary inherits all attributes and elements of a Book, plus two additional elements: **FromLang** and **ToLang**, as illustrated below.



The **FromLang** and **ToLang** elements store the source and destination language of the dictionary.

An XML instance file valid according to the schema above could therefore look as shown in the listing below (provided that it is in the same directory as the schema file):

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample"
xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd"
LastUpdated="2016-02-03T17:10:08.4977404">
   <Book ID="1" Format="E-book">
      <Title>The XMLSpy Handbook</Title>
      <Author>Altova</Author>
   </Book>
   <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/LibrarySample"
xsi:type="n1:DictionaryType">
      <Title>English-German Dictionary</Title>
      <Author>John Doe</Author>
      <FromLang>English</FromLang>
      <ToLang>German</ToLang>
   </Book>
</Library>
```

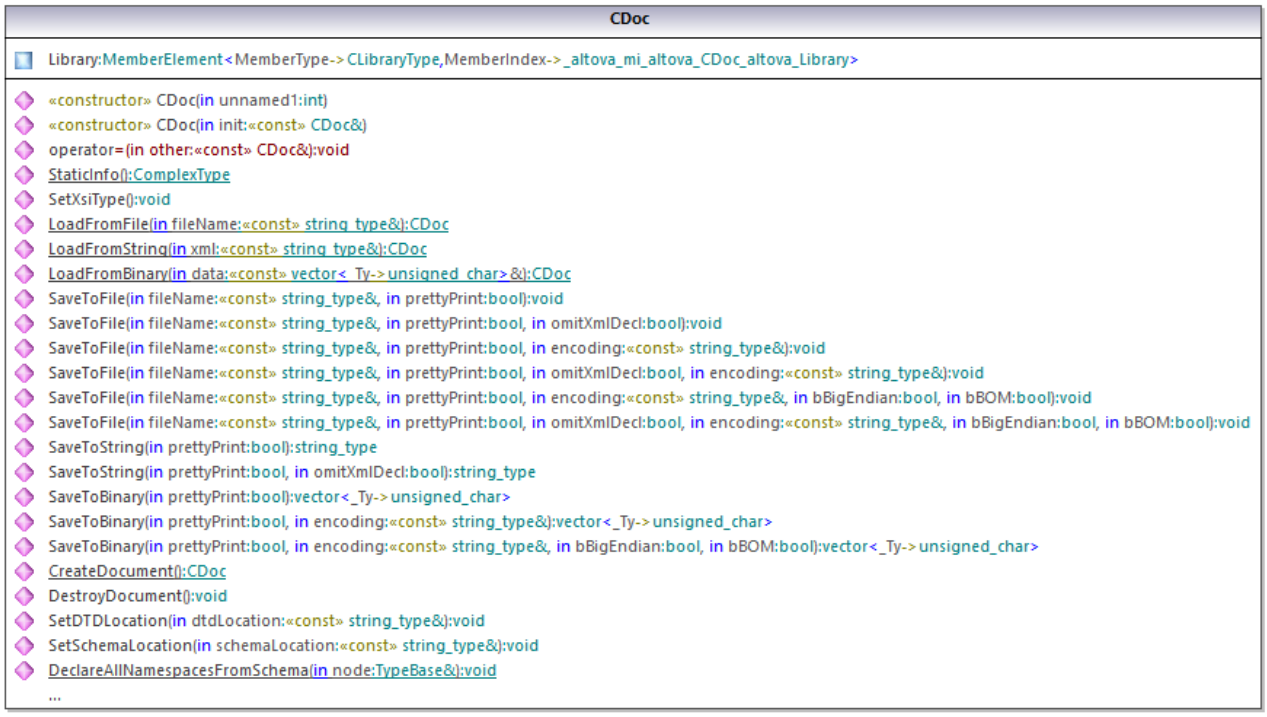
The next topics illustrate how to read from such a file programmatically, or write to such a file programmatically. To begin, generate the schema wrapper code from the schema above, using the steps described in Generating Code from XML Schemas or DTD 1021.

27.2.5.1 Reading and Writing XML Documents (C++)

After you generate code from the example schema 1033, a test C++ application is created, along with several supporting Altova libraries.

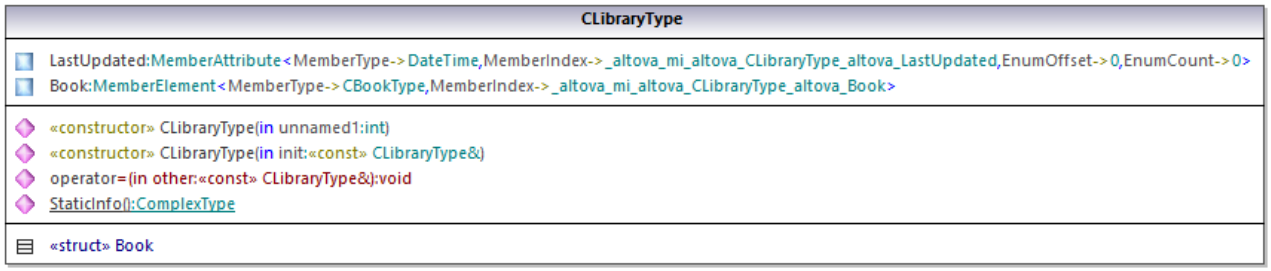
About the generated C++ libraries

The central class of the generated code is the `CDoc` class, which represents the XML document. Such a class is generated for every schema and its name depends on the schema file name (**Doc.xsd**, in this example). As shown in the diagram, this class provides methods for loading documents from files, binary streams, or strings (or saving documents to files, streams, strings). For a description of all members exposed by this class, see the class reference (`[YourSchema]::[CDoc]` 1071).

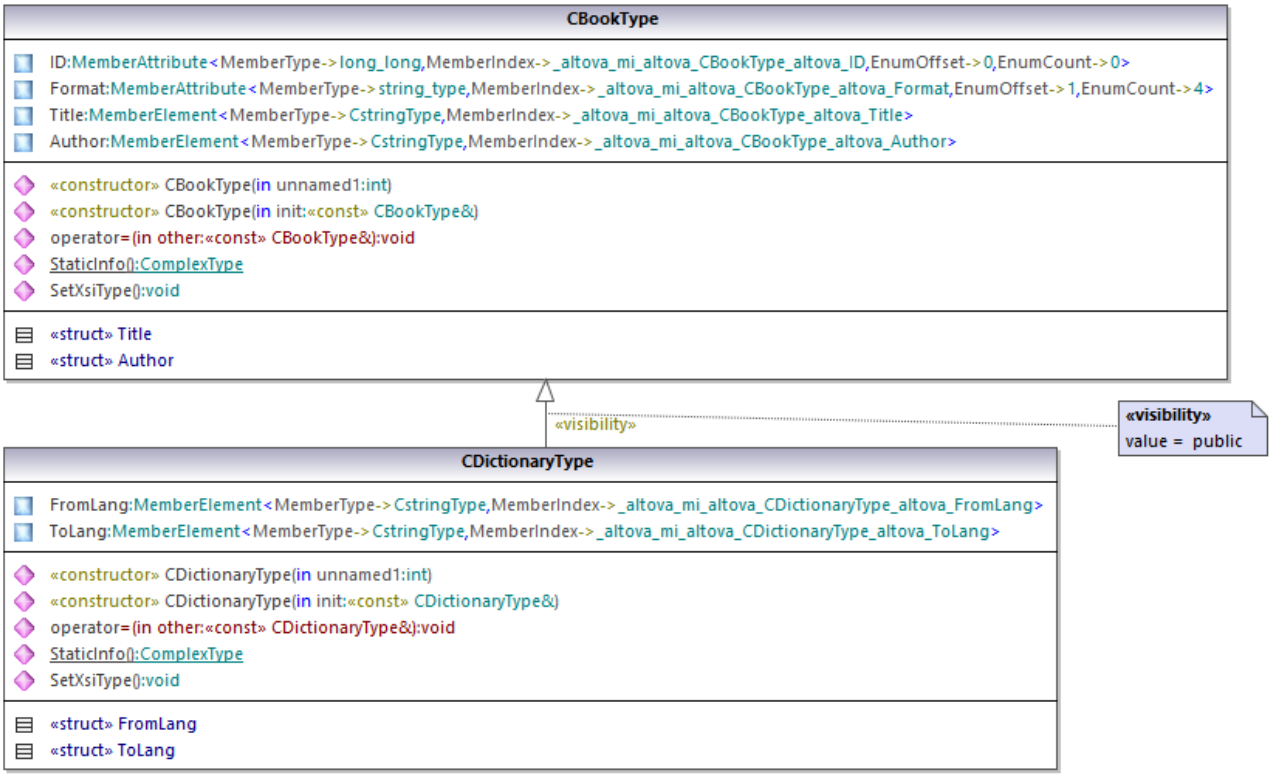


The `Library` field of the `CDoc` class represents the actual root of the document. **Library** is an element in the XML file, so in the C++ code it has a template class as type (`MemberElement`). The template class exposes methods and properties for interacting with the **Library** element. In general, each attribute and each element of a type in the schema is typed in the generated code with the `MemberAttribute` and `MemberElement` template classes, respectively. For more information, see `[YourSchema]::MemberAttribute`[1075] and `[YourSchema]::MemberElement`[1075] class reference.

The class `CLibraryType` is generated from the **LibraryType** complex type in the schema. Notice that the `CLibraryType` class contains two fields: `Book` and `LastUpdated`. According to the logic already mentioned above, these correspond to the **Book** element and **LastUpdated** attribute in the schema, and enable you to manipulate programmatically (append, remove, etc) elements and attributes in the instance XML document.



The `DictionaryType` is a complex type derived from **BookType** in the schema, so this relationship is also reflected in the generated classes. As illustrated in the diagram, the class `CDictionaryType` inherits the `CBookType` class.



If your XML schema defines simple types as enumerations, the enumerated values become available as `enum` values in the generated code. In the schema used in this example, a book format can be hardcover, paperback, e-book, and so on. Therefore, in the generated code, these values would be available through an `enum` that is a member of the `CBookFormatType` class.

Writing an XML document

1. Open the **LibraryTest.sln** solution in Visual Studio generated from the Library schema mentioned earlier in this example.

> While prototyping an application from a frequently changing XML schema, you may need to frequently generate code to the same directory, so that the schema changes are immediately reflected in the code. Note that the generated test application and the Altova libraries are overwritten every time when you generate code into the same target directory. Therefore, do not add code to the generated test application. Instead, integrate the Altova libraries into your project (see Integrating Schema Wrapper Libraries 1030).

2. In Solution Explorer, open the **LibraryTest.cpp** file, and edit the `Example()` method as shown below.

```
#include <ctime> // required to get current time
using namespace Doc; // required to work with Altova libraries

void Example()
{
```

 (C) 2015-2021 Altova GmbH

```
   // Create a new, empty XML document
   CDoc libDoc = CDoc::CreateDocument();

   // Create the root element <Library> and add it to the document
   CLibraryType lib = libDoc.Library.append();

   // Get current time and set the "LastUpdated" attribute using Altova classes
   time_t t = time(NULL);
   struct tm * now = localtime( & t );
   altova::DateTime dt = altova::DateTime(now->tm_year + 1900, now->tm_mon + 1, now-
>tm_mday, now->tm_hour, now->tm_min, now->tm_sec);
   lib.LastUpdated = dt;

   // Create a new <Book> and add it to the library
   CBookType book = lib.Book.append();

   // Set the "ID" attribute of the book
   book.ID = 1;

   // Set the "Format" attribute of the <Book> using an enumeration constant
   book.Format.SetEnumerationValue( CBookFormatType::k_Paperback );

   // Add the <Title> and <Author> elements, and set values
   book.Title.append() = _T("The XML Spy Handbook");
   book.Author.append() = _T("Altova");

   // Append a dictionary (book of derived type) and populate its attributes and
elements
   CDictionaryType dictionary = CDictionaryType(lib.Book.append().GetNode());
   dictionary.ID = 2;
   dictionary.Format.SetEnumerationValue( CBookFormatType::k_E_book);
   dictionary.Title.append() = _T("English-German Dictionary");
   dictionary.Author.append() = _T("John Doe");
   dictionary.FromLang.append() = _T("English");
   dictionary.ToLang.append() = _T("German");

   // Since dictionary a derived type, set the xsi:type attribute of the book element
   dictionary.SetXsiType();

   // Optionally, set the schema location
   libDoc.SetSchemaLocation(_T("Library.xsd"));

   // Save the XML document to a file with default encoding (UTF-8),
   // "true" causes the file to be pretty-printed.
   libDoc.SaveToFile(_T("GeneratedLibrary.xml"), true);

   // Destroy the document
   libDoc.DestroyDocument();
}
```

3. Press **F5** to start debugging. If the code was executed successfully, a **GeneratedLibrary.xml** file is created in the solution output directory.

Reading an XML document

1. Open the **LibraryTest.sln** solution in Visual Studio.
2. Save the code below as **Library1.xml** to a directory that can be read by the program code (for example, the same directory as **LibraryTest.sln**).

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample"
xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd"
LastUpdated="2016-02-03T17:10:08.4977404">
   <Book ID="1" Format="E-book">
      <Title>The XMLSpy Handbook</Title>
      <Author>Altova</Author>
   </Book>
   <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/LibrarySample"
xsi:type="n1:DictionaryType">
      <Title>English-German Dictionary</Title>
      <Author>John Doe</Author>
      <FromLang>English</FromLang>
      <ToLang>German</ToLang>
   </Book>
</Library>
```

3. In Solution Explorer, open the **LibraryTest.cpp** file, and edit the `Example()` method as shown below.

```
using namespace Doc;
void Example()
{
   // Load XML document
   CDoc libDoc = CDoc::LoadFromFile(_T("Library1.xml"));

   // Get the first (and only) root element <Library>
   CLibraryType lib = libDoc.Library.first();

   // Check whether an element exists:
   if (!lib.Book.exists())
   {
      tcout << "This library is empty." << std::endl;
      return;
   }

   // iteration: for each <Book>...
   for (Iterator<CBookType> itBook = lib.Book.all(); itBook; ++itBook)
   {
      // output values of ISBN attribute and (first and only) title element
      tcout << "ID: " << itBook->ID << std::endl;
      tcout << "Title: " << tstring(itBook->Title.first()) << std::endl;

      // read and compare an enumeration value
      if (itBook->Format.GetEnumerationValue() == CBookFormatType::k_Paperback)
         tcout << "This is a paperback book." << std::endl;

      // for each <Author>...
      for (CBookType::Author::iterator itAuthor = itBook->Author.all(); itAuthor; +
+itAuthor)
         tcout << "Author: " << tstring(itAuthor) << std::endl;

      // alternative: use count and index
      for (unsigned int j = 0; j < itBook->Author.count(); ++j)
         tcout << "Author: " << tstring(itBook->Author[j]) << std::endl;
   }

   // Destroy the document
```

```
    libDoc.DestroyDocument();
}
```

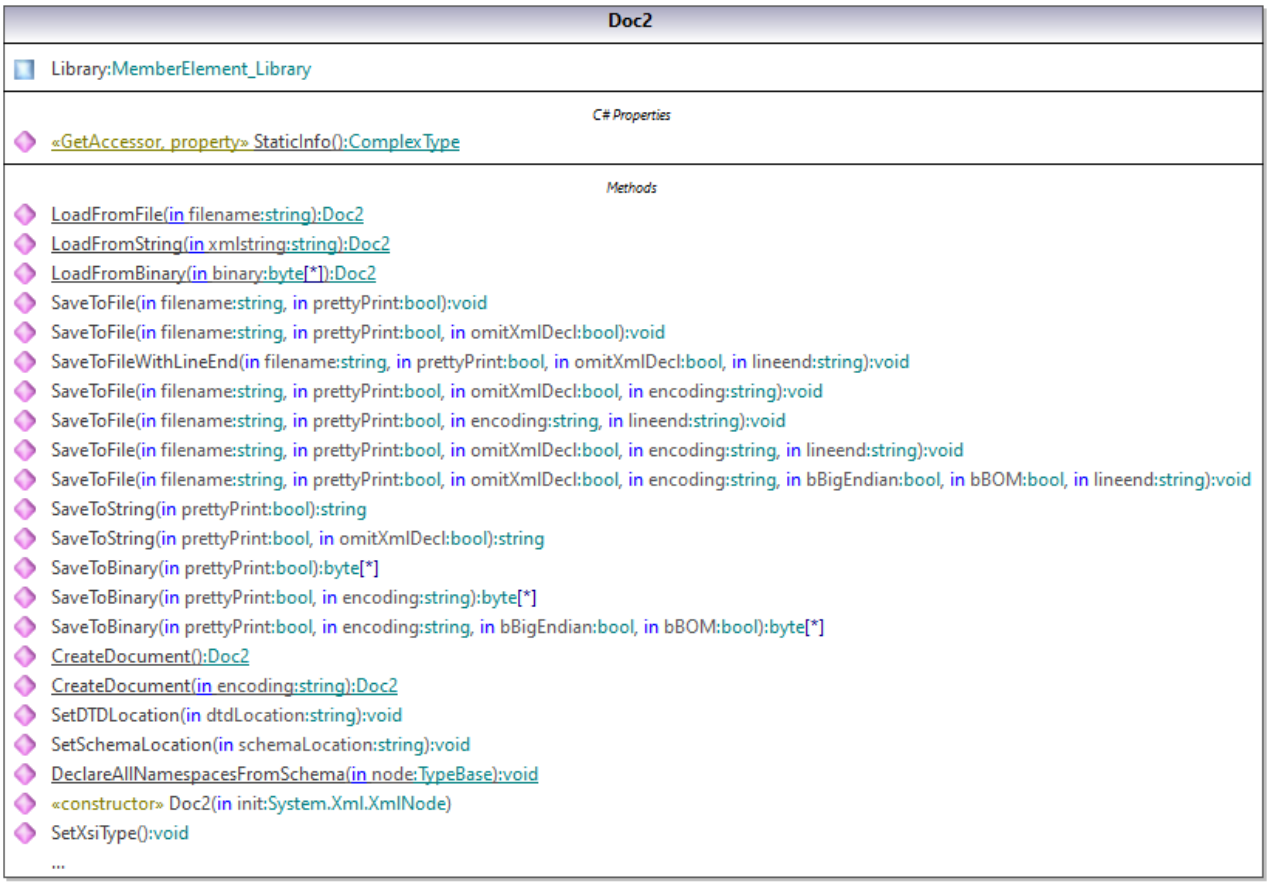
4. Press **F5** to start debugging.

27.2.5.2 Reading and Writing XML Documents (C#)

After you generate code from the example schema[1033], a test C# application is created, along with several supporting Altova libraries.

About the generated C# libraries

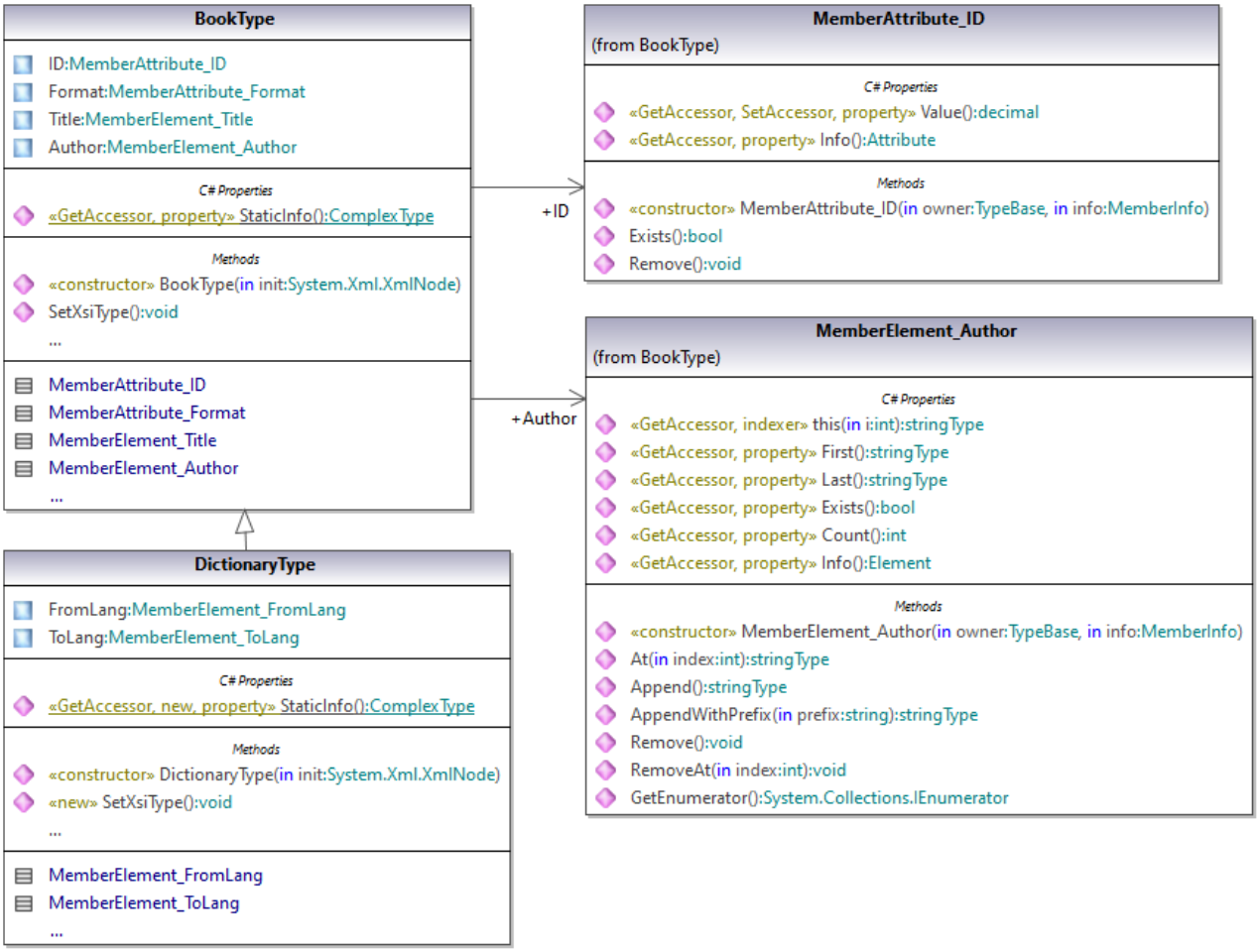
The central class of the generated code is the `Doc2` class, which represents the XML document. Such a class is generated for every schema and its name depends on the schema file name (**Doc.xsd**, in this example). Note that this class is called `Doc2` to avoid a possible conflict with the namespace name. As shown in the diagram, this class provides methods for loading documents from files, binary streams, or strings (or saving documents to files, streams, strings). For a description of this class, see the class reference ([YourSchema].[Doc][1087]).



The `Library` member of the `Doc2` class represents the actual root of the document.

According to the code generation rules mentioned in About Schema Wrapper Libraries (C#) 1026, member classes are generated for each attribute and for each element of a type. In the generated code, the name of such member classes is prefixed with `MemberAttribute_` and `MemberElement_`, respectively. Examples of such classes are `MemberAttribute_ID` and `MemberElement_Author`, generated from the **Author** element and **ID** attribute of a book, respectively (in the diagram below, they are classes nested under `BookType`). Such classes enable you to manipulate programmatically the corresponding elements and attributes in the instance XML document (for example, append, remove, set value, etc). For more information, see the `[YourSchemaType].MemberAttribute` 1090 and `[YourSchemaType].MemberElement` 1090 class reference.

Since the **DictionaryType** is a complex type derived from **BookType** in the schema, this relationship is also reflected in the generated classes. As illustrated in the diagram below, the class `DictionaryType` inherits the `BookType` class.



If your XML schema defines simple types as enumerations, the enumerated values become available as `Enum` values in the generated code. In the schema used in this example, a book format can be hardcover, paperback, e-book, and so on. Therefore, in the generated code, these values would be available through an `Enum` that is a member of the `BookFormatType` class.

Writing an XML document

1. Open the **LibraryTest.sln** solution in Visual Studio generated from the Library schema mentioned earlier in this example.

While prototyping an application from a frequently changing XML schema, you may need to frequently generate code to the same directory, so that the schema changes are immediately reflected in the code. Note that the generated test application and the Altova libraries are overwritten every time when you generate code into the same target directory. Therefore, do not add code to the generated test application. Instead, integrate the Altova libraries into your project (see Integrating Schema Wrapper Libraries 1030).

2. In Solution Explorer, open the **LibraryTest.cs** file, and edit the `Example()` method as shown below.

```
protected static void Example()
{
    // Create a new XML document
    Doc2 doc = Doc2.CreateDocument();
    // Append the root element
    LibraryType root = doc.Library.Append();

    // Create the generation date using Altova DateTime class
    Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);
    // Append the date to the root
    root.LastUpdated.Value = dt;

    // Add a new book
    BookType book = root.Book.Append();
    // Set the value of the ID attribute
    book.ID.Value = 1;
    // Set the format of the book (enumeration)
    book.Format.EnumerationValue = BookFormatType.EnumValues.eHardcover;
    // Set the Title and Author elements
    book.Title.Append().Value = "The XMLSpy Handbook";
    book.Author.Append().Value = "Altova";

    // Append a dictionary (book of derived type) and populate its attributes
and elements
    DictionaryType dictionary = new DictionaryType(root.Book.Append().Node);
    dictionary.ID.Value = 2;
    dictionary.Title.Append().Value = "English-German Dictionary";
    dictionary.Format.EnumerationValue = BookFormatType.EnumValues.eE_book;
    dictionary.Author.Append().Value = "John Doe";
    dictionary.FromLang.Append().Value = "English";
    dictionary.ToLang.Append().Value = "German";
    // Since it's a derived type, make sure to set the xsi:type attribute of the
book element
    dictionary.SetXsiType();

    // Optionally, set the schema location (adjust the path if
    // your schema is not in the same folder as the generated instance file)
    doc.SetSchemaLocation("Library.xsd");

    // Save the XML document with the "pretty print" option enabled
    doc.SaveToFile("GeneratedLibrary.xml", true);
}
```

3. Press **F5** to start debugging. If the code was executed successfully, a **GeneratedLibrary.xml** file is created in the solution output directory (typically, **bin/Debug**).

Reading an XML document

1. Open the **LibraryTest.sln** solution in Visual Studio.
2. Save the code below as **Library.xml** to the output directory of the project (by default, **bin/Debug**). This is the file that will be read by the program code.

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample"
xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd"
LastUpdated="2016-02-03T17:10:08.4977404">
   <Book ID="1" Format="E-book">
      <Title>The XMLSpy Handbook</Title>
      <Author>Altova</Author>
   </Book>
   <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/LibrarySample"
xsi:type="n1:DictionaryType">
      <Title>English-German Dictionary</Title>
      <Author>John Doe</Author>
      <FromLang>English</FromLang>
      <ToLang>German</ToLang>
   </Book>
</Library>
```

3. In Solution Explorer, open the **LibraryTest.cs** file, and edit the `Example()` method as shown below.

```
protected static void Example()
{
    // Load the XML file
    Doc2 doc = Doc2.LoadFromFile("Library.xml");
    // Get the root element
    LibraryType root = doc.Library.First;

    // Read the library generation date
    Altova.Types.DateTime dt = root.LastUpdated.Value;
    string dt_as_string = dt.ToString(DateTimeFormat.W3_dateTime);
    Console.WriteLine("The library generation date is: " + dt_as_string);

    // Iteration: for each <Book>...
    foreach (BookType book in root.Book)
    {
      // Output values of ID attribute and (first and only) title element
      Console.WriteLine("ID:    " + book.ID.Value);
      Console.WriteLine("Title: " + book.Title.First.Value);

      // Read and compare an enumeration value
      if (book.Format.EnumerationValue == BookFormatType.EnumValues.ePaperback)
         Console.WriteLine("This is a paperback book.");

      // Iteration: for each <Author>
      foreach (xs.stringType author in book.Author)
         Console.WriteLine("Author: " + author.Value);

      // Determine if this book is of derived type
      if (book.Node.Attributes.GetNamedItem("xsi:type") != null)
      {
```

```
         // Find the value of the xsi:type attribute
         string xsiTypeValue =
book.Node.Attributes.GetNamedItem("xsi:type").Value;
         // Get the namespace URI and the lookup prefix of this namespace
         string namespaceUri = book.Node.NamespaceURI;
         string prefix = book.Node.GetPrefixOfNamespace(namespaceUri);

         // if this book has DictionaryType
         if (namespaceUri == "http://www.nanonull.com/LibrarySample" &&
xsiTypeValue.Equals(prefix + ":DictionaryType"))
         {
            // output additional fields
            DictionaryType dictionary = new DictionaryType(book.Node);
            Console.WriteLine("Language from: " +
dictionary.FromLang.First.Value);
            Console.WriteLine("Language to: " + dictionary.ToLang.First.Value);
         }
         else
         {
            throw new Exception("Unexpected book type");
         }
      }
   }

   Console.ReadLine();
}
```

4. Press **F5** to start debugging. If the code was executed successfully, **Library.xml** will be read by the program code, and its contents displayed as console output.

Reading and writing elements and attributes

Values of attributes and elements can be accessed using the `Value` property of the generated member element or attribute class, for example:

```
// Output values of ID attribute and (first and only) title element
Console.WriteLine("ID:    " + book.ID.Value);
Console.WriteLine("Title: " + book.Title.First.Value);
```

To get the value of the **Title** element in this particular example, we also used the `First()` method, since this is the first (and only) **Title** element of a book. For cases when you need to pick a specific element from a list by index, use the `At()` method.

The class generated for each member element of a type implements the standard `System.Collections.IEnumerable` interface. This makes it possible to loop through multiple elements of the same type. In this particular example, you can loop through all books of a Library object as follows:

```
// Iteration: for each <Book>...
foreach (BookType book in root.Book)
{
    // your code here...
}
```

To add a new element, use the `Append()` method. For example, the following code appends the root element to the document:

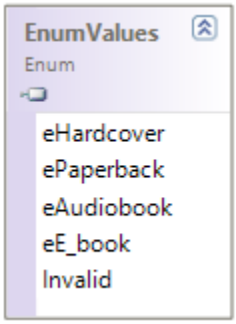
```
// Append the root element to the library
LibraryType root = doc.Library.Append();
```

You can set the value of an attribute (like ID in this example) as follows:

```
// Set the value of the ID attribute
book.ID.Value = 1;
```

Reading and writing enumeration values

If your XML schema defines simple types as enumerations, the enumerated values become available as `Enum` values in the generated code. In the schema used in this example, a book format can be hardcover, paperback, e-book, and so on. Therefore, in the generated code, these values would be available through an `Enum`:



To assign enumeration values to an object, use code such as the one below:

```
// Set the format of the book (enumeration)
book.Format.EnumerationValue = BookFormatType.EnumValues.eHardcover;
```

You can read such enumeration values from XML instance documents as follows:

```
// Read and compare an enumeration value
if (book.Format.EnumerationValue == BookFormatType.EnumValues.ePaperback)
Console.WriteLine("This is a paperback book.");
```

When an "if" condition is not enough, create a switch to determine each enumeration value and process it as required.

Working with xs:dateTime and xs:duration types

If the schema from which you generated code uses time and duration types such as `xs:dateTime`, or `xs:duration`, these are converted to Altova native classes in generated code. Therefore, to write a date or duration value to the XML document, do the following:

1. Construct an `Altova.Types.DateTime` 1077 or `Altova.Types.Duration` 1082 object (either from `System.DateTime`, or by using parts such as hours and minutes, see `Altova.Types.DateTime` 1077 and `Altova.Types.Duration` 1082 for more information).
2. Set the object as value of the required element or attribute, for example:

```
// Create the library generation date using Altova DateTime class
Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);
// Append the date to the root
root.LastUpdated.Value = dt;
```

To read a date or duration from an XML document, do the following:

1. Declare the element value (or attribute) as Altova.Types.DateTime[1077] or Altova.Types.Duration[1082] object.
2. Format the required element or attribute, for example:

```
// Read the library generation date
Altova.Types.DateTime dt = root.LastUpdated.Value;
string dt_as_string = dt.ToString(DateTimeFormat.W3_dateTime);
Console.WriteLine("The library generation date is: " + dt_as_string);
```

For more information, see Altova.Types.DateTime[1077] and Altova.Types.Duration[1082] class reference.

Working with derived types

If your XML schema defines derived types, you can preserve type derivation in XML documents that you create or load programmatically. Taking the schema used in this example, the following code listing illustrates how to create a new book of derived type `DictionaryType`:

```
// Append a dictionary (book of derived type) and populate its attributes and elements
DictionaryType dictionary = new DictionaryType(root.Book.Append().Node);
dictionary.ID.Value = 2;
dictionary.Title.Append().Value = "English-German Dictionary";
dictionary.Author.Append().Value = "John Doe";
dictionary.FromLanguage.Append().Value = "English";
dictionary.ToLanguage.Append().Value = "German";

// Since it's a derived type, make sure to set the xsi:type attribute of the book
element
dictionary.SetXsiType();
```

Note that it is important to set the `xsi:type` attribute of the newly created book. This ensures that the book type will be interpreted correctly by the schema when the XML document is validated.

When you load data from an XML document, the following code listing shows how to identify a book of derived type `DictionaryType` in the loaded XML instance. First, the code finds the value of the `xsi:type` attribute of the book node. If the namespace URI of this node is `http://www.nanonull.com/LibrarySample`, and if the URI lookup prefix and type matches the value of the `xsi:type` attribute, then this is a dictionary:

```
    // Determine if this book is of derived type
    if (book.Node.Attributes.GetNamedItem("xsi:type") != null)
    {
        // Find the value of the xsi:type attribute
        string xsiTypeValue = book.Node.Attributes.GetNamedItem("xsi:type").Value;
        // Get the namespace URI and the lookup prefix of this namespace
        string namespaceUri = book.Node.NamespaceURI;
        string prefix = book.Node.GetPrefixOfNamespace(namespaceUri);

        // if this book has DictionaryType
```

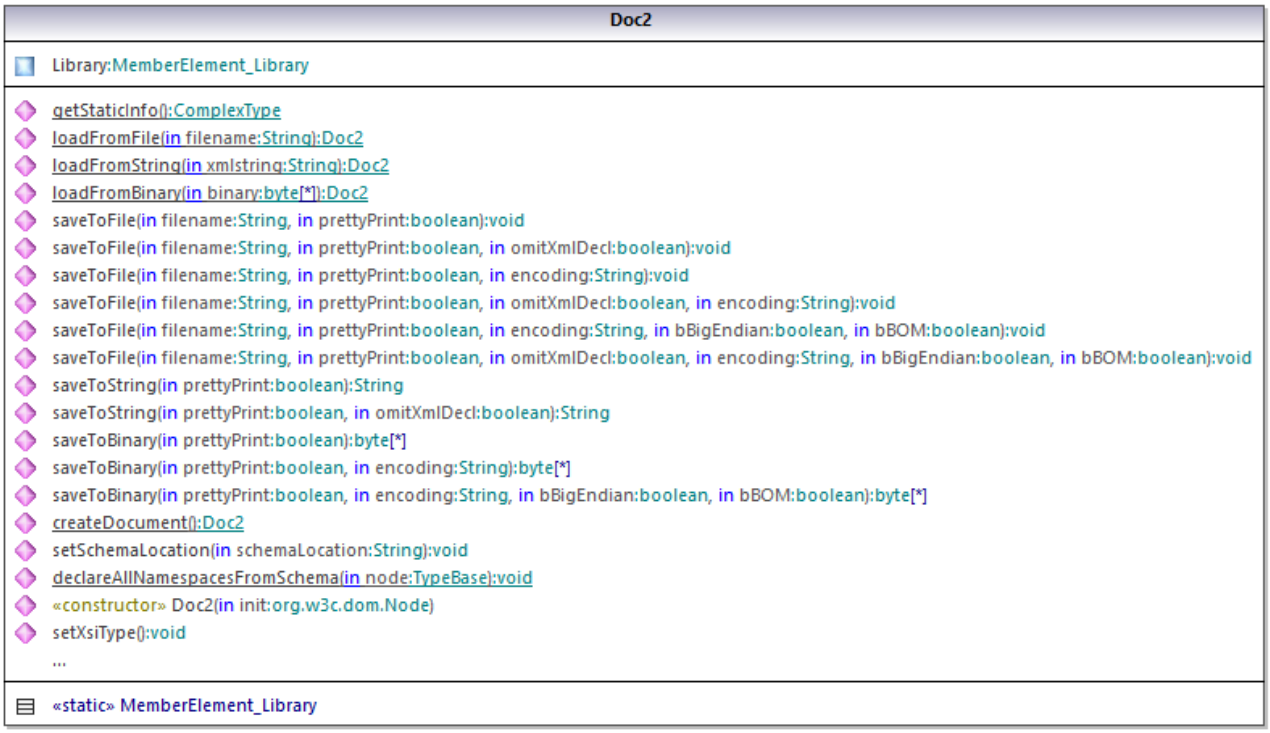
```
          if (namespaceUri == "http://www.nanonull.com/LibrarySample" &&
xsiTypeValue.Equals(prefix + ":DictionaryType"))
          {
             // output additional fields
             DictionaryType dictionary = new DictionaryType(book.Node);
             Console.WriteLine("Language from: " + dictionary.FromLang.First.Value);
             Console.WriteLine("Language to: " + dictionary.ToLang.First.Value);
          }
          else
          {
              throw new Exception("Unexpected book type");
          }
      }
```

27.2.5.3 Reading and Writing XML Documents (Java)

After you generate code from the example schema[1033], a test Java project is created, along with several supporting Altova libraries.

About the generated Java libraries

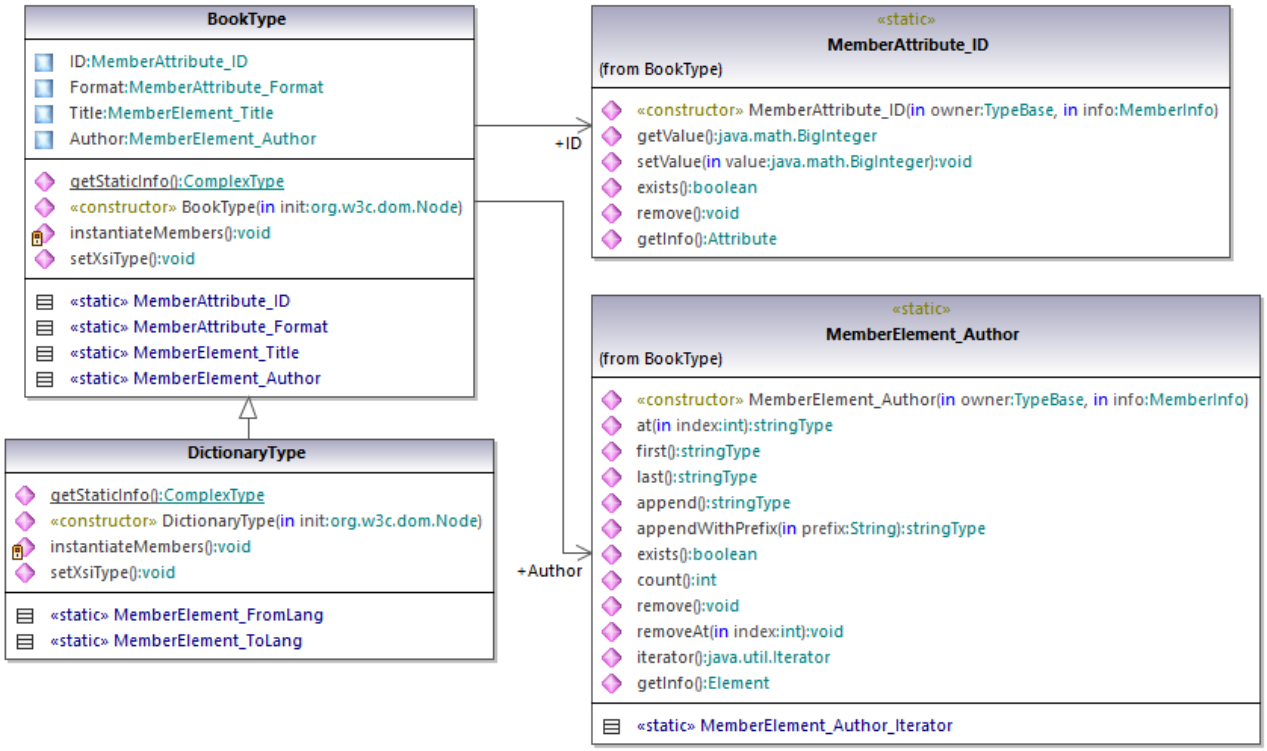
The central class of the generated code is the `Doc2` class, which represents the XML document. Such a class is generated for every schema and its name depends on the schema file name (**Doc.xsd**, in this example). Note that this class is called `Doc2` to avoid a possible conflict with the namespace name. As shown in the diagram, this class provides methods for loading documents from files, binary streams, or strings (or saving documents to files, streams, strings). For a description of this class, see the `com.[YourSchema].[Doc]`[1102] class reference.



The `Library` member of the `Doc2` class represents the actual root of the document.

According to the code generation rules mentioned in About Generated Java Code 1028, member classes are generated for each attribute and for each element of a type. In the generated code, the name of such member classes is prefixed with `MemberAttribute_` and `MemberElement_`, respectively. In the diagram below, examples of such classes are `MemberAttribute_ID` and `MemberElement_Author`, generated from the **Author** element and **ID** attribute of a book, respectively. Such classes enable you to manipulate programmatically the corresponding elements and attributes in the instance XML document (for example, append, remove, set value, etc). For more information, see the `com.[YourSchema].[YourSchemaType].MemberAttribute` 1105 and `com.[YourSchema].[YourSchemaType].MemberElement` 1105 class reference.

Since the **DictionaryType** is a complex type derived from **BookType** in the schema, this relationship is also reflected in the generated classes. As illustrated in the diagram below, the class `DictionaryType` inherits the `BookType` class.



If your XML schema defines simple types as enumerations, the enumerated values become available as `Enum` values in the generated code. In the schema used in this example, a book format can be hardcover, paperback, e-book, and so on. Therefore, in the generated code, these values would be available through an `Enum` that is a member of the `BookFormatType` class.

Writing an XML document

1. On the **File** menu of Eclipse, click **Import**, select **Existing Projects into Workspace**, and click **Next**.
2. Next to **Select root directory**, click **Browse**, select the directory to which you generated the Java code, and then click **Finish**.
3. In the Eclipse Package Explorer, expand the **com.LibraryTest** package and open the **LibraryTest.java** file.

While prototyping an application from a frequently changing XML schema, you may need to frequently generate code to the same directory, so that the schema changes are immediately reflected in the code. Note that the generated test application and the Altova libraries are overwritten every time when you generate code into the same target directory. Therefore, do not add code to the generated test application. Instead, integrate the Altova libraries into your project (see Integrating Schema Wrapper Libraries 1030).

4. Edit the `Example()` method as shown below.

```
protected static void example() throws Exception {
      // create a new, empty XML document
      Doc2 libDoc = Doc2.createDocument();

      // create the root element <Library> and add it to the document
      LibraryType lib = libDoc.Library.append();

      // set the "LastUpdated" attribute
      com.altova.types.DateTime dt = new com.altova.types.DateTime(DateTime.now());
      lib.LastUpdated.setValue(dt);

      // create a new <Book> and populate its elements and attributes
      BookType book = lib.Book.append();
      book.ID.setValue(java.math.BigInteger.valueOf(1));
      book.Format.setEnumerationValue( BookFormatType.EPAPERBACK );
      book.Title.append().setValue("The XML Spy Handbook");
      book.Author.append().setValue("Altova");

      // create a dictionary (book of derived type)  and populate its elements and
attributes
      DictionaryType dict = new DictionaryType(lib.Book.append().getNode());
      dict.ID.setValue(java.math.BigInteger.valueOf(2));
      dict.Title.append().setValue("English-German Dictionary");
      dict.Format.setEnumerationValue(BookFormatType.EE_BOOK);
      dict.Author.append().setValue("John Doe");
      dict.FromLang.append().setValue("English");
      dict.ToLang.append().setValue("German");
      dict.setXsiType();

      // set the schema location (this is optional)
      libDoc.setSchemaLocation("Library.xsd");

      // save the XML document to a file with default encoding (UTF-8). "true" causes
the file to be pretty-printed.
      libDoc.saveToFile("Library1.xml", true);
   }
```

5. Build the Java project and run it. If the code is executed successfully, a **Library1.xml** file is created in the project directory.

Reading an XML document

1. On the **File** menu of Eclipse, click **Import**, select **Existing Projects into Workspace**, and click **Next**.
2. Next to **Select root directory**, click **Browse**, select the directory to which you generated the Java code, and then click **Finish**.
3. Save the code below as **Library1.xml** to a local directory (you will need to refer to the path of the **Library1.xml** file from the sample code below).

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample"
xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd"
LastUpdated="2016-02-03T17:10:08.4977404">
   <Book ID="1" Format="E-book">
      <Title>The XMLSpy Handbook</Title>
      <Author>Altova</Author>
   </Book>
   <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/LibrarySample"
xsi:type="n1:DictionaryType">
      <Title>English-German Dictionary</Title>
      <Author>John Doe</Author>
      <FromLang>English</FromLang>
      <ToLang>German</ToLang>
   </Book>
</Library>
```

4. In the Eclipse Package Explorer, expand the **com.LibraryTest** package and open the **LibraryTest.java** file.
5. Edit the `Example()` method as shown below.

```
protected static void example() throws Exception {
     // load XML document from a path, make sure to adjust the path as necessary
     Doc2 libDoc = Doc2.loadFromFile("Library1.xml");

     // get the first (and only) root element <Library>
     LibraryType lib = libDoc.Library.first();

     // check whether an element exists:
     if (!lib.Book.exists()) {
        System.out.println("This library is empty.");
        return;
     }

     // read a DateTime schema type
     com.altova.types.DateTime dt = lib.LastUpdated.getValue();
     System.out.println("The library was last updated on: " + dt.toDateString());

     // iteration: for each <Book>...
     for (java.util.Iterator itBook = lib.Book.iterator(); itBook.hasNext();) {
        BookType book = (BookType) itBook.next();
        // output values of ID attribute and (first and only) title element
        System.out.println("ID: " + book.ID.getValue());
        System.out.println("Title: " + book.Title.first().getValue());

        // read and compare an enumeration value
        if (book.Format.getEnumerationValue() == BookFormatType.EPAPERBACK)
           System.out.println("This is a paperback book.");

        // for each <Author>...
        for (java.util.Iterator itAuthor = book.Author.iterator(); itAuthor
              .hasNext();)
           System.out.println("Author: " + ((com.Doc.xs.stringType)
itAuthor.next()).getValue());
```

```
        // find the derived type of this book
        // by looking at the value of the xsi:type attribute, using DOM
        org.w3c.dom.Node bookNode = book.getNode();
        if (bookNode.getAttributes().getNamedItem("xsi:type") != null) {
           // Get the value of the xsi:type attribute
           String xsiTypeValue =
bookNode.getAttributes().getNamedItem("xsi:type").getNodeValue();

           // Get the namespace URI and lookup prefix of this namespace
           String namespaceUri = bookNode.getNamespaceURI();
           String lookupPrefix = bookNode.lookupPrefix(namespaceUri);

           // If xsi:type matches the namespace URI and type of the book node
           if (namespaceUri == "http://www.nanonull.com/LibrarySample"
                 && (  xsiTypeValue.equals(lookupPrefix + ":DictionaryType" )))    {
              // ...then this is a book of derived type (dictionary)
              DictionaryType dictionary = new DictionaryType(   book.getNode());
              // output the value of the "FromLang" and "ToLang" elements
              System.out.println("From language: " +
dictionary.FromLang.first().getValue());
              System.out.println("To language: " +
dictionary.ToLang.first().getValue());
           }
           else
           {
              // throw an error
              throw new java.lang.Error("This book has an unknown type.");
           }
        }
     }
   }
```

6. Build the Java project and run it. If the code is executed successfully, **Library1.xml** will be read by the program code, and its contents displayed in the Console view.

Reading and writing elements and attributes

Values of attributes and elements can be accessed using the `getValue()` method of the generated member element or attribute class, for example:

```
// output values of ID attribute and (first and only) title element
System.out.println("ID: " + book.ID.getValue());
System.out.println("Title: " + book.Title.first().getValue());
```

To get the value of the **Title** element in this particular example, we also used the `first()` method, since this is the first (and only) **Title** element of a book. For cases when you need to pick a specific element from a list by index, use the `at()` method.

To iterate through multiple elements, use either index-based iteration or `java.util.Iterator`. For example, you can iterate through the books of a library as follows:

```
// index-based iteration
for (int j = 0; j < lib.Book.count(); ++j ) {
   // your code here
}
```

```
// alternative iteration using java.util.Iterator
for (java.util.Iterator itBook = lib.Book.iterator(); itBook.hasNext();) {
      // your code here
   }
```

To add a new element, use the `append()` method. For example, the following code appends an empty root **Library** element to the document:

```
// create the root element <Library> and add it to the document
LibraryType lib = libDoc.Library.append();
```

Once an element is appended, you can set the value of any of its elements or an attributes by using the setValue() method.

```
// set the value of the Title element
book.Title.append().setValue("The XML Spy Handbook");
// set the value of the ID attribute
book.ID.setValue(java.math.BigInteger.valueOf(1));
```

Reading and writing enumeration values

If your XML schema defines simple types as enumerations, the enumerated values become available as `Enum` values in the generated code. In the schema used in this example, a book format can be hardcover, paperback, e-book, and so on. Therefore, in the generated code, these values would be available through an `Enum` (see the `BookFormatType` class diagram above). To assign enumeration values to an object, use code such as the one below:

```
// set an enumeration value
book.Format.setEnumerationValue( BookFormatType.EPAPERBACK );
```

You can read such enumeration values from XML instance documents as follows:

```
// read an enumeration value
if (book.Format.getEnumerationValue() == BookFormatType.EPAPERBACK)
         System.out.println("This is a paperback book."
```

When an "if" condition is not enough, create a switch to determine each enumeration value and process it as required.

Working with xs:dateTime and xs:duration types

If the schema from which you generated code uses time and duration types such as `xs:dateTime`, or `xs:duration`, these are converted to Altova native classes in generated code. Therefore, to write a date or duration value to the XML document, do the following:

1. Construct a `com.altova.types.DateTime` 1092 or `com.altova.types.Duration` 1096 object.
2. Set the object as value of the required element or attribute, for example:

```
// set the value of an attribute of DateTime type
com.altova.types.DateTime dt = new com.altova.types.DateTime(DateTime.now());
lib.LastUpdated.setValue(dt);
```

To read a date or duration from an XML document:

1. Declare the element value (or attribute) as com.altova.types.DateTime[1092] or com.altova.types.Duration[1096] object.
2. Format the required element or attribute, for example:

```
// read a DateTime type
com.altova.types.DateTime dt = lib.LastUpdated.getValue();
   System.out.println("The library was last updated on: " + dt.toDateString());
```

For more information, see com.altova.types.DateTime[1092] and com.altova.types.Duration[1096] class reference.

Working with derived types

If your XML schema defines derived types, you can preserve type derivation in XML documents that you create or load programmatically. Taking the schema used in this example, the following code listing illustrates how to create a new book of derived type `DictionaryType`:

```
// create a dictionary (book of derived type)  and populate its elements and attributes
DictionaryType dict = new DictionaryType(lib.Book.append().getNode());
dict.ID.setValue(java.math.BigInteger.valueOf(2));
dict.Title.append().setValue("English-German Dictionary");
dict.Format.setEnumerationValue(BookFormatType.EE_BOOK);
dict.Author.append().setValue("John Doe");
dict.FromLang.append().setValue("English");
dict.ToLang.append().setValue("German");
dict.setXsiType();
```

Note that it is important to set the `xsi:type` attribute of the newly created book. This ensures that the book type will be interpreted correctly by the schema when the XML document is validated.

When you load data from an XML document, the following code listing shows how to identify a book of derived type `DictionaryType` in the loaded XML instance. First, the code finds the value of the `xsi:type` attribute of the book node. If the namespace URI of this node is `http://www.nanonull.com/LibrarySample`, and if the URI lookup prefix and type matches the value of the `xsi:type` attribute, then this is a dictionary:

```
         // find the derived type of this book
         // by looking at the value of the xsi:type attribute, using DOM
         org.w3c.dom.Node bookNode = book.getNode();
         if (bookNode.getAttributes().getNamedItem("xsi:type") != null) {
            // Get the value of the xsi:type attribute
            String xsiTypeValue =
bookNode.getAttributes().getNamedItem("xsi:type").getNodeValue();

            // Get the namespace URI and lookup prefix of the book node
            String namespaceUri = bookNode.getNamespaceURI();
            String lookupPrefix = bookNode.lookupPrefix(namespaceUri);

            // If xsi:type matches the namespace URI and type of the book node
            if (namespaceUri == "http://www.nanonull.com/LibrarySample"
                  && (  xsiTypeValue.equals(lookupPrefix + ":DictionaryType" )))     {
               // ...then this is a book of derived type (dictionary)
               DictionaryType dictionary = new DictionaryType(   book.getNode());
```

```
            // output the value of the "FromLang" and "ToLang" elements
            System.out.println("From language: " +
dictionary.FromLang.first().getValue());
            System.out.println("To language: " +
dictionary.ToLang.first().getValue());
         }
         else
         {
            // throw an error
            throw new java.lang.Error("This book has an unknown type.");
         }
      }
```

27.2.6 Example: Purchase Order

This example illustrates how to work with program code generated from a "main" XML schema that imports other schemas. Each of the imported schema has a different target namespace. The goal here is to create programmatically an XML document where all elements are prefixed according to their namespace. More specifically, the XML document created from your C++, C#, or Java code should look like the one below:

```
<?xml version="1.0" encoding="utf-8"?>
<p:Purchase xsi:schemaLocation="http://NamespaceTest.com/Purchase Main.xsd"
            xmlns:p="http://NamespaceTest.com/Purchase"
            xmlns:o="http://NamespaceTest.com/OrderTypes"
            xmlns:c="http://NamespaceTest.com/CustomerTypes"
            xmlns:cmn="http://NamespaceTest.com/CommonTypes"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <p:OrderDetail>
        <o:Item>
            <o:ProductName>Lawnmower</o:ProductName>
            <o:Quantity>1</o:Quantity>
            <o:UnitPrice>148.42</o:UnitPrice>
        </o:Item>
    </p:OrderDetail>
    <p:PaymentMethod>VISA</p:PaymentMethod>
    <p:CustomerDetails>
        <c:Name>Alice Smith</c:Name>
        <c:DeliveryAddress>
            <cmn:Line1>123 Maple Street</cmn:Line1>
            <cmn:Line2>Mill Valley</cmn:Line2>
        </c:DeliveryAddress>
        <c:BillingAddress>
            <cmn:Line1>8 Oak Avenue</cmn:Line1>
            <cmn:Line2>Old Town</cmn:Line2>
        </c:BillingAddress>
    </p:CustomerDetails>
</p:Purchase>
```

The main schema used in this example is called **Main.xsd**. As illustrated in the code listing below, it imports three other schemas: **CommonTypes.xsd**, **CustomerTypes.xsd**, and **OrderTypes.xsd**. To get the same results as in this example, save all the code listings below to files, and use the same file names as above. Notice that the schema maps each of the prefixes `ord`, `pur`, `cmn`, and `cust` to some namespace (Order types, Purchase types, Common types, and Customer types, respectively). This means that, in the generated code,

the classes corresponding to Orders, Purchases, Customers, and so on, will be available under their respective namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://NamespaceTest.com/Purchase"
      xmlns:ord="http://NamespaceTest.com/OrderTypes"
      xmlns:pur="http://NamespaceTest.com/Purchase"
      xmlns:cmn="http://NamespaceTest.com/CommonTypes"
      xmlns:cust="http://NamespaceTest.com/CustomerTypes"
      elementFormDefault="qualified">
    <xs:import schemaLocation="CommonTypes.xsd"
namespace="http://NamespaceTest.com/CommonTypes" />
    <xs:import schemaLocation="CustomerTypes.xsd"
namespace="http://NamespaceTest.com/CustomerTypes" />
    <xs:import schemaLocation="OrderTypes.xsd"
namespace="http://NamespaceTest.com/OrderTypes" />
    <xs:element name="Purchase">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="OrderDetail" type="ord:OrderType" />
                <xs:element name="PaymentMethod" type="cmn:PaymentMethodType" />
                <xs:element ref="pur:CustomerDetails" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="CustomerDetails" type="cust:CustomerType" />
</xs:schema>
```

Main.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://NamespaceTest.com/CommonTypes"
      elementFormDefault="qualified">
   <xs:complexType name="AddressType">
      <xs:sequence>
         <xs:element name="Line1" type="xs:string"/>
         <xs:element name="Line2" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
   <xs:simpleType name="PriceType">
      <xs:restriction base="xs:decimal">
         <xs:fractionDigits value="2"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="PaymentMethodType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="VISA"/>
         <xs:enumeration value="MasterCard"/>
         <xs:enumeration value="Cash"/>
         <xs:enumeration value="AMEX"/>
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

CommonTypes.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://NamespaceTest.com/CustomerTypes"
      xmlns:cmn="http://NamespaceTest.com/CommonTypes"
      elementFormDefault="qualified">
    <xs:import schemaLocation="CommonTypes.xsd"
namespace="http://NamespaceTest.com/CommonTypes" />
    <xs:complexType name="CustomerType">
        <xs:sequence>
            <xs:element name="Name" type="xs:string" />
            <xs:element name="DeliveryAddress" type="cmn:AddressType" />
            <xs:element name="BillingAddress" type="cmn:AddressType" />
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

CustomerTypes.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://NamespaceTest.com/OrderTypes"
      xmlns:cmn="http://NamespaceTest.com/CommonTypes"
      elementFormDefault="qualified">
    <xs:import schemaLocation="CommonTypes.xsd"
namespace="http://NamespaceTest.com/CommonTypes" />
    <xs:complexType name="OrderType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" name="Item">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="ProductName" type="xs:string" />
                        <xs:element name="Quantity" type="xs:int" />
                        <xs:element name="UnitPrice" type="cmn:PriceType" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

OrderTypes.xsd

To complete this example, take the following steps:

1. Save all schemas from the code listings above to files on the disk, making sure that you preserve the indicated file names.
2. Generate the schema wrapper code from the **Main.xsd** schema above, using the steps described in Generating Code from XML Schemas or DTD [1021]. After completing this step, you should have generated a compilable program in the language of your choice (C++, C#, or Java).
3. Add code to your C++, C#, or Java program from one the following example code listings, as required:

   - XML Namespaces and Prefixes (C++) [1058]

- XML Namespaces and Prefixes (C#) 1059
- XML Namespaces and Prefixes (Java) 1060

27.2.6.1 XML Namespaces and Prefixes (C++)

After you generate code from the example schema 1055, a test C++ application is created, along with several supporting Altova libraries. Recall that the example schema (**Main.xsd**) has multiple namespace declarations. Consequently, the generated code includes namespaces that correspond to namespace aliases (prefixes) from the schema, namely: `Main::ord`, `Main::pur`, `Main::cmn`, and `Main::cust`.

In general, in order to control XML namespaces and prefixes with the help of the schema wrapper libraries, you have the following methods at your disposal:

- DeclareAllNamespacesFromSchema() 1071. Call this method if you want to declare the same namespaces in your XML instance as in the schema. Otherwise, if you need different namespaces as in this example, then `DeclareNamespace()` should be used. The method `DeclareAllNamespacesFromSchema()` is not used in this example because we specifically want to create XML elements with prefixes that are slightly different from those declared in the schema.
- DeclareNamespace() 1074. Call this method to create or override the existing namespace prefix attribute on an element. The element must already be created using either the `append()` or `appendWithPrefix()` methods, as further illustrated below.
- appendWithPrefix(). 1075 Use this method to append an instance element with a specific prefix. To create the XML instance illustrated in this example, it was sufficient to call this method for the root element only. All other elements were appended using just append() 1075, and their prefixes were added automatically based on their namespaces, according to the rules above.

The code listing below shows you how to create an XML document with multiple namespace declarations and prefixed element names. Specifically, it generates a Purchase Order instance as illustrated in the Example: Purchase Order 1055. Importantly, for illustrative purposes, some prefixes are overridden in the XML instance (that is, they are not exactly the same as the ones declared in the schema).

```
void Example()
{
   // Create the XML document and append the root element
   Main::pur::CMain doc = Main::pur::CMain::CreateDocument();
   Main::pur::CPurchaseType purchase = doc.Purchase.appendWithPrefix(_T("p"));

   // Set schema location
   doc.SetSchemaLocation(_T("Main.xsd"));

   // Declare namespaces on root element
   purchase.DeclareNamespace(_T("o"), _T("http://NamespaceTest.com/OrderTypes"));
   purchase.DeclareNamespace(_T("c"), _T("http://NamespaceTest.com/CustomerTypes"));
   purchase.DeclareNamespace(_T("cmn"), _T("http://NamespaceTest.com/CommonTypes"));

   // Append the OrderDetail element
   Main::ord::COrderType order = purchase.OrderDetail.append();
   Main::ord::CItemType item = order.Item.append();
   item.ProductName.append() = _T("Lawnmower");
   item.Quantity.append() = 1;
   item.UnitPrice.append() = 148.42;

   // Append the PaymentMethod element
   Main::cmn::CPaymentMethodTypeType paymentMethod = purchase.PaymentMethod.append();
```

```
   paymentMethod.SetEnumerationValue(Main::cmn::CPaymentMethodTypeType::k_VISA);

   // Append the CustomerDetails element
   Main::cust::CCustomerType customer = purchase.CustomerDetails.append();
   customer.Name.append() = _T("Alice Smith");
   Main::cmn::CAddressType deliveryAddress = customer.DeliveryAddress.append();
   deliveryAddress.Line1.append() = _T("123 Maple Street");
   deliveryAddress.Line2.append() = _T("Mill Valley");
   Main::cmn::CAddressType billingAddress = customer.BillingAddress.append();
   billingAddress.Line1.append() = _T("8 Oak Avenue");
   billingAddress.Line2.append() = _T("Old Town");

   // Save to file and release object from memory
   doc.SaveToFile(_T("Main1.xml"), true);
   doc.DestroyDocument();
}
```

27.2.6.2 XML Namespaces and Prefixes (C#)

After you generate code from the example schema[1055], a test C# application is created, along with several supporting Altova libraries. Recall that the example schema (**Main.xsd**) has multiple namespace declarations. Consequently, the generated code includes namespaces that correspond to namespace aliases (prefixes) from the schema, namely: `Main.ord`, `Main.pur`, `Main.cmn`, and `Main.cust`.

In general, in order to control XML namespaces and prefixes with the help of the schema wrapper libraries, you have the following methods at your disposal:

- DeclareAllNamespacesFromSchema()[1087]. Call this method if you want to declare the same namespaces in your XML instance as in the schema. Otherwise, if you need different namespaces as in this example, then `DeclareNamespace()` should be used. The method `DeclareAllNamespacesFromSchema()` is not used in this example because we specifically want to create XML elements with prefixes that are slightly different from those declared in the schema.
- DeclareNamespace()[1089]. Call this method to create or override the existing namespace prefix attribute on an element. The element must already be created using either the `Append()` or `AppendWithPrefix()` methods, as further illustrated below.
- AppendWithPrefix().[1090] Use this method to append an instance element with a specific prefix. To create the XML instance illustrated in this example, it was sufficient to call this method for the root element only. All other elements were appended using just Append()[1090], and their prefixes were added automatically based on their namespaces, according to the rules above.

The code listing below shows you how to create an XML document with multiple namespace declarations and prefixed element names. Specifically, it generates a Purchase Order instance as illustrated in the Example: Purchase Order[1055]. Importantly, for illustrative purposes, some prefixes are overridden in the XML instance (that is, they are not exactly the same as the ones declared in the schema).

```
protected static void Example()
{
   // Create the XML document and append the root element
   pur.Main2 doc = pur.Main2.CreateDocument();
   pur.PurchaseType purchase = doc.Purchase.AppendWithPrefix("p");

   // Set schema location
   doc.SetSchemaLocation(@"Main.xsd");
```

```
    // Declare namespaces on root element
    purchase.DeclareNamespace("o", "http://NamespaceTest.com/OrderTypes");
    purchase.DeclareNamespace("c", "http://NamespaceTest.com/CustomerTypes");
    purchase.DeclareNamespace("cmn", "http://NamespaceTest.com/CommonTypes");

    // Append the OrderDetail element
    ord.OrderType order = purchase.OrderDetail.Append();
    ord.ItemType item = order.Item.Append();
    item.ProductName.Append().Value = "Lawnmower";
    item.Quantity.Append().Value = 1;
    item.UnitPrice.Append().Value = 148.42M;

    // Append the PaymentMethod element
    cmn.PaymentMethodTypeType paymentMethod = purchase.PaymentMethod.Append();
    paymentMethod.EnumerationValue = cmn.PaymentMethodTypeType.EnumValues.eVISA;

    // Append the CustomerDetails element
    cust.CustomerType customer = purchase.CustomerDetails.Append();
    customer.Name.Append().Value = "Alice Smith";
    cmn.AddressType deliveryAddress = customer.DeliveryAddress.Append();
    deliveryAddress.Line1.Append().Value = "123 Maple Street";
    deliveryAddress.Line2.Append().Value = "Mill Valley";
    cmn.AddressType billingAddress = customer.BillingAddress.Append();
    billingAddress.Line1.Append().Value = "8 Oak Avenue";
    billingAddress.Line2.Append().Value = "Old Town";

    // Save to file
    doc.SaveToFile("PurchaseOrder.xml", true);
}
```

27.2.6.3 XML Namespaces and Prefixes (Java)

After you generate code from the example schema[1055], a test Java application is created, along with several supporting Altova libraries. Recall that the example schema (**Main.xsd**) has multiple namespace declarations. Consequently, the generated code includes namespaces that correspond to namespace aliases (prefixes) from the schema, namely: `com.Main.ord`, `com.Main.pur`, `com.Main.cmn`, and `com.Main.cust`.

In general, in order to control XML namespaces and prefixes with the help of the schema wrapper libraries, you have the following methods at your disposal:

- declareAllNamespacesFromSchema()[1102]. Call this method if you want to declare the same namespaces in your XML instance as in the schema. Otherwise, if you need different namespaces as in this example, then `declareNamespace()` should be used. The method `declareAllNamespacesFromSchema()` is not used in this example because we specifically want to create XML elements with prefixes that are slightly different from those declared in the schema.
- declareNamespace()[1103]. Call this method to create or override the existing namespace prefix attribute on an element. The element must already be created using either the `append()` or `appendWithPrefix()` methods, as further illustrated below.
- appendWithPrefix().[1105] Use this method to append an instance element with a specific prefix. To create the XML instance illustrated in this example, it was sufficient to call this method for the root element only. All other elements were appended using just append()[1105], and their prefixes were added automatically based on their namespaces, according to the rules above.

The code listing below shows you how to create an XML document with multiple namespace declarations and prefixed element names. Specifically, it generates a Purchase Order instance as illustrated in the Example:

Purchase Order[1055]. Importantly, for illustrative purposes, some prefixes are overridden in the XML instance (that is, they are not exactly the same as the ones declared in the schema).

```
protected static void example() throws Exception {
   // Create the XML document and append the root element
   com.Main.pur.Main2 doc = com.Main.pur.Main2.createDocument();
   com.Main.pur.PurchaseType purchase = doc.Purchase.appendWithPrefix("p");

   // Set schema location
   doc.setSchemaLocation("Main.xsd");

   // Declare namespaces on root element
   purchase.declareNamespace("o", "http://NamespaceTest.com/OrderTypes");
   purchase.declareNamespace("c", "http://NamespaceTest.com/CustomerTypes");
   purchase.declareNamespace("cmn", "http://NamespaceTest.com/CommonTypes");

   // Append the OrderDetail element
   com.Main.ord.OrderType order = purchase.OrderDetail.append();
   com.Main.ord.ItemType item = order.Item.append();
   item.ProductName.append().setValue("Lawnmower");
   item.Quantity.append().setValue(1);
   java.math.BigDecimal price = new java.math.BigDecimal("148.42");
   item.UnitPrice.append().setValue(price);

   // Append the PaymentMethod element
   com.Main.cmn.PaymentMethodTypeType paymentMethod = purchase.PaymentMethod.append();
   paymentMethod.setEnumerationValue(com.Main.cmn.PaymentMethodTypeType.EVISA);

   // Append the CustomerDetails element
   com.Main.cust.CustomerType customer = purchase.CustomerDetails.append();
   customer.Name.append().setValue("Alice Smith");
   com.Main.cmn.AddressType deliveryAddress = customer.DeliveryAddress.append();
   deliveryAddress.Line1.append().setValue("123 Maple Street");
   deliveryAddress.Line2.append().setValue("Mill Valley");
   com.Main.cmn.AddressType billingAddress = customer.BillingAddress.append();
   billingAddress.Line1.append().setValue("8 Oak Avenue");
   billingAddress.Line2.append().setValue("Old Town");

   // Save to file
   doc.saveToFile("PurchaseOrder.xml", true);
}
```

27.3 Reference to Generated Classes (C++)

This chapter includes a description of C++ classes generated with XMLSpy from a DTD or XML schema (see Generating Code from XML Schemas or DTDs 1021). You can integrate these classes into your code to read, modify, and write XML documents.

Note: The generated code does include other supporting classes, which are not listed here and are subject to modification.

27.3.1 altova::DateTime

This class enables you to process XML attributes or elements that have date and time types, such as `xs:dateTime`.

Constructors

Name	Description
`DateTime()`	Initializes a new instance of the `DateTime` class to 12:00:00 midnight, January 1, 0001.
`DateTime(__int64 value, short timezone)`	Initializes a new instance of the `DateTime` class. The `value` parameter represents the number of ticks (100-nanosecond intervals) that have elapsed since 12:00:00 midnight, January 1, 0001.
`DateTime(int year, unsigned char month, unsigned char day, unsigned char hour, unsigned char minute, double second)`	Initializes a new instance of the `DateTime` class to the year, month, day, hour, minute, and second supplied as argument.
`DateTime(int year, unsigned char month, unsigned char day, unsigned char hour, unsigned char minute, double second, short timezone)`	Initializes a new instance of the `DateTime` class to the year, month, day, hour, minute, second and timezone supplied as argument. The timezone is expressed in minutes and can be positive or negative. For example, the timezone "UTC-01:00" is expressed as "-60".

Methods

Name	Description
`unsigned char Day() const`	Returns the day of month of the current `DateTime` object. The return values range from 1 through 31.
`int DayOfYear() const`	Returns the day of year of the current `DateTime` object. The return values range from 1 through 366.
`bool HasTimezone() const`	Returns Boolean **true** if the current `DateTime` object has a timezone defined; **false** otherwise.

(C) 2015-2021 Altova GmbH

Name	Description
`unsigned char Hour() const`	Returns the hour of the current `DateTime` object. The return values range from 0 through 23.
`static bool IsLeapYear(int year)`	Returns Boolean **true** if the year of the `DateTime` class is a leap year; **false** otherwise.
`unsigned char Minute() const`	Returns the minute of the current `DateTime` object. The return values range from 0 through 59.
`unsigned char Month() const`	Returns the month of the current `DateTime` object. The return values range from 1 through 12.
`__int64 NormalizedValue() const`	Returns the value of the `DateTime` object expressed as the Coordinated Universal Time (UTC).
`double Second() const`	Returns the second of the current `DateTime` object. The return values range from 0 through 59.
`void SetTimezone(short tz)`	Sets the timezone of the current `DateTime` object to the timezone value supplied as argument. The **tz** argument is expressed in minutes and can be positive or negative.
`short Timezone() const`	Returns the timezone, in minutes, of the current `DateTime` object. Before using this method, make sure that the object actually has a timezone, by calling the `HasTimezone()` method.
`__int64 Value() const`	Returns the value of the `DateTime` object, expressed in the number of ticks (100-nanosecond intervals) that have elapsed since 12:00:00 midnight, January 1, 0001.
`int Weekday() const`	Returns the day of week of the current `DateTime` object, as an integer. Values range from 0 through 6, where 0 is Monday (ISO-8601).
`int Weeknumber() const`	Returns the number of week in the year of the current `DateTime` object. The return values are according to ISO-8601.
`int WeekOfMonth() const`	Returns the number of week in the month of the current `DateTime` object. The return values are according to ISO-8601.
`int Year() const`	Returns the year of the current `DateTime` object.

Example

```
void Example()
{
   // initialize a new DateTime instance to 12:00:00 midnight, January 1st, 0001
   altova::DateTime dt1 = altova::DateTime();

   // initialize a new DateTime instance using the year, month, day, hour, minute, and
second
   altova::DateTime dt2 = altova::DateTime(2015, 11, 10, 9, 8, 7);

   // initialize a new DateTime instance using the year, month, day, hour, minute,
```

```
second, and UTC +01:00 timezone
    altova::DateTime dt = altova::DateTime(2015, 11, 22, 13, 53, 7, 60);

   // Get the value of this DateTime object
   std::cout << "The number of ticks of the DateTime object is: " << dt.Value() <<
std::endl;

   // Get the year
   cout << "The year is: " << dt.Year() << endl;
   // Get the month
   cout << "The month is: " << (int)dt.Month() << endl;
   // Get the day of the month
   cout << "The day of the month is: " << (int) dt.Day() << endl;
   // Get the day of the year
   cout << "The day of the year is: " << dt.DayOfYear() << endl;
   // Get the hour
   cout << "The hour is: " << (int) dt.Hour() << endl;
   // Get the minute
   cout << "The minute is: " << (int) dt.Minute() << endl;
   // Get the second
   cout << "The second is: " << dt.Second() << endl;
   // Get the weekday
   cout << "The weekday is: " << dt.Weekday() << endl;
   // Get the week number
   cout << "The week of year is: " << dt.Weeknumber() << endl;
   // Get the week in month
   cout << "The week of month is: " << dt.WeekOfMonth() << endl;

   // Check whether a DateTime instance has a timezone
   if (dt.HasTimezone() == TRUE)
   {
      // output the value of the Timezone
      cout <<  "The timezone is: " << dt.Timezone() << endl;
   }
   else
   {
      cout <<  "No timezone has been defined." << endl;
   }

   // Construct a DateTime object with a timezone UTC+01:00 (Vienna)
   altova::DateTime vienna_dt = DateTime(2015, 11, 23, 14, 30, 59, +60);
   // Output the result in readable format
   cout << "The Vienna time: "
       << (int) vienna_dt.Month()
       << "-" << (int) vienna_dt.Day()
       << " " << (int) vienna_dt.Hour()
       << ":" << (int) vienna_dt.Minute()
       << ":" << (int) vienna_dt.Second()
       << endl;

   // Convert the value to UTC time
   DateTime utc_dt = DateTime(vienna_dt.NormalizedValue());
   // Output the result in readable format
   cout << "The UTC time:    "
      << (int) utc_dt.Month()
      << "-" << (int) utc_dt.Day()
      << " " << (int) utc_dt.Hour()
      << ":" << (int) utc_dt.Minute()
      << ":" << (int) utc_dt.Second()
```

```
      << endl;

   // Check if a year is a leap year
   int year = 2016;
   if( altova::DateTime::IsLeapYear(year) )
   { cout << year << " is a leap year" << endl; }
   else
   { cout << year << " is not a leap year" << endl; }
}
```

27.3.2 altova::Duration

This class enables you to process XML attributes or elements of type `xs:duration`.

Constructors

Name	Description
`Duration()`	Initializes a new instance of the `Duration` class to an empty value.
`Duration(const DayTimeDuration& dt)`	Initializes a new instance of the `Duration` class to a duration defined by the `dt` argument (see altova::DayTimeDuration 1067).
`Duration(const YearMonthDuration& ym)`	Initializes a new instance of the `Duration` class to the duration defined by the `ym` argument (see altova::YearMonthDuration 1068).
`Duration(const YearMonthDuration& ym, const DayTimeDuration& dt)`	Initializes a new instance of the `Duration` class to the duration defined by both the `dt` and the `ym` arguments (see altova::YearMonthDuration 1068 and altova::DayTimeDuration 1067).

Methods

Name	Description
`int Days() const`	Returns the number of days in the current `Duration` instance.
`DayTimeDuration DayTime() const`	Returns the day and time duration in the current `Duration` instance, expressed as a `DayTimeDuration` object (see altova::DayTimeDuration 1067).
`int Hours() const`	Returns the number of hours in the current `Duration` instance.
`bool IsNegative() const`	Returns Boolean **true** if the current `Duration` instance is negative.
`bool IsPositive() const`	Returns Boolean **true** if the current `Duration` instance is positive.
`int Minutes() const`	Returns the number of minutes in the current `Duration` instance.
`int Months() const`	Returns the number of months in the current `Duration` instance.
`double Seconds() const`	Returns the number of seconds in the current `Duration` instance.

Name	Description
`YearMonthDuration YearMonth() const`	Returns the year and month duration in the current `Duration` instance, expressed as a `YearMonthDuration` object (see altova::YearMonthDuration 1068).
`int Years() const`	Returns the number of years in the current `Duration` instance.

Example

The following code listing illustrates creating a new `Duration` object, as well as reading values from it.

```
void ExampleDuration()
{
   // Create an empty Duration object
   altova::Duration empty_duration = altova::Duration();

   // Create a Duration object using an existing duration value
   altova::Duration duration1 = altova::Duration(empty_duration);

   // Create a YearMonth duration of six years and five months
   altova::YearMonthDuration yrduration = altova::YearMonthDuration(6, 5);

   // Create a DayTime duration of four days, three hours, two minutes, and one second
   altova::DayTimeDuration dtduration = altova::DayTimeDuration(4, 3, 2, 1);

   // Create a Duration object by combining the two previously created durations
   altova::Duration duration = altova::Duration(yrduration, dtduration);

   // Get the number of years in this Duration instance
   cout << "Years:  " << duration.Years() << endl;

   // Get the number of months in this Duration instance
   cout << "Months: " << duration.Months() << endl;

   // Get the number of days in this Duration instance
   cout << "Days:   " << duration.Days() << endl;

   // Get the number of hours in this Duration instance
   cout << "Hours:  " << duration.Hours() << endl;

   // Get the number of hours in this Duration instance
   cout << "Minutes: " << duration.Minutes() << endl;

   // Get the number of seconds in this Duration instance
   cout << "Seconds: " << duration.Seconds() << endl;
}
```

27.3.3 altova::DayTimeDuration

This class enables you to process XML schema duration types that consist of a day and time part.

Constructors

Name	Description
`DayTimeDuration()`	Initializes a new instance of the `DayTimeDuration` class to an empty value.
`DayTimeDuration(int days, int hours, int minutes, double seconds)`	Initializes a new instance of the `DayTimeDuration` class to the number of days, hours, minutes, and seconds supplied as arguments.
`explicit DayTimeDuration(__int64 value)`	Initializes a new instance of the `DayTimeDuration` class to a duration that consists of as many ticks (100-nanosecond intervals) as supplied in the **value** argument.

Methods

Name	Description
`int Days() const`	Returns the number of days in the current `DayTimeDuration` instance.
`int Hours() const`	Returns the number of hours in the current `DayTimeDuration` instance.
`bool IsNegative() const`	Returns Boolean **true** if the current `DayTimeDuration` instance is negative.
`bool IsPositive() const`	Returns Boolean **true** if the current `DayTimeDuration` instance is positive.
`int Minutes() const`	Returns the number of minutes in the current `DayTimeDuration` instance.
`double Seconds() const`	Returns the number of seconds in the current `DayTimeDuration` instance.
`__int64 Value() const`	Returns the value (in ticks) of the current `DayTimeDuration` instance.

27.3.4 altova::YearMonthDuration

This class enables you to process XML schema duration types that consist of a year and month part.

Constructors

Name	Description
`YearMonthDuration()`	Initializes a new instance of the `YearMonthDuration` class to an empty value.
`YearMonthDuration(int years, int months)`	Initializes a new instance of the `YearMonthDuration` class to the number of years and months supplied in the **years** and **months** arguments.
`explicit YearMonthDuration(int value)`	Initializes a new instance of the `YearMonthDuration` class to a duration that consists of as many ticks (100-nanosecond intervals) as supplied in the **value** argument.

Methods

Name	Description
`bool IsNegative() const`	Returns Boolean **true** if the current `YearMonthDuration` instance is negative.
`bool IsPositive() const`	Returns Boolean **true** if the current `YearMonthDuration` instance is positive.
`int Months() const`	Returns the number of months in the current `YearMonthDuration` instance.
`int Value() const`	Returns the value (in ticks) of the current `YearMonthDuration` instance.
`int Years()`	Returns the number of years in the current `YearMonthDuration` instance.

27.3.5 altova::meta::Attribute

This class enables you to access schema information about classes generated from attributes. Note that this class is not meant to provide dynamic information about particular instances of an attribute in an XML document. Instead, it enables you to obtain programmatically information about a particular attribute defined in the XML schema.

Methods

Name	Description
`SimpleType GetDataType()`	Returns the type of the attribute content.

(C) 2015-2021 Altova GmbH

Name	Description
`string_type GetLocalName()`	Returns the local name of the attribute.
`string_type GetNamespaceURI()`	Returns the namespace URI of the attribute.
`bool IsRequired()`	Returns true if the attribute is required.

Operators

Name	Description
`bool operator()`	Returns true if this is not the NULL Attribute.
`bool operator!()`	Returns true if this is the NULL Attribute.

27.3.6 altova::meta::ComplexType

This class enables you to access schema information about classes generated from complex types. Note that this class is not meant to provide dynamic information about particular instances of a complex type in an XML document. Instead, it enables you to obtain programmatically information about a particular complex type defined in the XML schema.

Methods

Name	Description
`Attribute FindAttribute(const char_type* localName, const char_type* namespaceURI)`	Finds the attribute with the specified local name and namespace URI.
`Element FindElement(const char_type* localName, const char_type* namespaceURI)`	Finds the element with the specified local name and namespace URI.
`std::vector<Attribute> GetAttributes()`	Returns a list of all attributes.
`ComplexType GetBaseType()`	Returns the base type of this type.
`SimpleType GetContentType()`	Returns the simple type of the content.
`std::vector<Element> GetElements()`	Returns a list of all elements.
`string_type GetLocalName()`	Returns the local name of the type.
`string_type GetNamespaceURI()`	Returns the namespace URI of the type.

Operators

Name	Description
`bool operator()`	Returns true if this is not the NULL ComplexType.
`bool operator!()`	Returns true if this is the NULL ComplexType.

27.3.7 altova::meta::Element

This class enables you to access information about classes generated from schema elements. Note that this class is not meant to provide dynamic information about particular instances of an element in an XML document. Instead, it enables you to obtain programmatically information about a particular element defined in the XML schema.

Methods

Name	Description
`ComplexType GetDataType()`	Returns the type of the element. Note that this is always a complex type even if declared as simple in the original schema. Use GetContentType() of the returned object to get the simple content type.
`string_type GetLocalName()`	Returns the local name of the element.
`unsigned int GetMaxOccurs()`	Returns the maxOccurs value defined in the schema.
`unsigned int GetMinOccurs()`	Returns the minOccurs value defined in the schema.
`string_type GetNamespaceURI()`	Returns the namespace URI of the element.

Operators

Name	Description
`bool operator()`	Returns true if this is not the NULL Element.
`bool operator!()`	Returns true if this is the NULL Element.

27.3.8 altova::meta::SimpleType

This class enables you to access schema information about classes generated from simple types. Note that this class is not meant to provide dynamic information about particular instances of simple types in an XML document. Instead, it enables you to obtain programmatically information about a particular simple type defined in the XML schema.

(C) 2015-2021 Altova GmbH

Methods

Name	Description
`SimpleType GetBaseType()`	Returns the base type of this type.
`std::vector<string_type> GetEnumerations()`	Returns a list of all enumeration facets.
`unsigned int GetFractionDigits()`	Returns the value of this facet.
`unsigned int GetLength()`	Returns the value of this facet.
`string_type GetLocalName()`	Returns the local name of the type.
`string_type GetMaxExclusive()`	Returns the value of this facet.
`string_type GetMaxInclusive()`	Returns the value of this facet.
`unsigned int GetMaxLength()`	Returns the value of this facet.
`string_type GetMinExclusive()`	Returns the value of this facet.
`string_type GetMinInclusive()`	Returns the value of this facet.
`unsigned int GetMinLength()`	Returns the value of this facet.
`string_type GetNamespaceURI()`	Returns the namespace URI of the type.
`std::vector<string_type> GetPatterns()`	Returns a list of all pattern facets.
`unsigned int GetTotalDigits()`	Returns the value of this facet.
`WhitespaceType GetWhitespace()`	Returns the value of the whitespace facet, which is one of: • Whitespace_Unknown • Whitespace_Preserve • Whitespace_Replace • Whitespace_Collapse

Operators

Name	Description
`bool operator()`	Returns true if this is not the NULL SimpleType.
`bool operator!()`	Returns true if this is the NULL SimpleType.

27.3.9 [YourSchema]::[CDoc]

When code is generated from an XML Schema, the generated code provides a document class with the same name as the schema. This class contains all possible root elements as members, as well as the following

methods. Note that, in the method names below, "CDoc" stands for the name of the generated document class itself.

Methods

Name	Description
`static CDoc CreateDocument()`	Creates a new, empty XML document. Must be released using DestroyDocument().
`static void DeclareAllNamespacesFromSchema(ElementType& node)`	Declares all namespaces from the XML Schema on the element supplied as argument (typically, the XML root element). Calling this method is useful if your schema has multiple namespace declarations, each mapped to a prefix, and you would like to declare all of them on the element supplied as argument.
`void DestroyDocument()`	Destroys a document. All references to the document and its nodes are invalidated. This must be called when you finished working with a document.
`static CDoc LoadFromBinary(const std:vector<unsigned char>& xml)`	Loads an XML document from a byte array.
`static CDoc LoadFromFile(const string_type& fileName)`	Loads an XML document from a file.
`static CDoc LoadFromString(const string_type& xml)`	Loads an XML document from a string.
`std::vector<unsigned char> SaveToBinary(bool prettyPrint)`	Saves an XML document to a byte array. When set to true, the `prettyPrint` argument re-formats the XML document for better readability.
`std::vector<unsigned char> SaveToBinary(bool prettyPrint, const string_type & encoding)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding.
`std::vector<unsigned char> SaveToBinary(bool prettyPrint, const string_type & encoding, bool bBigEndian, bool bBOM)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding. Byte order and Unicode byte-order mark can be specified for Unicode encodings.
`void SaveToFile(const string_type & fileName, bool prettyPrint)`	Saves an XML document to a file, with optional "pretty-print" formatting.
`void SaveToFile(const string_type & fileName, bool omitXmlDecl)`	Saves an XML document to a file. If the `omitXmlDecl` argument is set to true, the XML declaration will not be written.
`void SaveToFile(const string_type & fileName, bool omitXmlDecl, const string_type & encoding)`	Saves an XML document to a file with the specified encoding. If the `omitXmlDecl` argument is set to true, the XML declaration will not be written.
`void SaveToFile(const string_type & fileName, bool prettyPrint, bool omitXmlDecl, const string_type & encoding, bool bBigEndian, bool bBOM)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding. Byte order and Unicode byte-order mark can be specified for Unicode encodings.

(C) 2015-2021 Altova GmbH

Name	Description
`void SaveToFile(const string_type & fileName, bool prettyPrint, bool omitXmlDecl, const string_type & encoding, bool bBigEndian, bool bBOM, const string_type & lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding and the specified line end. Byte order and Unicode byte-order mark can be specified for Unicode encodings. This method is only available if you generated the code for the Xerces3 XML library (see Code Generator Options 1108).
`void SaveToFile(const string_type& fileName, bool prettyPrint, bool omitXmlDecl, const string_type & encoding, const string_type & lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding and the specified line end. This method is only available if you generated the code for the Xerces3 XML library (see Code Generator Options 1108).
`void SaveToFile(const string_type & fileName, bool prettyPrint, const string_type & encoding)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding.
`void SaveToFile(const string_type& fileName, bool prettyPrint, const string_type & encoding, bool bBigEndian, bool bBOM)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding. Byte order and Unicode byte-order mark can be specified for Unicode encodings.
`void SaveToFile(const string_type& fileName, bool prettyPrint, const string_type & encoding, bool bBigEndian, bool bBOM, const string_type & lineend)`	Saves an XML document to a file with the specified encoding and the specified line end. Byte order and Unicode byte-order mark can be specified for Unicode encodings. This method is only available if you generated the code for the Xerces3 XML library (see Code Generator Options 1108).
`void SaveToFile(const string_type& fileName, bool prettyPrint, const string_type & encoding, const string_type & lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding and the specified line end. This method is only available if you generated the code for the Xerces3 XML library (see Code Generator Options 1108).
`string_type SaveToString(bool prettyPrint)`	Saves an XML document to a string, with optional "pretty-print" formatting.
`string_type SaveToString(bool prettyPrint, bool omitXmlDecl)`	Saves an XML document to a string, with optional "pretty-print" formatting. If the `omitXmlDecl` argument is set to true, the XML declaration will not be written.
`void SetDTDLocation(const string_type & dtdLocation)`	Adds a DOCTYPE declaration with the specified system ID. A root element must already exist. This method is not supported for MSXML, since it is not possible to add a DOCTYPE declaration to a document in memory.
`void SetSchemaLocation(const string_type & schemaLocation)`	Adds an xsi:schemaLocation or xsi:noNamespaceSchemaLocation attribute to the root element. A root element must already exist.

27.3.10 [YourSchema]::[ElementType]

This class provides methods for manipulating XML elements from your schema. Methods of this class can be called on elements, not on the XML document itself. Note that, in order to call methods of this class, you don't need to instantiate the class directly. Any element created using the **append()** or **appendWithPrefix()** methods is of `[ElementType]` type.

Methods

Name	Description

`void DeclareNamespace(const string_type prefix, const string_type nsURI)`	This method takes two arguments that are both of string type: the prefix and the namespace URI that you want to use. The prefix supplied as argument will be mapped to the namespace URI value supplied as argument. If the prefix supplied as argument is empty, the method creates or overrides the default namespace declaration in the element.
	
	For example, let's assume that the XML document has an XML element called "purchase". If you call
	
	`purchase.DeclareNamespace(_T("ord"), _T("http://OrderTypes"));`
	
	then the XML document becomes
	
	`<purchase xmlns:ord="http://OrderTypes" />`
	
	Another example, if you call:
	
	`purchase.DeclareNamespace(_T(""), _T("http://OrderTypes"));`
	
	then the XML document becomes
	
	`<purchase xmlns="http://OrderTypes" />`
	
	Note: The declared namespace is used when appending subsequent child elements or attributes, according to the following rules:
	
	1. If the child namespace is the default, then use empty prefix.
	2. If the child namespace is equal to the parent one, then use the parent prefix.
	3. Otherwise, search for nearest prefix from parent to top, using the lookup algorithm described in section "B.2: Namespace Prefix Lookup" at https://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20021022/namespaces-algorithms.html.

Name	Description
	4. If there is no prefix for element namespace found, then use empty prefix.

27.3.11 [YourSchema]::MemberAttribute

When code is generated from an XML schema, a class such as this one is created for each member attribute of a type.

Methods

Name	Description
`bool exists()`	Returns true if the attribute exists.
`int GetEnumerationValue()`	Generated for enumeration types only. Returns one of the constants generated for the possible values, or "Invalid" if the value does not match any of the enumerated values in the schema.
`altova::meta::Attribute info()`	Returns an object for querying schema information (see `altova::meta::Attribute` 1068).
`void remove()`	Removes the attribute from its parent element.
`void SetEnumerationValue(int)`	Generated for enumeration types only. Pass one of the constants generated for the possible values to this method to set the value.

27.3.12 [YourSchema]::MemberElement

When code is generated from an XML schema, a class such as this one is created for each member element of a type. In the descriptions below, "MemberType" stands for the name of the member element itself.

Methods

Name	Description
`Iterator<MemberType> all()`	Returns an object for iterating instances of the member element.
`MemberType append()`	Creates a new element and appends it to its parent.
`MemberType appendWithPrefix(string_type prefix)`	Creates a new element having the prefix supplied as argument, and appends it to its parent. For an example, see Example: Purchase Order 1055.
`unsigned int count()`	Returns the count of elements.

Name	Description
`int GetEnumerationValue()`	Generated for enumeration types only. Returns one of the constants generated for the possible values, or **Invalid** if the value does not match any of the enumerated values in the schema.
`bool exists()`	Returns true if at least one element exists.
`MemberType first()`	Returns the first instance of the member element.
`MemberType operator[](unsigned int index)`	Returns the member element specified by the index.
`altova::meta::Element info()`	Returns an object for querying schema information (see `altova::meta::Element` 1070).
`MemberType last()`	Returns the last instance of the member element.
`void remove()`	Deletes all occurrences of the element from its parent.
`void removeAt(unsigned int index)`	Deletes the occurrence of the element specified by the index.
`void SetEnumerationValue(int)`	Generated for enumeration types only. Pass one of the constants generated for the possible values to this method to set the value.

(C) 2015-2021 Altova GmbH

27.4 Reference to Generated Classes (C#)

This chapter includes a description of C# classes generated with XMLSpy from a DTD or XML schema (see Generating Code from XML Schemas or DTDs 1021). You can integrate these classes into your code to read, modify, and write XML documents.

Note: The generated code does include other supporting classes, which are not listed here and are subject to modification.

27.4.1 Altova.Types.DateTime

This class enables you to process XML attributes or elements that have date and time types, such as `xs:dateTime`.

Constructors

	Name	Description
	`DateTime(DateTime obj)`	Initializes a new instance of the `DateTime` class to the `DateTime` object supplied as argument.
	`DateTime(System.DateTime newvalue)`	Initializes a new instance of the `DateTime` class to the `System.DateTime` object supplied as argument.
	`DateTime(int year, int month, int day, int hour, int minute, double second, int offsetTZ)`	Initializes a new instance of the `DateTime` class to the year, month, day, hour, minute, second, and timezone offset supplied as arguments.
	`DateTime(int year, int month, int day, int hour, int minute, double second)`	Initializes a new instance of the `DateTime` class to the year, month, day, hour, minute, and second supplied as arguments.
	`DateTime(int year, int month, int day)`	Initializes a new instance of the `DateTime` class to the year, month and day supplied as arguments.

Properties

	Name	Description
	`bool HasTimezone`	Gets a Boolean value which indicates if the `DateTime` has a timezone.
	`static DateTime Now`	Gets a `DateTime` object that is set to the current date and time on this computer.
	`short TimezoneOffset`	Gets or sets the timezone offset, in minutes, of the `DateTime` object.
	`System.DateTime Value`	Gets or sets the value of the `DateTime` object as a `System.DateTime` value.

Methods

	Name	Description
	`int CompareTo(object obj)`	The `DateTime` class implements the `IComparable` interface. This method compares the current instance of `DateTime` to another object and returns an integer that indicates whether the current instance precedes, follows, or occurs in the same position in the sort order as the other object. See also https://msdn.microsoft.com/en-us/library/system.icomparable.compareto(v=vs.110).aspx
	`override bool Equals(object obj)`	Returns **true** if the specified object is equal to the current object; **false** otherwise.
	`System.DateTime GetDateTime(bool correctTZ)`	Returns a `System.DateTime` object from the current `Altova.Types.DateTime` instance. The `correctTZ` Boolean argument specifies whether the time of the returned object must be adjusted according to the timezone of the current `Altova.Types.DateTime` instance.
	`override int GetHashCode()`	Returns the hash code of the current instance.
	`int GetWeekOfMonth()`	Returns the number of the week in month as an integer.

	`static DateTime Parse( string s )`	Creates a `DateTime` object from the string supplied as argument. For example, the following sample string values would be converted successfully to a `DateTime` object:
		
		`2015-01-01T23:23:23`
		`2015-01-01`
		`2015-11`
		`23:23:23`
		
		An exception is raised if the string cannot be converted to a `DateTime` object.
		
		Note that this method is static and can only be called on the `Altova.Types.DateTime` class itself, not on an instance of the class.

	`static DateTime Parse(string s, DateTimeFormat format)`	Creates a `DateTime` object from a string, using the format supplied as argument. For the list of possible formats, see Altova.Types.DateTimeFormat 1080. An exception is raised if the string cannot be converted to a `DateTime` object. Note that this method is static and can only be called on the `Altova.Types.DateTime` class itself, not on an instance of the class.
	`override string ToString()`	Converts the `DateTime` object to a string.

	Name	Description
	`string ToString(DateTimeFormat format)`	Converts the `DateTime` object to a string, using the format supplied as argument. For the list of possible formats, see Altova.Types.DateTimeFormat 1080.

Operators

Name	Description
`!=`	Determines if `DateTime` a is not equal to `DateTime` b.
`<`	Determines if `DateTime` a is less than `DateTime` b.
`<=`	Determines if `DateTime` a is less than or equal to `DateTime` b.
`==`	Determines if `DateTime` a is equal to `DateTime` b.
`>`	Determines if `DateTime` a is greater than `DateTime` b.
`>=`	Determines if `DateTime` a is greater than or equal to `DateTime` b.

Examples

Before using the following code listings in your program, ensure the Altova types are imported:

```
using Altova.Types;
```

The following code listing illustrates various ways to create `DateTime` objects:

```
protected static void DateTimeExample1()
{
   // Create a DateTime object from the current system time
   Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);
   Console.WriteLine("The current time is: " + dt.ToString());

   // Create an Altova DateTime object from parts (no timezone)
   Altova.Types.DateTime dt1 = new Altova.Types.DateTime(2015, 10, 12, 10, 50, 33);
   Console.WriteLine("My custom time is : " + dt1.ToString());

   // Create an Altova DateTime object from parts (with UTC+60 minutes timezone)
   Altova.Types.DateTime dt2 = new Altova.Types.DateTime(2015, 10, 12, 10, 50, 33, 60);
   Console.WriteLine("My custom time with timezone is : " + dt2.ToString());

   // Create an Altova DateTime object by parsing a string
   Altova.Types.DateTime dt3 = Altova.Types.DateTime.Parse("2015-01-01T23:23:23");
   Console.WriteLine("Time created from string: " + dt3.ToString());

   // Create an Altova DateTime object by parsing a string formatted as schema date
   Altova.Types.DateTime dt4 = Altova.Types.DateTime.Parse("2015-01-01",
DateTimeFormat.W3_date);
   Console.WriteLine("Time created from string formatted as schema date: " +
dt4.ToString());
}
```

The following code listing illustrates various ways to format `DateTime` objects:

```
protected static void DateTimeExample2()
{
   // Create a DateTime object from the current system time
   Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);

   // Output the unformatted DateTime
   Console.WriteLine("Unformatted time: " + dt.ToString());

   // Output this DateTime formatted using various formats
   Console.WriteLine("S_DateTime:       " + dt.ToString(DateTimeFormat.S_DateTime));
   Console.WriteLine("S_Days:           " + dt.ToString(DateTimeFormat.S_Days));
   Console.WriteLine("S_Seconds:        " + dt.ToString(DateTimeFormat.S_Seconds));
   Console.WriteLine("W3_date:          " + dt.ToString(DateTimeFormat.W3_date));
   Console.WriteLine("W3_dateTime:      " + dt.ToString(DateTimeFormat.W3_dateTime));
   Console.WriteLine("W3_gDay:          " + dt.ToString(DateTimeFormat.W3_gDay));
   Console.WriteLine("W3_gMonth:        " + dt.ToString(DateTimeFormat.W3_gMonth));
   Console.WriteLine("W3_gMonthDay:     " + dt.ToString(DateTimeFormat.W3_gMonthDay));
   Console.WriteLine("W3_gYear:         " + dt.ToString(DateTimeFormat.W3_gYear));
   Console.WriteLine("W3_gYearMonth:    " + dt.ToString(DateTimeFormat.W3_gYearMonth));
   Console.WriteLine("W3_time:          " + dt.ToString(DateTimeFormat.W3_time));
}
```

27.4.2 Altova.Types.DateTimeFormat

The `DateTimeFormat` enum type has the following constant values:

Value	Description	Example
S_DateTime	Formats the value as standard dateTime, with a precision of a ten-millionth of a second, including timezone.	`2015-11-12 12:19:03.9019132+01:00`
S_Days	Formats the value as number of days elapsed since the UNIX epoch.	`735913.631897345108796296 2963`
S_Seconds	Formats the value as number of seconds elapsed since the UNIX epoch, with a precision of a ten-millionth of a second.	`63582937678.0769062`
W3_date	Formats the value as schema date.	`2015-11-12`
W3_dateTime	Formats the value as schema dateTime.	`2015-11-12T15:12:14.5194251`
W3_gDay	Formats the value as schema gDay.	`---12` (assuming that the date is 12th of the month)
W3_gMonth	Formats the value as schema gMonth.	`--11`

Value	Description	Example
		(assuming that the month is November)
W3_gMonthDay	Formats the value as schema gMonthDay.	`--11-12` (assuming that the date is 12th of November)
W3_gYear	Formats the value as schema gYear.	`2015` (assuming that the year is 2015)
W3_gYearMonth	Formats the value as schema gYearMonth.	`2015-11` (assuming that the year is 2015 and the month is November)
W3_time	Formats the value as schema time, with a precision of a ten-millionth of a second.	`15:19:07.5582719`

27.4.3 Altova.Types.Duration

This class enables you to process XML attributes or elements of type `xs:duration`.

Constructors

	Name	Description
	`Duration(Duration obj)`	Initializes a new instance of the `Duration` class to the `Duration` object supplied as argument.
	`Duration(System.TimeSpan newvalue)`	Initializes a new instance of the `Duration` class to the `System.TimeSpan` object supplied as argument.
	`Duration(long ticks)`	Initializes a new instance of the `Duration` class to the number of ticks supplied as argument.
	`Duration(int newyears, int newmonths, int days, int hours, int minutes, int seconds, double partseconds, bool bnegative)`	Initializes a new instance of the `Duration` class to a duration built from parts supplied as arguments.

Properties

	Name	Description
	`int Months`	Gets or sets the number of months of the current instance of `Duration`.
	`System.TimeSpan Value`	Gets or sets the value (as `System.TimeSpan`) of the current instance of `Duration`.
	`int Years`	Gets or sets the number of years of the current instance of `Duration`.

Methods

	Name	Description
	`override bool Equals(object other)`	Returns **true** if the specified object is equal to the current object; **false** otherwise.
	`override int GetHashCode()`	Returns the hash code of the current instance.
	`bool IsNegative()`	Returns **true** if the current instance of `Duration` represents a negative duration.
	`static Duration Parse( string s, ParseType pt )`	Returns an `Altova.Types.Duration` object parsed from the string supplied as argument, using the parse type supplied as argument. Valid parse type values:

(C) 2015-2021 Altova GmbH

	Name	Description
		DURATION Parse duration assuming that year, month, day, as well as time duration parts exist. **YEARMONTH** Parse duration assuming that only year and month parts exist. **DAYTIME** Parse duration assuming that only the day and time parts exist. Note that this method is static and can only be called on the class itself, not on an instance of the class.
	`override string ToString()`	Converts the current `Duration` instance to string. For example, a time span of 3 hours, 4 minutes, and 5 seconds would be converted to "PT3H4M5S".
	`string ToYearMonthString()`	Converts the current `Duration` instance to string, using the "Year and Month" parse type.

Operators

Name	Description
`!=`	Determines if `Duration` a is not equal to `Duration` b.
`==`	Determines if `Duration` a is equal to `Duration` b.

Examples

Before using the following code listings in your program, ensure the Altova types are imported:

```
using Altova.Types;
```

The following code listing illustrates various ways to create `Duration` objects:

```
protected static void DurationExample1()
{
   // Create a new time span of 3 hours, 4 minutes, and 5 seconds
   System.TimeSpan ts = new TimeSpan(3, 4, 5);
   // Create a Duration from the time span
   Duration dr = new Duration(ts);
   // The output is: PT3H4M5S
   Console.WriteLine("Duration created from TimeSpan: " + dr.ToString());

   // Create a negative Altova.Types.Duration from 6 years, 5 months, 4 days, 3 hours,
       // 2 minutes, 1 second, and .33 of a second
   Duration dr1 = new Duration(6, 5, 4, 3, 2, 1, .33, true);
   // The output is: -P6Y5M4DT3H2M1.33S
   Console.WriteLine("Duration created from parts: " + dr1.ToString());

   // Create a Duration from a string using the DAYTIME parse type
```

```
   Duration dr2 = Altova.Types.Duration.Parse("-P4DT3H2M1S",
Duration.ParseType.DAYTIME);
   // The output is -P4DT3H2M1S
   Console.WriteLine("Duration created from string: " + dr2.ToString());

   // Create a duration from ticks
   Duration dr3 = new Duration(System.DateTime.UtcNow.Ticks);
   // Output the result
   Console.WriteLine("Duration created from ticks: " + dr3.ToString());
}
```

The following code listing illustrates getting values from `Duration` objects:

```
protected static void DurationExample2()
{
   // Create a negative Altova.Types.Duration from 6 years, 5 months, 4 days, 3 hours,
       // 2 minutes, 1 second, and .33 of a second
   Duration dr = new Duration(6, 5, 4, 3, 2, 1, .33, true);
   // The output is: -P6Y5M4DT3H2M1.33S
   Console.WriteLine("The complete duration is: " + dr.ToString());

   // Get only the year and month part as string
   string dr1 = dr.ToYearMonthString();
   Console.WriteLine("The YEARMONTH part is: " + dr1);

   // Get the number of years in duration
   Console.WriteLine("Years: " + dr.Years);

   // Get the number of months in duration
   Console.WriteLine("Months: " + dr.Months);
}
```

27.4.4 Altova.Xml.Meta.Attribute

This class enables you to access schema information about classes generated from attributes. Note that this class is not meant to provide dynamic information about particular instances of an attribute in an XML document. Instead, it enables you to obtain programmatically information about a particular attribute defined in the XML schema.

Properties

	Name	Description
	`SimpleType DataType`	Returns the type of the attribute content.
	`string LocalName`	Returns the local name of the attribute.
	`string NamespaceURI`	Returns the namespace URI of the attribute.
	`XmlQualifiedName QualifiedName`	Returns the qualified name of the attribute.
	`bool Required()`	Returns true if the attribute is required.

27.4.5 Altova.Xml.Meta.ComplexType

This class enables you to access schema information about classes generated from complex types. Note that this class is not meant to provide dynamic information about particular instances of a complex type in an XML document. Instead, it enables you to obtain programmatically information about a particular complex type defined in the XML schema.

Properties

	Name	Description
	`Attribute[] Attributes`	Returns a list of all attributes.
	`ComplexType BaseType`	Returns the base type of this type or null if no base type exists.
	`SimpleType ContentType`	Returns the simple type of the content.
	`Element[] Elements`	Returns a list of all elements.
	`string LocalName`	Returns the local name of the type.
	`string NamespaceURI`	Returns the namespace URI of the type.
	`XmlQualifiedName QualifiedName`	Returns the qualified name of this type.

Methods

	Name	Description
	`ComplexType BaseType`	Returns the base type of this type.
	`bool Equals(obj)`	Checks if two info objects refer to the same type, based on qualified name comparison. Returns true if the type has the same qualified name.
	`Attribute FindAttribute(string localName, string namespaceURI)`	Finds the attribute with the specified local name and namespace URI.
	`Element FindElement(string localName, string namespaceURI)`	Finds the element with the specified local name and namespace URI.

27.4.6 Altova.Xml.Meta.Element

This class enables you to access information about classes generated from schema elements. Note that this class is not meant to provide dynamic information about particular instances of an element in an XML document. Instead, it enables you to obtain programmatically information about a particular element defined in the XML schema.

Properties

	Name	Description
	`ComplexType DataType`	Returns the type of the element. Note that this is always a complex type even if declared as simple in the original schema. Use the `ContentType` property of the returned object to get the simple content type.
	`string` `LocalName`	Returns the local name of the element.
	`int` `MaxOccurs`	Returns the `maxOccurs` value defined in the schema.
	`int` `MinOccurs`	Returns the `minOccurs` value defined in the schema.
	`string` `NamespaceURI`	Returns the namespace URI of the element.
	`XmlQualifiedName QualifiedName`	Returns the qualified name of the element.

27.4.7 Altova.Xml.Meta.SimpleType

This class enables you to access schema information about classes generated from simple types. Note that this class is not meant to provide dynamic information about particular instances of simple types in an XML document. Instead, it enables you to obtain programmatically information about a particular simple type defined in the XML schema.

Properties

	Name	Description
	`SimpleType BaseType`	Returns the base type of this type.
	`string``[] Enumerations`	Returns a list of all enumeration facets.
	`int` `FractionDigits`	Returns the value of this facet.
	`int` `Length`	Returns the value of this facet.
	`string` `LocalName`	Returns the local name of the type.
	`string` `MaxExclusive`	Returns the value of this facet.
	`string` `MaxInclusive`	Returns the value of this facet.
	`int` `MaxLength`	Returns the value of this facet.
	`string` `MinExclusive`	Returns the value of this facet.
	`string` `MinInclusive`	Returns the value of this facet.
	`int` `MinLength`	Returns the value of this facet.

(C) 2015-2021 Altova GmbH

	Name	Description
	`string NamespaceURI`	Returns the namespace URI of the type.
	`string[] Patterns`	Returns the pattern facets, or null if no patterns are specified.
	`XmlQualifiedName QualifiedName`	Returns the qualified name of this type.
	`int TotalDigits`	Returns the value of this facet.
	`WhitespaceType Whitespace`	Returns the whitespace normalization facet.

27.4.8 [YourSchema].[Doc]

When code is generated from an XML Schema, the generated code provides a document class with the same name as the schema. This class contains all possible root elements as members, as well as the members listed below. Note that, in the method names below, "Doc" stands for the name of the generated document class itself.

Methods

	Name	Description
	`static Doc CreateDocument()`	Creates a new, empty XML document.
	`static Doc CreateDocument(string encoding)`	Creates a new, empty XML document, with encoding of type "encoding".
	`static void DeclareAllNamespacesFromSchema(Altova.Xml.ElementType node)`	Declares all namespaces from the XML Schema on the element supplied as argument (typically, the XML root element). Calling this method is useful if your schema has multiple namespace declarations, each mapped to a prefix, and you would like to declare all of them on the element supplied as argument.
	`static Doc LoadFromBinary(byte[] binary)`	Loads an XML document from a byte array.
	`static Doc LoadFromFile(string filename)`	Loads an XML document from a file.
	`static Doc LoadFromString(string xmlstring)`	Loads an XML document from a string.
	`byte[] SaveToBinary(bool prettyPrint)`	Saves an XML document to a byte array, with optional "pretty-print" formatting.
	`byte[] SaveToBinary(bool prettyPrint, string encoding)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding.

	Name	Description
	`byte[] SaveToBinary(bool prettyPrint, string encoding, bool bBigEndian, bool bBOM)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding, byte order, and BOM (Byte Order Mark).
	`void SaveToFile(string fileName, bool prettyPrint)`	Saves an XML document to a file, with optional "pretty-print" formatting.
	`void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl)`	Saves an XML document to a file, with optional "pretty-print" formatting. When `omitXmlDecl` is true, the XML declaration will not be written.
	`void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl, string encoding)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding. When `omitXmlDecl` is true, the XML declaration will not be written.
	`void SaveToFile(string fileName, bool prettyPrint, string encoding, string lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding, and line ending character(s).
	`void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl, string encoding, string lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding, and line ending character(s). When `omitXmlDecl` is true, the XML declaration will not be written.
	`void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl, string encoding, bool bBigEndian, bool bBOM, string lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding, byte order, BOM (Byte Order Mark), and line ending character(s). When `omitXmlDecl` is true, the XML declaration will not be written.
	`void SaveToFileWithLineEnd(string fileName, bool prettyPrint, bool omitXmlDecl, string lineend)`	Saves an XML document to a file, with optional "pretty-print" formatting, and line ending character(s). When `omitXmlDecl` is true, the XML declaration will not be written.
	`string SaveToString(bool prettyPrint)`	Saves an XML document to a file, with optional "pretty-print" formatting.
	`string SaveToString(bool prettyPrint, bool omitXmlDecl)`	Saves an XML document to a file, with optional "pretty-print" formatting. When `omitXmlDecl` is true, the XML declaration will not be written.
	`void SetDTDLocation(string dtdLocation)`	Adds a DOCTYPE declaration with the specified system ID. A root element must already exist.
	`void SetSchemaLocation(string schemaLocation)`	Adds an xsi:schemaLocation or xsi:noNamespaceSchemaLocation attribute to the root element. A root element must already exist.

(C) 2015-2021 Altova GmbH

27.4.9 [YourSchema].[ElementType]

This class provides methods for manipulating XML elements from your schema. Methods of this class can be called on elements, not on the XML document itself. Note that, in order to call methods of this class, you don't need to instantiate the class directly. Any element created using the **`Append()`** or **`AppendWithPrefix()`** methods is of `[ElementType]` type.

Methods

	Name	Description

	`void DeclareNamespace(string prefix, string nsURI)`	This method takes two arguments that are both of string type: the prefix and the namespace URI that you want to use. The prefix supplied as argument will be mapped to the namespace URI value supplied as argument. If the prefix supplied as argument is empty, the method creates or overrides the default namespace declaration in the element.
		
		For example, let's assume that the XML document has an XML element called "purchase". If you call
		
		`purchase.DeclareNamespace("ord", "http://OrderTypes");`
		
		then the XML document becomes
		
		`<purchase xmlns:ord="http://OrderTypes" />`
		
		Another example, if you call:
		
		`purchase.DeclareNamespace("", "http://OrderTypes");`
		
		then the XML document becomes
		
		`<purchase xmlns="http://OrderTypes" />`
		
		Note: The declared namespace is used when appending subsequent child elements or attributes, according to the following rules:
		
		1. If the child namespace is the default, then use empty prefix.
		2. If the child namespace is equal to the parent one, then use the parent prefix.
		3. Otherwise, search for nearest prefix from parent to top, using the lookup algorithm described in section "B.2: Namespace Prefix Lookup" at https://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20021022/namespaces-algorithms.html.

	Name	Description
		4. If there is no prefix for element namespace found, then use empty prefix.

27.4.10 [YourSchemaType].MemberAttribute

When code is generated from an XML schema, a class is created for each member attribute of a type. In the descriptions below, "AttributeType" stands for the type of the member attribute itself.

Methods

	Name	Description
	`bool Exists()`	Returns true if the attribute exists.
	`void Remove()`	Removes the attribute from its parent element.

Properties

	Name	Description
	`int` `EnumerationValue`	Generated for enumeration types only. Sets or gets the attribute value using one of the constants generated for the possible values. Returns **Invalid** if the value does not match any of the enumerated values in the schema.
	`Altova.Xml.Meta.Attribute Info`	Returns an object for querying schema information (see `Altova.Xml.Meta.Attribute` 1084).
	`AttributeType Value`	Sets or gets the attribute value.

27.4.11 [YourSchemaType].MemberElement

When code is generated from an XML schema, a class with the following members is created for each member element of a type. The class implements the standard `System.Collections.IEnumerable` interface, so it can be used with the `foreach` statement.

In the descriptions below, "MemberType" stands for the type of the member element itself.

Methods

	Name	Description
	`MemberType Append()`	Creates a new element and appends it to its parent.
	`MemberType AppendWithPrefix(string prefix)`	Creates a new element having the prefix supplied as argument, and appends it to its parent. For an example, see Example: Purchase Order 1055.

	Name	Description
	`MemberType At(int index)`	Returns the member element specified by the index.
	`System.Collections.IEnumerator GetEnumerator()`	Returns an object for iterating instances of the member element.
	`void Remove()`	Deletes all occurrences of the element from its parent.
	`void RemoveAt(int index)`	Deletes the occurrence of the element specified by the index.

Properties

	Name	Description
	`int Count`	Returns the count of elements.
	`int EnumerationValue`	Generated for enumeration types only. Sets or gets the element value using one of the constants generated for the possible values. Returns **Invalid** if the value does not match any of the enumerated values in the schema.
	`bool Exists`	Returns true if at least one element exists.
	`MemberType First`	Returns the first instance of the member element.
	`Altova.Xml.Meta.Element Info`	Returns an object for querying schema information (see `Altova.Xml.Meta.Element` 1085).
	`MemberType Last`	Returns the last instance of the member element.
	`MemberType this[int index]`	Returns the member element specified by the index.
	`MemberType Value`	Sets or gets the element content (only generated if element can have mixed or simple content).

27.5 Reference to Generated Classes (Java)

This chapter includes a description of Java classes generated with XMLSpy from a DTD or XML schema (see Generating Code from XML Schemas or DTDs 1021). You can integrate these classes into your code to read, modify, and write XML documents.

Note: The generated code does include other supporting classes, which are not listed here and are subject to modification.

27.5.1 com.altova.types.DateTime

This class enables you to process XML attributes or elements that have date and time types, such as `xs:dateTime`.

Constructors

	Name	Description
C	`public DateTime()`	Initializes a new instance of the `DateTime` class to an empty value.
C	`public DateTime(DateTime newvalue)`	Initializes a new instance of the `DateTime` class to the `DateTime` value supplied as argument.
C	`public DateTime(int newyear, int newmonth, int newday, int newhour, int newminute, int newsecond, double newpartsecond, int newoffsetTZ)`	Initializes a new instance of the `DateTime` class to the year, month, day, hour, minute, second, the fractional part of the second, and timezone supplied as arguments. The fractional part of the second newpartsecond must be between 0 and 1. The timezone offset newoffsetTZ can be either positive or negative and is expressed in minutes.
C	`public DateTime(int newyear, int newmonth, int newday, int newhour, int newminute, int newsecond, double newpartsecond)`	Initializes a new instance of the `DateTime` class to the year, month, day, hour, minute, second, and the fractional part of a second supplied as arguments.
C	`public DateTime(int newyear, int newmonth, int newday)`	Initializes a new instance of the `DateTime` class to the year, month, and day supplied as arguments.
C	`public DateTime(Calendar newvalue)`	Initializes a new instance of the `DateTime` class to the `java.util.Calendar` value supplied as argument.

Methods

	Name	Description
S	`static DateTime now()`	Returns the current time as a `DateTime` object.
S	`static DateTime parse(String s)`	Returns a `DateTime` object parsed from the string value supplied as argument. For example, the following sample string values

(C) 2015-2021 Altova GmbH

	Name	Description

		would be converted successfully to a `DateTime` object:
		
		`2015-11-24T12:54:47.969+01:00`
		`2015-11-24T12:54:47`
		`2015-11-24`

	int `getDay()`	Returns the day of the current `DateTime` instance.
	int `getHour()`	Returns the hour of the current `DateTime` instance.
	int `getMillisecond()`	Returns the millisecond of the current `DateTime` instance, as an integer value.
	int `getMinute()`	Returns the minute of the current `DateTime` instance.
	int `getMonth()`	Returns the month of the current `DateTime` instance.
	double `getPartSecond()`	Returns the fractional part of the second of the current `DateTime` instance, as a **double** value. The return value is greater than zero and smaller than one, for example: `0.313`
	int `getSecond()`	Returns the second of the current `DateTime` instance.
	int `getTimezoneOffset()`	Returns the timezone offset, in minutes, of the current `DateTime` instance. For example, the timezone "UTC-01:00" would be returned as: `-60`
	`Calendar getValue()`	Returns the current `DateTime` instance as a `java.util.Calendar` value.
	int `getWeekday()`	Returns the day in week of the current `DateTime` instance. Values range from 0 through 6, where 0 is Monday (ISO-8601).
	int `getYear()`	Returns the year of the current `DateTime` instance.
	int `hasTimezone()`	Returns information about the timezone of the current `DateTime` instance. Possible return values are: `CalendarBase.TZ_MISSING` — A timezone offset is not defined. `CalendarBase.TZ_UTC` — The timezone is UTC. `CalendarBase.TZ_OFFSET` — A timezone offset has been defined.
	void `setDay( int nDay )`	Sets the day of the current `DateTime` instance to the value supplied as argument.
	void `setHasTimezone( int nHasTZ )`	Sets the timezone information of the current `DateTime` instance to the value supplied as argument. This method can be used to

	Name	Description
		strip the timezone information or set the timezone to UTC (Coordinated Universal Time). Valid values for the nHasTZ argument: `CalendarBase.TZ_MISSING` Set the timezone offset to undefined. `CalendarBase.TZ_UTC` Set the timezone to UTC. `CalendarBase.TZ_OFFSET` If the current object has a timezone offset, leave it unchanged.
	`void setHour( int nHour )`	Sets the hour of the current `DateTime` instance to the value supplied as argument.
	`void setMinute( int nMinute )`	Sets the minute of the current `DateTime` instance to the value supplied as argument.
	`void setMonth( int nMonth )`	Sets the month of the current `DateTime` instance to the value supplied as argument.
	`void setPartSecond( double nPartSecond )`	Sets the fractional part of the second of the current `DateTime` instance to the value supplied as argument.
	`void setSecond( int nSecond )`	Sets the second of the current `DateTime` instance to the value supplied as argument.
	`void setTimezoneOffset( int nOffsetTZ )`	Sets the timezone offset of the current `DateTime` instance to the value supplied as argument. The value nOffsetTZ must be an integer (positive or negative) and must be expressed in minutes.
	`void setYear( int nYear )`	Sets the year of the current `DateTime` instance to the value supplied as argument.
	`String toString()`	Returns the string representation of the current `DateTime` instance, for example: `2015-11-24T15:50:56.968+01:00`

Examples

Before using the following code listings in your program, ensure the Altova types are imported:

```
import com.altova.types.*;
```

The following code listing illustrates various ways to create `DateTime` objects:

```
protected static void DateTimeExample1()
{
   // Initialize a new instance of the DateTime class to the current time
   DateTime dt = new DateTime(DateTime.now());
```

```
   System.out.println("DateTime created from current date and time: " + dt.toString());

   // Initialize a new instance of the DateTime class by supplying the parts
   DateTime dt1 = new DateTime(2015, 11, 23, 14, 30, 24, .459);
   System.out.println("DateTime from parts (no timezone): " + dt1.toString());

   // Initialize a new instance of the DateTime class by supplying the parts
   DateTime dt2 = new DateTime(2015, 11, 24, 14, 30, 24, .459, -60);
   System.out.println("DateTime from parts (with negative timezone): " +
dt2.toString());

   // Initialize a new instance of the DateTime class by parsing a string value
 DateTime dt3 = DateTime.parse("2015-11-24T12:54:47.969+01:00");
   System.out.println("DateTime parsed from string: " + dt3.toString());
}
```

The following code listing illustrates getting values from `DateTime` objects:

```
protected static void DateTimeExample2()
   {
      // Initialize a new instance of the DateTime class to the current time
      DateTime dt = new DateTime(DateTime.now());

      // Output the formatted year, month, and day of this DateTime instance
      String str1 = String.format("Year: %d; Month: %d; Day: %d;", dt.getYear(),
dt.getMonth(), dt.getDay());
      System.out.println(str1);

      // Output the formatted hour, minute, and second of this DateTime instance
      String str2 = String.format("Hour: %d; Minute: %d; Second: %d;", dt.getHour(),
dt.getMinute(), dt.getSecond());
      System.out.println(str2);

      // Return the timezone (in minutes) of this DateTime instance
      System.out.println("Timezone: " + dt.getTimezoneOffset());

      // Get the DateTime as a java.util.Calendar value
      java.util.Calendar dt_java = dt.getValue();
      System.out.println("" + dt_java.toString());

      // Return the day of week of this DateTime instance
      System.out.println("Weekday: " + dt.getWeekday());

      // Check whether the DateTime instance has a timezone defined
      switch(dt.hasTimezone())
      {
         case CalendarBase.TZ_MISSING:
            System.out.println("No timezone.");
            break;
         case CalendarBase.TZ_UTC:
            System.out.println("The timezone is UTC.");
            break;
         case CalendarBase.TZ_OFFSET:
            System.out.println("This object has a timezone.");
            break;
         default:
            System.out.println("Unable to determine whether a timezone is defined.");
            break;
```

```
      }
   }
```

The following code listing illustrates changing the timezone offset of a `DateTime` object:

```
protected static void DateTimeExample3()
{
   // Create a new DateTime object with timezone -0100 UTC
   DateTime dt = new DateTime(2015, 11, 24, 14, 30, 24, .459, -60);
   // Output the value before the change
   System.out.println("Before: " + dt.toString());
   // Change the offset to +0100 UTC
   dt.setTimezoneOffset(60);
   // Output the value after the change
   System.out.println("After:  " + dt.toString());
}
```

27.5.2 com.altova.types.Duration

This class enables you to process XML attributes or elements of type `xs:duration`.

Constructors

	Name	Description
C	`Duration(Duration newvalue)`	Initializes a new instance of the `Duration` class to the `Duration` object supplied as argument.
C	`Duration(int newyear, int newmonth, int newday, int newhour, int newminute, int newsecond, double newpartsecond, boolean newisnegative)`	Initializes a new instance of the `Duration` class to a duration built from parts supplied as arguments.

Methods

	Name	Description
S	`static Duration getFromDayTime( int newday, int newhour, int newminute, int newsecond, double newpartsecond )`	Returns a `Duration` object created from the number of days, hours, minutes, seconds, and fractional second parts supplied as argument.
S	`static Duration getFromYearMonth( int newyear, int newmonth )`	Returns a `Duration` object created from the number of years and months supplied as argument.
S	`static Duration parse( String s )`	Returns a `Duration` object created from the string supplied as argument. For example, the string -P1Y1M1DT1H1M1.333S can be used to create a negative duration of one year, one month,

	Name	Description
		one day, one hour, one minute, one second, and 0.333 fractional parts of a second. To create a negative duration, append the minus sign (-) to the string.
S	**static** Duration parse(String s, ParseType pt)	Returns a `Duration` object created from the string supplied as argument, using a specific parse format. The parse format can be any of the following: **ParseType.DAYTIME** — May be used when the string **s** consists of any of the following: days, hours, minutes, seconds, fractional second parts, for example -P4DT4H4M4.774S. **ParseType.DURATION** — May be used when the string **s** consists of any of the following: years, months, days, hours, minutes, seconds, fractional second parts, for example P1Y1M1DT1H1M1.333S. **ParseType.YEARMONTH** — May be used when the string **s** consists of any of the following: years, months. For example: P3Y2M.
	int getDay()	Returns the number of days in the current `Duration` instance.
	long getDayTimeValue()	Returns the day and time value (in milliseconds) of the current `Duration` instance. Years and months are ignored.
	int getHour()	Returns the number of hours in the current `Duration` instance.
	int getMillisecond()	Returns the number of milliseconds in the current `Duration` instance.
	int getMinute()	Returns the number of minutes in the current `Duration` instance.
	int getMonth()	Returns the number of months in the current `Duration` instance.
	double getPartSecond()	Returns the number of fractional second parts in the current `Duration` instance.
	int getSecond()	Returns the number of seconds in the current `Duration` instance.
	int getYear()	Returns the number of years in the current `Duration` instance.
	int getYearMonthValue()	Returns the year and month value (in months) of the current `Duration` instance. Days, hours, seconds, and milliseconds are ignored.

	Name	Description
	`boolean isNegative()`	Returns Boolean **true** if the current `Duration` instance is negative.
	`void setDayTimeValue(long l)`	Sets the duration to the number of milliseconds supplied as argument, affecting only the day and time part of the duration.
	`void setNegative( boolean isnegative )`	Converts the current `Duration` instance to a negative duration.
	`void setYearMonthValue(int l)`	Sets the duration to the number of months supplied as argument. Only the years and months part of the duration is affected.
	`String toString()`	Returns the string representation of the current `Duration` instance, for example: `-P4DT4H4M4.774S`
	`String toYearMonthString()`	Returns the string representation of the YearMonth part of the current `Duration` instance, for example: `P1Y2M`

Examples

Before using the following code listings in your program, ensure the Altova types are imported:

```
import com.altova.types.*;
import com.altova.types.Duration.ParseType;
```

The following code listing illustrates various ways to create `Duration` objects:

```
protected static void ExampleDuration()
{
   // Create a negative duration of 1 year, 1 month, 1 day, 1 hour, 1 minute, 1
second,
   // and 0.333 fractional second parts
   Duration dr = new Duration(1, 1, 1, 1, 1, 1, .333, true);

   // Create a duration from an existing Duration object
   Duration dr1 = new Duration(dr);

   // Create a duration of 4 days, 4 hours, 4 minutes, 4 seconds, .774 fractional
second parts
   Duration dr2 = Duration.getFromDayTime(4, 4, 4, 4, .774);

   // Create a duration of 3 years and 2 months
   Duration dr3 = Duration.getFromYearMonth(3, 2);

   // Create a duration from a string
   Duration dr4 = Duration.parse("-P4DT4H4M4.774S");

   // Create a duration from a string, using specific parse formats
   Duration dr5 = Duration.parse("-P1Y1M1DT1H1M1.333S", ParseType.DURATION);
```


(C) 2015-2021 Altova GmbH

```
   Duration dr6 = Duration.parse("P3Y2M", ParseType.YEARMONTH);
   Duration dr7 = Duration.parse("-P4DT4H4M4.774S", ParseType.DAYTIME);
}
```

The following code listing illustrates getting and setting the value of `Duration` objects:

```
protected static void DurationExample2()
{
   // Create a duration of 1 year, 2 month, 3 days, 4 hours, 5 minutes, 6 seconds,
       // and 333 milliseconds
   Duration dr = new Duration(1, 2, 3, 4, 5, 6, .333, false);
   // Output the number of days in this duration
   System.out.println(dr.getDay());

   // Create a positive duration of one year and 333 milliseconds
   Duration dr1 = new Duration(1, 0, 0, 0, 0, 0, .333, false);
   // Output the day and time value in milliseconds
   System.out.println(dr1.getDayTimeValue());

   // Create a positive duration of 1 year, 1 month, 1 day, 1 hour, 1 minute, 1
second,
       // and 333 milliseconds
   Duration dr2 = new Duration(1, 1, 1, 1, 1, 1, .333, false);
   // Output the year and month value in months
   System.out.println(dr2.getYearMonthValue());

   // Create a positive duration of 1 year and 1 month
   Duration dr3 = new Duration(1, 1, 0, 0, 0, 0, 0, false);
   // Output the value
   System.out.println("The duration is now: " + dr3.toString());
   // Set the DayTime part of duration to 1000 milliseconds
   dr3.setDayTimeValue(1000);
   // Output the value
   System.out.println("The duration is now: " + dr3.toString());
   // Set the YearMonth part of duration to 1 month
   dr3.setYearMonthValue(1);
   // Output the value
   System.out.println("The duration is now: " + dr3.toString());
   // Output the year and month part of the duration
   System.out.println("The YearMonth part of the duration is: " +
dr3.toYearMonthString());
}
```

27.5.3 com.altova.xml.meta.Attribute

This class enables you to access schema information about classes generated from attributes. Note that this class is not meant to provide dynamic information about particular instances of an attribute in an XML document. Instead, it enables you to obtain programmatically information about a particular attribute defined in the XML schema.

Methods

	Name	Description
	`SimpleType getDataType()`	Returns the type of the attribute content.
	`String getLocalName()`	Returns the local name of the attribute.
	`String getNamespaceURI()`	Returns the namespace URI of the attribute.
	`boolean isRequired()`	Returns true if the attribute is required.

27.5.4 com.altova.xml.meta.ComplexType

This class enables you to access schema information about classes generated from complex types. Note that this class is not meant to provide dynamic information about particular instances of a complex type in an XML document. Instead, it enables you to obtain programmatically information about a particular complex type defined in the XML schema.

Methods

	Name	Description
	`Attribute findAttribute(String localName, String namespaceURI)`	Finds the attribute with the specified local name and namespace URI.
	`Element findElement(String localName, String namespaceURI)`	Finds the element with the specified local name and namespace URI.
	`Attribute[] GetAttributes()`	Returns a list of all attributes.
	`ComplexType getBaseType()`	Returns the base type of this type.
	`SimpleType getContentType()`	Returns the simple type of the content.
	`Element[] GetElements()`	Returns a list of all elements.
	`String getLocalName()`	Returns the local name of the type.
	`String getNamespaceURI()`	Returns the namespace URI of the type.

27.5.5 com.altova.xml.meta.Element

This class enables you to access information about classes generated from schema elements. Note that this class is not meant to provide dynamic information about particular instances of an element in an XML document. Instead, it enables you to obtain programmatically information about a particular element defined in the XML schema.

Methods

	Name	Description
	`ComplexType getDataType()`	Returns the type of the element. Note that this is always a complex type even if declared as simple in the original schema. Use `getContentType()` of the returned object to get the simple content type.
	`String getLocalName()`	Returns the local name of the element.
	`int` `getMaxOccurs()`	Returns the maxOccurs value defined in the schema.
	`int` `getMinOccurs()`	Returns the minOccurs value defined in the schema.
	`String getNamespaceURI()`	Returns the namespace URI of the element.

27.5.6 com.altova.xml.meta.SimpleType

This class enables you to access schema information about classes generated from simple types. Note that this class is not meant to provide dynamic information about particular instances of simple types in an XML document. Instead, it enables you to obtain programmatically information about a particular simple type defined in the XML schema.

Methods

	Name	Description
	`SimpleType getBaseType()`	Returns the base type of this type.
	`String[] getEnumerations()`	Returns an array of all enumeration facets.
	`int` `getFractionDigits()`	Returns the value of this facet.
	`int` `getLength()`	Returns the value of this facet.
	`String getLocalName()`	Returns the local name of the type.
	`String getMaxExclusive()`	Returns the value of this facet.
	`String getMaxInclusive()`	Returns the value of this facet.
	`int` `getMaxLength()`	Returns the value of this facet.
	`String getMinExclusive()`	Returns the value of this facet.
	`String getMinInclusive()`	Returns the value of this facet.
	`int` `getMinLength()`	Returns the value of this facet.
	`String getNamespaceURI()`	Returns the namespace URI of the type.
	`String[] getPatterns()`	Returns an array of all pattern facets.

	Name	Description
	`int getTotalDigits()`	Returns the value of this facet.
	`int getWhitespace()`	Returns the value of the whitespace facet, which is one of: `com.altova.typeinfo.WhitespaceType.Whitespace_Unknown` `com.altova.typeinfo.WhitespaceType.Whitespace_Preserve` `com.altova.typeinfo.WhitespaceType.Whitespace_Replace` `com.altova.typeinfo.WhitespaceType.Whitespace_Collapse`

27.5.7 com.[YourSchema].[Doc]

When code is generated from an XML Schema, the generated code provides a document class with the same name as the schema. This class contains all possible root elements as members, as well as the members listed below. Note that, in the method names below, "Doc" stands for the name of the generated document class itself.

Methods

	Name	Description
S	`static Doc createDocument()`	Creates a new, empty XML document.
S	`static void declareAllNamespacesFromSchema(com.altova.xml.ElementType node)`	Declares all namespaces from the XML Schema on the element supplied as argument (typically, the XML root element). Calling this method is useful if your schema has multiple namespace declarations, each mapped to a prefix, and you would like to declare all of them on the element supplied as argument.
S	`static Doc loadFromBinary(byte[] xml)`	Loads an XML document from a byte array.
S	`static Doc loadFromFile(String fileName)`	Loads an XML document from a file.
S	`static Doc loadFromString(String xml)`	Loads an XML document from a string.
	`byte[] saveToBinary(boolean prettyPrint)`	Saves an XML document to a byte array, with optional "pretty-print" formatting.
	`byte[] saveToBinary(boolean prettyPrint, String encoding)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding.
	`byte[] saveToBinary(boolean prettyPrint, String encoding, boolean bigEndian, boolean writeBOM)`	Saves an XML document to a byte array, with optional "pretty-print" formatting, with the specified encoding. Byte order and Unicode byte-order mark can be specified for Unicode encodings.

	Name	Description
	`void saveToFile(String fileName, boolean prettyPrint)`	Saves an XML document to a file, with optional "pretty-print" formatting.
	`void saveToFile(String fileName, boolean prettyPrint, boolean omitXmlDecl)`	Saves an XML document to a file, with optional "pretty-print" formatting, with UTF-8 encoding. When `omitXmlDecl` is true, the XML declaration will not be written.
	`void saveToFile(String fileName, boolean prettyPrint, boolean omitXmlDecl, String encoding)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding. When `omitXmlDecl` is true, the XML declaration will not be written.
	`void saveToFile(String fileName, boolean prettyPrint, boolean omitXmlDecl, String encoding, boolean bBigEndian, boolean bBOM)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding. When `omitXmlDecl` is true, the XML declaration will not be written. Byte order and Unicode byte-order mark can be specified for Unicode encodings.
	`void saveToFile(String fileName, boolean prettyPrint, String encoding)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding.
	`void saveToFile(String fileName, boolean prettyPrint, String encoding, boolean bBigEndian, boolean bBOM)`	Saves an XML document to a file, with optional "pretty-print" formatting, with the specified encoding. Byte order and Unicode byte-order mark can be specified for Unicode encodings.
	`String saveToString(boolean prettyPrint)`	Saves an XML document to a string, with optional "pretty-print" formatting.
	`String saveToString(boolean prettyPrint, boolean omitXmlDecl)`	Saves an XML document to a string, with optional "pretty-print" formatting. When `omitXmlDecl` is true, the XML declaration will not be written.
	`void setSchemaLocation(String schemaLocation)`	Adds an `xsi:schemaLocation` or `xsi:noNamespaceSchemaLocation` attribute to the root element. A root element must already exist.

27.5.8 com.[YourSchema].[ElementType]

This class provides methods for manipulating XML elements from your schema. Methods of this class can be called on elements, not on the XML document itself. Note that, in order to call methods of this class, you don't need to instantiate the class directly. Any element created using the **`append()`** or **`appendWithPrefix()`** methods is of [`ElementType`] type.

Methods

	Name	Description

	`void declareNamespace(String prefix, String nsURI)`	This method takes two arguments that are both of string type: the prefix and the namespace URI that you want to use. The prefix supplied as argument will be mapped to the namespace URI value supplied as argument. If the prefix supplied as argument is empty, the method creates or overrides the default namespace declaration in the element.
		
		For example, let's assume that the XML document has an XML element called "purchase". If you call
		
		`purchase.declareNamespace("ord", "http://OrderTypes");`
		
		then the XML document becomes
		
		`<purchase xmlns:ord="http://OrderTypes" />`
		
		Another example, if you call:
		
		`purchase.declareNamespace("", "http://OrderTypes");`
		
		then the XML document becomes
		
		`<purchase xmlns="http://OrderTypes" />`
		
		Note: The declared namespace is used when appending subsequent child elements or attributes, according to the following rules:
		
		1. If the child namespace is the default, then use empty prefix.
		2. If the child namespace is equal to the parent one, then use the parent prefix.
		3. Otherwise, search for nearest prefix from parent to top, using the lookup algorithm described in section "B.2: Namespace Prefix Lookup" at https://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20021022/namespaces-algorithms.html.
		4. If there is no prefix for element namespace found, then use empty prefix.

(C) 2015-2021 Altova GmbH

27.5.9 com.[YourSchema].[YourSchemaType].MemberAttribute

When code is generated from an XML schema, a class is created for each member attribute of a type. In the descriptions below, "AttributeType" stands for the type of the member attribute itself.

Methods

	Name	Description
	`boolean exists()`	Returns true if the attribute exists.
	`int getEnumerationValue()`	Generated for enumeration types only. Returns one of the constants generated for the possible values, or **Invalid** if the value does not match any of the enumerated values in the schema.
	`com.altova.xml.meta.Attribute getInfo()`	Returns an object for querying schema information (see `com.altova.xml.meta.Attribute` 1099).
	`AttributeType getValue()`	Gets the attribute value.
	`void remove()`	Removes the attribute from its parent element.
	`void setEnumerationValue(int)`	Generated for enumeration types only. Pass one of the constants generated for the possible values to this method to set the value.
	`void setValue(AttributeType value)`	Sets the attribute value.

27.5.10 com.[YourSchema].[YourSchemaType].MemberElement

When code is generated from an XML schema, a class with the following members is created for each member element of a type. In the descriptions below, "MemberType" stands for the type of the member element itself.

Methods

	Name	Description
	`MemberType append()`	Creates a new element and appends it to its parent.
	`MemberType appendWithPrefix(String prefix)`	Creates a new element having the prefix supplied as argument, and appends it to its parent. For an example, see Example: Purchase Order 1055 .
	`MemberType at(int index)`	Returns the instance of the member element at the specified index.
	`int count()`	Returns the count of elements.

	Name	Description
	`boolean exists()`	Returns true if at least one element exists.
	`MemberType first()`	Returns the first instance of the member element.
	`int getEnumerationValue()`	Generated for enumeration types only. Returns one of the constants generated for the possible values, or **Invalid** if the value does not match any of the enumerated values in the schema.
	`com.altova.xml.meta.Element getInfo()`	Returns an object for querying schema information (see `com.altova.xml.meta.Element` 1100).
	`MemberType getValue()`	Gets the element content (only generated if element can have simple or mixed content).
	`java.util.Iterator iterator()`	Returns an object for iterating instances of the member element.
	`MemberType last()`	Returns the last instance of the member element.
	`void remove()`	Deletes all occurrences of the element from its parent.
	`void removeAt(int index)`	Deletes the occurrence of the element specified by the index.
	`void setEnumerationValue(int index)`	Generated for enumeration types only. Pass one of the constants generated for the possible values to this method to set the value.
	`void setValue(MemberType value)`	Sets the element content (only generated if element can have simple or mixed content).

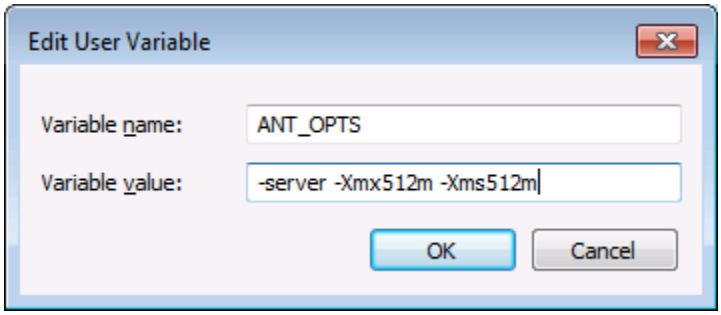
(C) 2015-2021 Altova GmbH

27.6 Code Generation Tips

Resolving "Out of memory" exceptions during Java compilation

Complex schemas can produce a large amount of code, which might cause a `java.lang.OutofMemory` exception during compilation using Ant. To rectify this:

- Add the environment variable `ANT_OPTS`, which sets specific Ant options such as the memory to be allocated to the compiler, and set its value as shown below.



- To make sure that the compiler and the generated code run in the same process as Ant, change the `fork` attribute, in **build.xml**, to `false`.

You may need to customize the values depending on the amount of memory in your machine and the size of the project you are working with. For more details, see your Java VM documentation.

When running the `ant jar` command, you may get an error message similar to "`[...] archive contains more than 65535 entities`". To prevent this, it is recommended that you use Ant 1.9 or later, and, in the **build.xml** file, add `zip64mode="as-needed"` to the `<jar>` element.

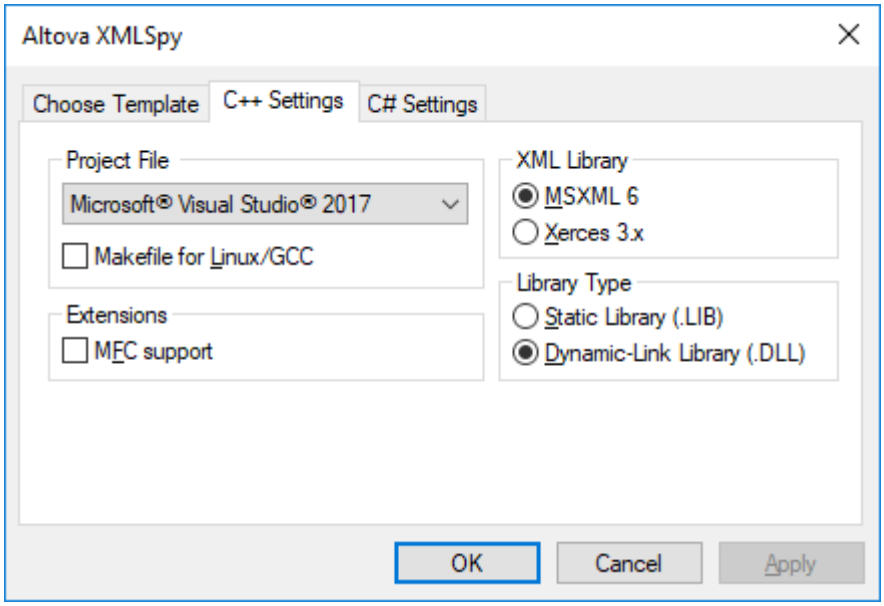
Reserving method names

When customizing code generation using the supplied SPL files, it might be necessary to reserve names to avoid collisions with other symbols. To do this:

1. Navigate to the program installation directory, for example, `C:\Program Files\Altova\XMLSpy2021`.
2. In the **spl** subdirectory, locate the directory corresponding to the programming language, for example, **..\spl\java**.
3. Open the **settings.spl** file and insert a new line into the "reserve" section, for example, **`reserve "myReservedWord"`**.
4. Regenerate the program code.

27.7 Code Generator Options

The menu option **DTD/Schema | Generate Program Code** lets you specify the target programming language as well as specific code generation settings.



The available settings are as follows.

C++ Settings	Defines the specific compiler settings for the C++ environment, namely: • The Visual Studio version (2010, 2013, 2015, 2017, 2019) • Whether a makefile for Linux with GCC compiler must be generated. • The XML library (MSXML, Xerces 3.x) • Whether static or dynamic libraries must be generated • Whether code must be generated with or without MFC support The **Makefile for Linux/GCC** option adds makefiles to the generated code. C++ source files are generated so that they are portable using `#ifdef` constructs to support different compilers and operating systems. Note the following if you intend to compile the generated code with GCC (GNU Compiler Collection) on Linux: • For Linux/GCC compilation, the only supported XML Library is **Xerces 3.x**. • Selecting the check box **MFC support** has no effect on compilation with Linux/GCC.
C# Settings	Select the option **Microsoft .NET Core 3.1** or **Microsoft .NET 5.0** to generate a Visual Studio solution targeting the respective platforms.

	If you need to target the **.NET Framework** platform for a specific Visual Studio version, select any of the **Microsoft Visual Studio 2010-2019** options—in this case, the generated solution will target the .NET Framework version corresponding to the respective Visual Studio version.

27.8 SPL Reference

This section gives an overview of SPL (Spy Programming Language), the code generator's template language.

It is assumed that you have prior programming experience, and are familiar with operators, functions, variables and classes, as well as the basics of object-oriented programming - which is used heavily in SPL. You should also have detailed knowledge of XML Schema.

The templates used by XMLSpy are supplied in the **...\XMLSpy\spl** folder. You can use these files as an aid to help you in developing your own templates.

How code generator works

Inputs to the code generator are the template files (.spl) and the object model provided by XMLSpy. The template files contain SPL instructions for creating files, reading information from the object model and performing calculations, interspersed with literal code fragments in the target programming language.

The template file is interpreted by the code generator and outputs **.cpp, .java, .cs** source code files, project files, or any other type of file depending on the template. The source code can then be compiled into an executable file that accesses XML data described by the schema file.

SPL files have access to a wide variety of information that is collated from the source schemas. Note that an SPL file is not tied to a specific schema, but allows access to all schemas. Make sure you write your SPL files generically and avoid structures which apply to specific schemas.

Example: Creating a new file in SPL

This is a very basic SPL file. It creates a file named **test.cpp,** and places the include statement within it. The close command completes the template.

```
[create "test.cpp"]
#include "stdafx.h"
[close]
```

27.8.1 Basic SPL structure

An SPL file contains literal text to output, interspersed with code generator instructions.

Code generator instructions are enclosed in square brackets '**[**' and '**]**'. Multiple statements can be included in a bracket pair. Additional statements have to be separated by a new line or a colon '**:**'.

Valid examples are:

```
[$x = 42
$x = $x + 1]
```

or

(C) 2015-2021 Altova GmbH

```
[$x = 42: $x = $x + 1]
```

Adding text to files

Text not enclosed by **[** and **]**, is written directly to the current output file. If there is no current output file, the text is ignored (see Using files[1115] how to create an output file).

To output literal square brackets, escape them with a backslash: **\[** and **\]**; to output a backslash use ****.

Comments

Comments inside an instruction block always begin with a **'** character, and terminate on the next line, or at a block close character **]**.

27.8.2 Declarations

The following statements are evaluated while parsing the SPL template file. They are **not** affected by flow control statements like conditions, loops or subroutines, and are always evaluated exactly once.

These keywords, like all keywords in SPL, are not case sensitive.

Remember that all of these declarations must be inside a block delimited by square brackets.

map ... to ...

```
map mapname key to value [, key to value ]...
```

This statement adds information to a map. See below for specific uses.

```
map schemanativetype schematype to typespec
```

The specified built-in XML Schema type will be mapped to the specified native type or class, using the specified formatter. This setting applies only to code generation for version 2007r3 and higher. Typespec is a native type or class name, followed by a comma, followed by the formatter class instance.

Example:

```
map schemanativetype "double" to "double,Altova::DoubleFormatter"
```

map type ... to ...

```
map type schematype to classname
```

The specified built-in XML Schema type will be mapped to the specified class. This setting applies only to code generation for version 2007 or lower.

Example:

```
map type "float" to "CSchemaFloat"
```

default ... is ...

```
default setting is value
```

This statement allows you to affect how class and member names are derived from the XML Schema. Note that the setting names are case sensitive.

Example:

```
default "InvalidCharReplacement" is "_"
```

Setting name	Explanation
ValidFirstCharSet	Allowed characters for starting an identifier
ValidCharSet	Allowed characters for other characters in an identifier
InvalidCharReplacement	The character that will replace all characters in names that are not in the ValidCharSet
AnonTypePrefix	Prefix for names of anonymous types*
AnonTypeSuffix	Suffix for names of anonymous types*
ClassNamePrefix	Prefix for generated class names
ClassNameSuffix	Suffix for generated class names
EnumerationPrefix	Prefix for symbolic constants declared for enumeration values
EnumerationUpperCase	"on" to convert the enumeration constant names to upper case
FallbackName	If a name consists only of characters that are not in ValidCharSet, use this one

* Names of anonymous types are built from AnonTypePrefix + element name + AnonTypeSuffix

reserve

```
reserve word
```

Adds the specified word to the list of reserved words. This ensures that it will never be generated as a class or member name.

(C) 2015-2021 Altova GmbH

Example:

```
reserve "while"
```

include

includes the specified file as SPL source. This allows you to split your template into multiple files for easier editing and handling.

```
include filename
```

Example:

```
include "Module.cpp"
```

27.8.3 Variables

Any non-trivial SPL file will require variables. Some variables are predefined[1114] by the code generator, and new variables may be created simply by assigning values to them.

The **$** character is used when **declaring** or **using** a variable, a variable name is always prefixed by **$**. Variable names are **case sensitive**.

Variables types:

- integer - also used as boolean, where 0 is false and everything else is true
- string
- object - provided by XMLSpy
- iterator - see foreach[1118] statement

Variable types are declared by first assignment:

```
[$x = 0]
```

x is now an integer.

```
[$x = "teststring"]
```

x is now treated as a string.

Strings

String constants are always enclosed in double quotes, like in the example above. **\n** and **\t** inside double quotes are interpreted as newline and tab, **\"** is a literal double quote, and \\ is a backslash. String constants can also span multiple lines.

String concatenation uses the **&** character:

```
[$BasePath = $outputpath & "/" & $JavaPackageDir]
```

Objects

Objects represent the information contained in the XML schema. Objects have **properties**, which can be accessed using the **.** operator. It is not possible to create new objects in SPL (they are predefined by the code generator, derived from the input schema), but it is possible to assign objects to variables.

Example:

```
class [=$class.Name]
```

This example outputs the word "class", followed by a space and the value of the **Name** property of the **$class** object.

27.8.4 Predefined variables

After a Schema file is analyzed by the code generator, the objects in the table below exist in the Template Engine.

Name	Type	Description
$schematype	integer	1 for DTD, 2 for XML Schema
$TheLibrary	Library 1122	The library derived from the XML Schema or DTD
$module	string	Name of the source Schema without extension
$outputpath	string	The output path specified by the user, or the default output path

For C++ generation only:

Name	Type	Description
$domtype	integer	1 for MSXML, 2 for Xerces
$libtype	integer	1 for static LIB, 2 for DLL
$mfc	boolean	True if MFC support is enabled
$VSVersion	integer	Specifies the Visual Studio version. Valid values: **0** No Visual Studio project **2010** Visual Studio 2010 **2013** Visual Studio 2013 **2015** Visual Studio 2015

(C) 2015-2021 Altova GmbH

Name	Type	Description
		2017 Visual Studio 2017 **2019** Visual Studio 2019

For C# generation only:

Name	Type	Description
$VSVersion	integer	Specifies the Visual Studio version. Valid values: **0** No Visual Studio project **2010** Visual Studio 2010 **2013** Visual Studio 2013 **2015** Visual Studio 2015 **2017** Visual Studio 2017 **2019** Visual Studio 2019

27.8.5 Creating output files

These statements are used to create output files from the code generation. Remember that all of these statements must be inside a block delimited by square brackets.

create

```
create filename
```

creates a new file. The file has to be closed with the **close** statement. All following output is written to the specified file.

Example:

```
[create $outputpath & "/" & $JavaPackageDir & "/" & $application.Name & ".java"]
package [=$JavaPackageName];

public class [=$application.Name]Application {
...
}
[close]
```

close

closes the current output file.

```
=$variable
```

writes the value of the specified variable to the current output file.

Example:

```
[$x = 20+3]
The result of your calculation is [=$x] - so have a nice day!
```

The file output will be:

```
The result of your calculation is 23 - so have a nice day!
```

write

```
write string
```

writes the string to the current output file.

Example:

```
[write "C" & $name]
```

This can also be written as:

```
C[=$name]
```

filecopy ... to ...

```
filecopy source to target
```

copies the source file to the target file, without any interpretation.

(C) 2015-2021 Altova GmbH

Example:

```
filecopy "java/mapforce/mapforce.png" to $outputpath & "/" & $JavaPackageDir &
"/mapforce.png"
```

27.8.6 Operators

Operators in SPL work like in most other programming languages.

List of SPL operators in descending precedence order:

.	Access object property
( )	Expression grouping
true	boolean constant "true"
false	boolean constant "false"
&	String concatenation
-	Sign for negative number
not	Logical negation
*	Multiply
/	Divide
%	Modulo
+	Add
-	Subtract
<=	Less than or equal
<	Less than
>=	Greater than or equal
>	Greater than
=	Equal
<>	Not equal
and	Logical conjunction (with short circuit evaluation)
or	Logical disjunction (with short circuit evaluation)
=	Assignment

27.8.7 Conditions

SPL allows you to use standard "if" statements. The syntax is as follows:

```
if condition
        statements
else
        statements
```

```
endif
```

or, without else:

```
if condition
	statements
endif
```

Note: There are no round brackets enclosing the condition.

As in any other programming language, conditions are constructed with logical and comparison operators[1117].

Example:

```
[if $namespace.ContainsPublicClasses and $namespace.Prefix <> ""]
	whatever you want ['inserts whatever you want, in the resulting file]
[endif]
```

Switch

SPL also contains a multiple choice statement.

Syntax:

```
switch $variable
	case X:
		statements
	case Y:
	case Z:
		statements
	default:
		statements
endswitch
```

The case labels must be constants or variables.

The switch statement in SPL does not fall through the cases (as in C), so there is no need for a "break" statement.

27.8.8 Collections and foreach

Collections and iterators

A collection contains multiple objects - like a ordinary array. Iterators solve the problem of storing and incrementing array indexes when accessing objects.

Syntax:

```
foreach iterator in collection
	statements
```

```
next
```

Example:

```
[foreach $class in $classes
       if not $class.IsInternal
              ]      class [=$class.Name];
[      endif
next]
```

Example 2:

```
[foreach $i in 1 To 3
      Write "// Step " & $i & "\n"
      ' Do some work
next]
```

In the first line:

$classes is the global object[1114] of all generated types. It is a collection of single class objects.

Foreach steps through all the items in $classes, and executes the code following the instruction, up to the **next** statement, for each of them.

In each iteration, **$class** is assigned to the next class object. You simply work with the class object instead of using, classes[i]->Name(), as you would in C++.

All collection iterators have the following additional properties:

Index	The current index, starting with 0
IsFirst	true if the current object is the first of the collection (index is 0)
IsLast	true if the current object is the last of the collection
Current	The current object (this is implicit if not specified and can be left out)

Example:

```
[foreach $enum in $facet.Enumeration
       if not $enum.IsFirst
              ], [
       endif
       ]"[=$enum.Value]"[
next]
```

27.8.9 Subroutines

Code generator supports subroutines in the form of procedures or functions.

Features:

- By-value and by-reference passing of values
- Local/global parameters (local within subroutines)
- Local variables
- Recursive invocation (subroutines may call themselves)

27.8.9.1 Subroutine declaration

Subroutines

Syntax example:

```
Sub SimpleSub()
        … lines of code
 EndSub
```

- **Sub** is the keyword that denotes the procedure.
- **SimpleSub** is the name assigned to the subroutine.
- Round **parenthesis** can contain a parameter list.
- The code block of a subroutine starts immediately after the closing parameter parenthesis.
- **EndSub** denotes the end of the code block.

Note: Recursive or cascaded subroutine **declaration** is not permitted, i.e. a subroutine may not contain another subroutine.

Parameters

Parameters can also be passed by procedures using the following syntax:

- All parameters must be variables
- Variables must be prefixed by the **$** character
- Local variables are defined in a subroutine
- Global variables are declared explicitly, outside of subroutines
- Multiple parameters are separated by the comma character "," within round parentheses
- Parameters can pass values

Parameters - passing values

Parameters can be passed in two ways, by value and by reference, using the keywords **ByVal** and **ByRef** respectively.

Syntax:

```
' define sub CompleteSub()
[Sub CompleteSub( $param, ByVal $paramByValue, ByRef $paramByRef )
] ...
```

- **ByVal** specifies that the parameter is passed by value. Note that most objects can only be passed by reference.
- **ByRef** specifies that the parameter is passed by reference. This is the default if neither ByVal nor ByRef is specified.

(C) 2015-2021 Altova GmbH

Function return values

To return a value from a subroutine, use the **return** statement. Such a function can be called from within an expression.

Example:

```
' define a function
[Sub MakeQualifiedName( ByVal $namespacePrefix, ByVal $localName )
if $namespacePrefix = ""
  return $localName
else
  return $namespacePrefix & ":" & $localName
endif
EndSub
]
```

27.8.9.2 Subroutine invocation

Use **call** to invoke a subroutine, followed by the procedure name and parameters, if any.

```
Call SimpleSub()
```

or

```
Call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
```

Function invocation

To invoke a function (any subroutine that contains a **return** statement), simply use its name inside an expression. Do not use the **call** statement to call functions. Example:

```
$QName = MakeQualifiedName($namespace, "entry")
```

27.8.9.3 Subroutine example

The following example shows subroutine declaration and invocation.

```
[create $outputpath & $module & "output.txt"

' define sub SimpleSub()
Sub SimpleSub()
]SimpleSub() called
[endsub

' execute sub SimpleSub()
Call SimpleSub()
```

```
$ParamByValue       = "Original Value"
]ParamByValue       = [=$ParamByValue]
[$ParamByRef  = "Original Value"
]ParamByRef   = [=$ParamByRef]

' define sub CompleteSub()
[Sub CompleteSub( $param, ByVal $paramByValue, ByRef $paramByRef )
]CompleteSub called.
      param = [=$param]
      paramByValue = [=$paramByValue]
      paramByRef = [=$paramByRef]
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]     Set values inside Sub
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]CompleteSub finished.
[endsub

' run sub CompleteSub()
Call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
]
ParamByValue=[=$ParamByValue]
ParamByRef=[=$ParamByRef]
[
Close
]
```

27.8.10 Built in Types

The section describes the properties of the built-in types used in the predefined variables[1114] which describe the parsed schema.

27.8.10.1 Library

This object represents the whole library generated from the XML Schema or DTD.

Property	Type	Description
SchemaNamespaces	Namespace[1123] collection	Namespaces in this library
SchemaFilename	string	Name of the XSD or DTD file this library is derived from
SchemaType	integer	1 for DTD, 2 for XML Schema
Guid	string	A globally unique ID
CodeName	string	Generated library name (derived from schema file name)

(C) 2015-2021 Altova GmbH

27.8.10.2 Namespace

One namespace object per XML Schema namespace is generated. Schema components that are not in any namespace are contained in a special namespace object with an empty NamespaceURI. Note that for DTD, namespaces are also derived from attributes whose names begin with "xmlns".

Property	Type	Description
CodeName	string	Name for generated code (derived from prefix)
LocalName	string	Namespace prefix
NamespaceURI	string	Namespace URI
Types	Type 1123 collection	All types contained in this namespace
Library	Library 1122	Library containing this namespace

27.8.10.3 Type

This object represents a complex or simple type. It is used to generate a class in the target language. There is one additional type per library that represents the document, which has all possible root elements as members.

Anonymous types have an empty LocalName.

Property	Type	Description
CodeName	string	Name for generated code (derived from local name or parent declaration)
LocalName	string	Original name in the schema
Namespace	Namespace 1123	Namespace containing this type
Attributes	Member 1124 collection	Attributes contained in this type*
Elements	Member 1124 collection	Child elements contained in this type
IsSimpleType	boolean	True for simple types, false for complex types
IsDerived	boolean	True if this type is derived from another type, which is also represented by a Type object
IsDerivedByExtension	boolean	True if this type is derived by extension
IsDerivedByRestriction	boolean	True if this type is derived by restriction
IsDerivedByUnion	boolean	True if this type is derived by union

Property	Type	Description
IsDerivedByList	boolean	True if this type is derived by list
BaseType	Type	The base type of this type (if IsDerived is true)
IsDocumentRootType	boolean	True if this type represents the document itself
Library	Library 1122	Library containing this type
IsFinal	boolean	True if declared as final in the schema
IsMixed	boolean	True if this type can have mixed content
IsAbstract	boolean	True if this type is declared as abstract
IsGlobal	boolean	True if this type is declared globally in the schema
IsAnonymous	boolean	True if this type is declared locally in an element

For simple types only:

Property	Type	Description
IsNativeBound	boolean	True if native type binding exists
NativeBinding	NativeBinding 1125	Native binding for this type
Facets	Facets 1125	Facets of this type
Whitespace	string	Shortcut to the Whitespace facet

* Complex types with text content (these are types with mixed content and complexType with simpleContent) have an additional unnamed attribute member that represents the text content.

27.8.10.4 Member

This object represents an attribute or element in the XML Schema. It is used to create class members of types.

Property	Type	Description
CodeName	string	Name for generated code (derived from local name or parent declaration)
LocalName	string	Original name in the schema. Empty for the special member representing text content of complex types.
NamespaceURI	string	The namespace URI of this Element/Attribute within XML instance documents/streams.

(C) 2015-2021 Altova GmbH

Property	Type	Description
DeclaringType	Type 1123	Type originally declaring the member (equal to ContainingType for non-inherited members)
ContainingType	Type 1123	Type where this is a member of
DataType	Type 1123	Data type of this member's content
Library	Library 1122	Library containing this member's DataType
IsAttribute	boolean	True for attributes, false for elements
IsOptional	boolean	True if minOccurs = 0 or optional attribute
IsRequired	boolean	True if minOccurs > 0 or required attribute
IsFixed	boolean	True for fixed attributes, value is in Default property
IsDefault	boolean	True for attributes with default value, value is in Default property
IsNillable	boolean	True for nillable elements
IsUseQualified	boolean	True if NamespaceURI is not empty
MinOccurs	integer	minOccurs, as in schema. 1 for required attributes
MaxOccurs	integer	maxOccurs, as in schema. 0 for prohibited attributes, -1 for unbounded
Default	string	Default value

27.8.10.5 NativeBinding

This object represents the binding of a simple type to a native type in the target programming language, as specified by the "schemanativetype" map.

Property	Type	Description
ValueType	string	Native type
ValueHandler	string	Formatter class instance

27.8.10.6 Facets

This object represents all facets of a simple type. Inherited facets are merged with the explicitly declared facets. If a Length facet is in effect, MinLength and MaxLength are set to the same value.

Property	Type	Description
DeclaringType	Type	Type facets are declared on
Whitespace	string	"preserve", "collapse" or "replace"
MinLength	integer	Facet value
MaxLength	integer	Facet value
MinInclusive	integer	Facet value
MinExclusive	integer	Facet value
MaxInclusive	integer	Facet value
MaxExclusive	integer	Facet value
TotalDigits	integer	Facet value
FractionDigits	integer	Facet value
List	Facet collection	All facets as list

Facet

This object represents a single facet with its computed value effective for a specific type.

Property	Type	Description
LocalName	string	Facet name
NamespaceURI	string	Facet namespace
FacetType	string	one of "normalization", "lexicalspace", "valuespace-length", "valuespace-enum" or "valuespace-range"
DeclaringType	Type 1123	Type this facet is declared on
FacetCheckerName	string	Name of facet checker (from schemafacet map)
FacetValue	string or integer	Actual value of this facet

(C) 2015-2021 Altova GmbH

28 メニューコマンド

ユーザーリファレンスには、XMLSpy のメニューコマンドに関する全てが記述されており、それらの一般的な使用方法について説明します。網羅的な内容になるよう努力はしておりますが、ユーザーリファレンスで扱われていない事柄に関する質問などありましたら、Altova W eb サイトのFAQ またはディスカッションフォーラムを参照ください。適切な答えが見つからない場合、Altova サポートセンター[1501]にお問い合わせください。

準的なW indows コマンド(開く、保存、切り取り、コピー、貼り付けなど)は、ファイル[1128]と編集[1148] メニューに収められています。これらのメニューにはXML やインターネットに関するコマンドも収められています。

28.1 ファイルメニュー

ファイルメニューには、一般的な Windows ソフトウェア製品で使用されるような 新規作成[1128]、開く[1132]、保存[1138]、印刷[1144]、印刷設定[1146]、と終了[1146] などのファイルの操作を行うためのコマンドが収められています。XMLSpy には、XML 固有およびアプリケーション固有のコマンドが含まれています。

28.1.1 新規作成

このセクション:

- アイコンとショートカット[1129]
- 説明[1129]
- 新規ドキュメントのテンプレートを作成[1129]
- DTD/XML スキーマを新規 XML ドキュメントへ割り当て[1130]
- 新規 XML ドキュメントのドキュメント要素を指定[1131]
- 新規ドキュメントの作成時に SPS を割り当てる[1131]
- XBRL タクソノミウィザードにて新規 XBRL タクソノミを作成する[1132]

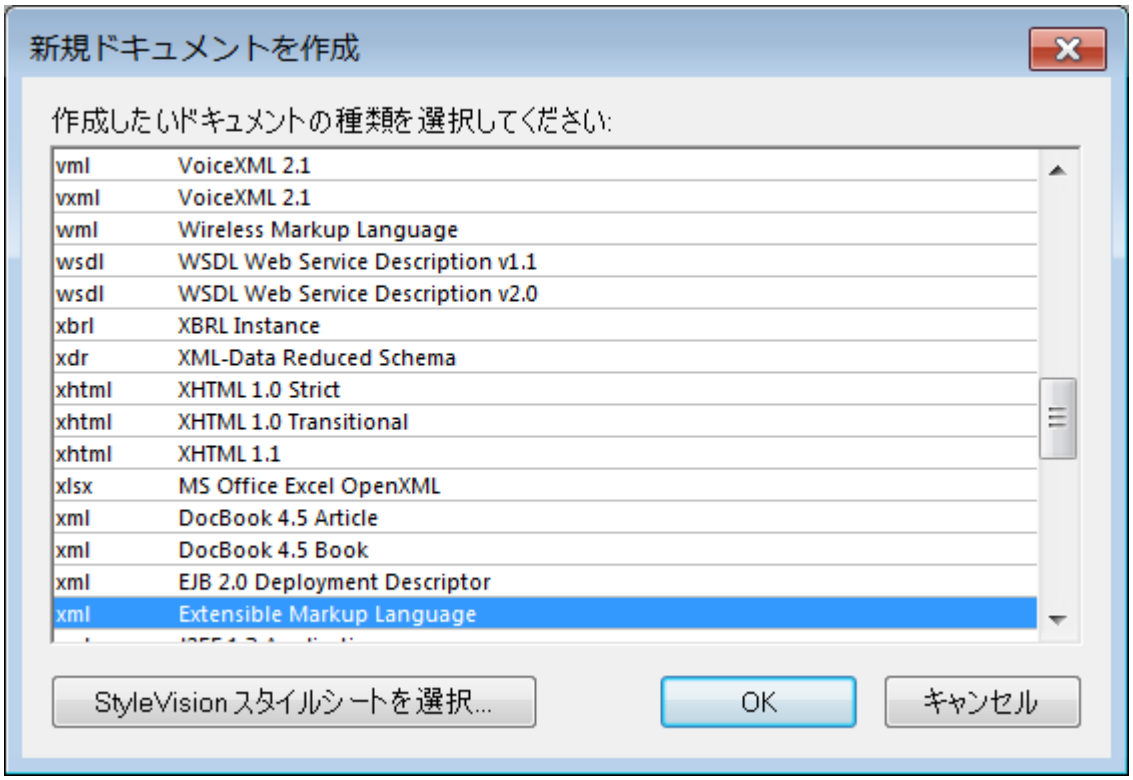
(C) 2015-2021 Altova GmbH

アイコンとショートカット

アイコン:	
ショートカット:	**Ctrl+N**

説明

新規作成コマンドは、新しいドキュメントを作成するために使用されます。**新規作成**をクリックすると、新規ドキュメントを作成ダイアログが表示され、作成するドキュメントの種類を選択することができます。作成したい種類のドキュメントが表示されていない場合は、XML を選択して、ファイルを保存する際に拡張子を変更してください。「ツール | オプション | ファイルの型[1458]」セクションから新しい種類のファイルを追加することができます。



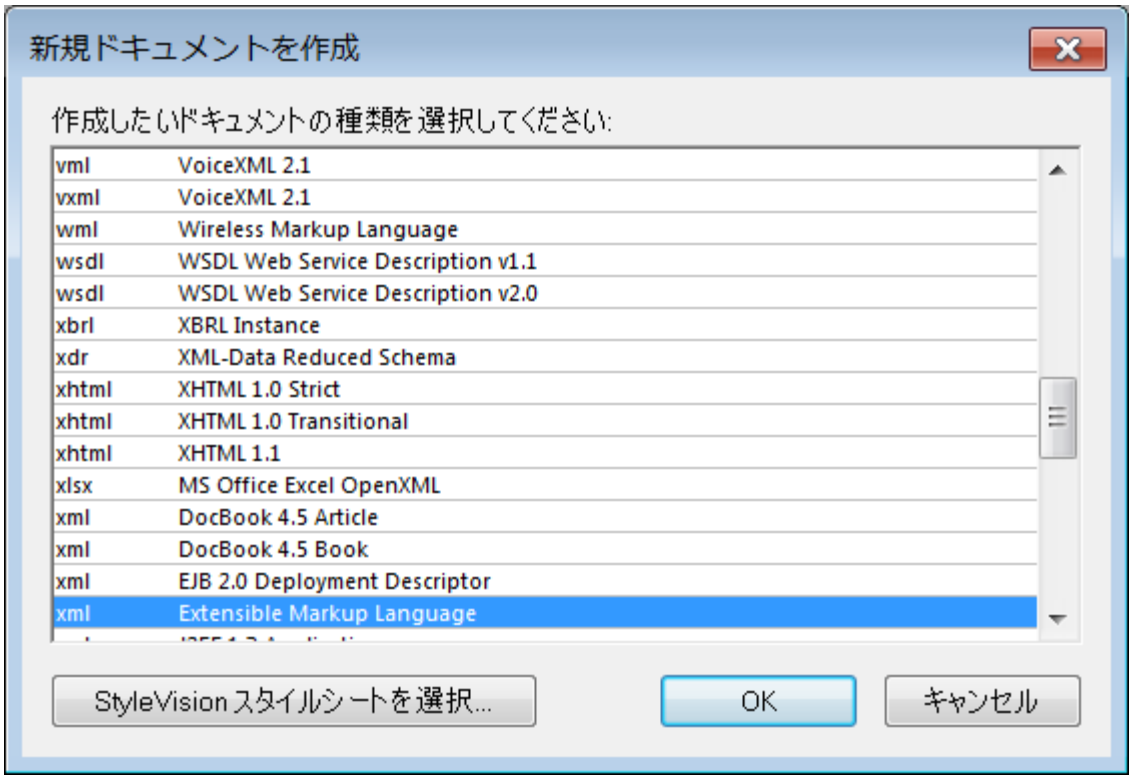
新規ドキュメントのテンプレート

様々な種類のファイルに対して複数のテンプレートを作成することができます。これらのテンプレートは、新規ドキュメントを作成ダイアログから直接開かれ、編集することができるようになります。新規ドキュメントを作成ダイアログに表示される独自のテンプレートを作成するには、まずテンプレートドキュメント作成し、全てのテンプレートが含まれるフォルダー内部にそのファイルを保存します。

以下の操作を行ってください:

1. Windows のエクスプローラーや、その他のナビゲーションツールを使用して、アプリケーション フォルダー[33] の `XMLSpy\Template` フォルダーを開きます。new.xxx という名前の基本的なファイルを選択します(.xxx というのが .xml や .xslt といったファイルの拡張子になります)。
2. そのファイルを XMLSpy で開き、必要な変更を行います。このファイルがテンプレートファイルになります。

3. 編集が終了したら、「**ファイル | 名前を付けて保存**」を選択し、ファイルを ¥Template フォルダーへ適当な名前とともに保存しなおします（ここでは例として「MyXMLTemplate.xml」という名前で保存しています）。これで「MyXMLTemplate」という名前のテンプレートが作成され、新規ドキュメントを作成ダイアログのリストに表示されるようになります。



4. テンプレートを開くには、「**ファイル | 新規作成**」を選択し、そのテンプレート（この場合は「my-xml」）を選択します。

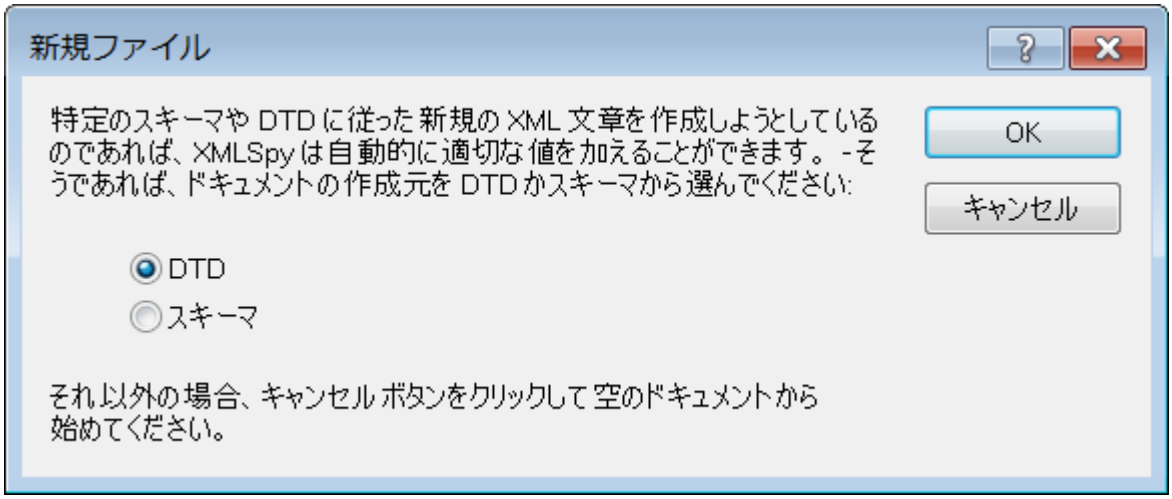
テンプレートを削除するには、該当するテンプレートファイルをテンプレートフォルダーから削除してください。

DTD/XML スキーマを新規 XML ドキュメントへ割り当て

標準的なスキーマ（DTD または XML スキーマ）をベースにした種類のドキュメントを新たに作成すると、適切な DTD または XML スキーマの割り当てとともにドキュメントは開かれます。例えば、XHTML ファイルは、DTD http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd が割り当てられた状態で開かれます。XML スキーマ（.xsd）ファイルは http://www.w3.org/2001/XMLschema スキーマドキュメントが割り当てられた状態で開かれます。

スキーマが定まっていないファイル（例えば空の XML ファイル）を新たに作成する場合、作成するドキュメントにスキーマ（DTD または XML スキーマ）を割り当てるよう促されます。

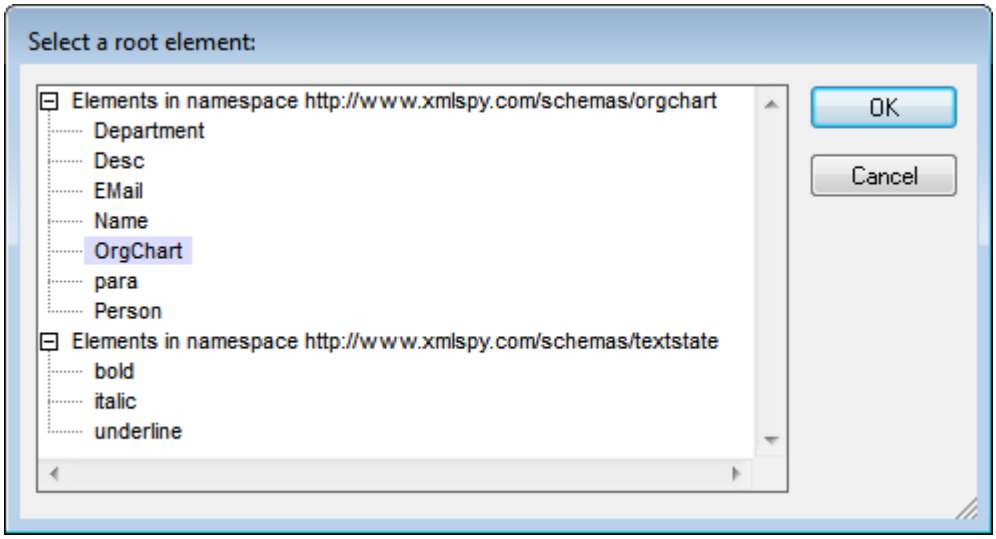
(C) 2015-2021 Altova GmbH



ドキュメントに対して DTD または XML スキーマの割り当てを行う場合は、「OK」をクリックすることでスキーマを参照することができます。このダイアログの「キャンセル」をクリックすると、どのスキーマも割り当てられていないファイルが作成されます。

新規 XML ドキュメントのドキュメント要素を指定

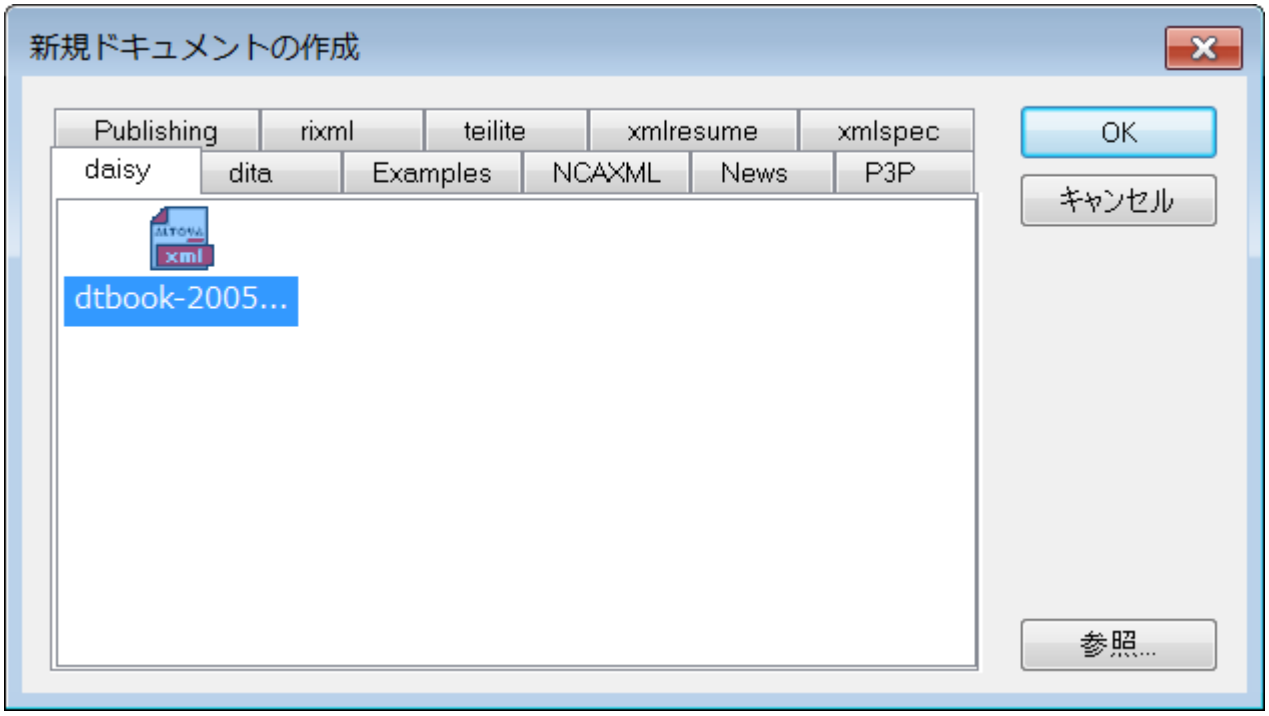
XML スキーマを選択する際に、そのスキーマ内にドキュメント（ルート）要素となることができるグローバル要素が複数個格納されている場合があります。新規ファイルダイアログにて選択された XML スキーマに、2つ以上のグローバル要素が定義されている場合、ルート要素を選択ダイアログが表示されます。



「OK」をクリックすると、この要素 (`OrgChart`) をルート要素とした XML ドキュメントが新たに作成されます。

新規ドキュメントの作成時に SPS を割り当てる

XML ドキュメントが新たに作成されると、ドキュメントを Authentic View で閲覧するために、StyleVision Power Stylesheet (`.sps` ファイル) を割り当てることができます。（上のスクリーンショットにある）新規ドキュメントを作成ダイアログにて、StyleVision スタイルシートを選択をクリックすると、新規ドキュメントの作成ダイアログが表示されます（以下のスクリーンショットを参照）。



フォルダータブから目的の SPS を参照することができます。他にも、参照ボタンをクリックして、目的のStyleVision Power スタイルシートを選択することができます。

XBRL タクソノミウィザードにて新規 XBRL タクソノミを作成する

新規ドキュメントを作成ダイアログにて XBRL タクソノミスキーマ(.xsd) が選択されると、XBRL タクソノミを新規に作成するウィザードが起動します。ウィザードに関しては、ドキュメンテーションのXBRL セクション[741] にて記述されます。

28.1.2 開く

アイコンとショートカット

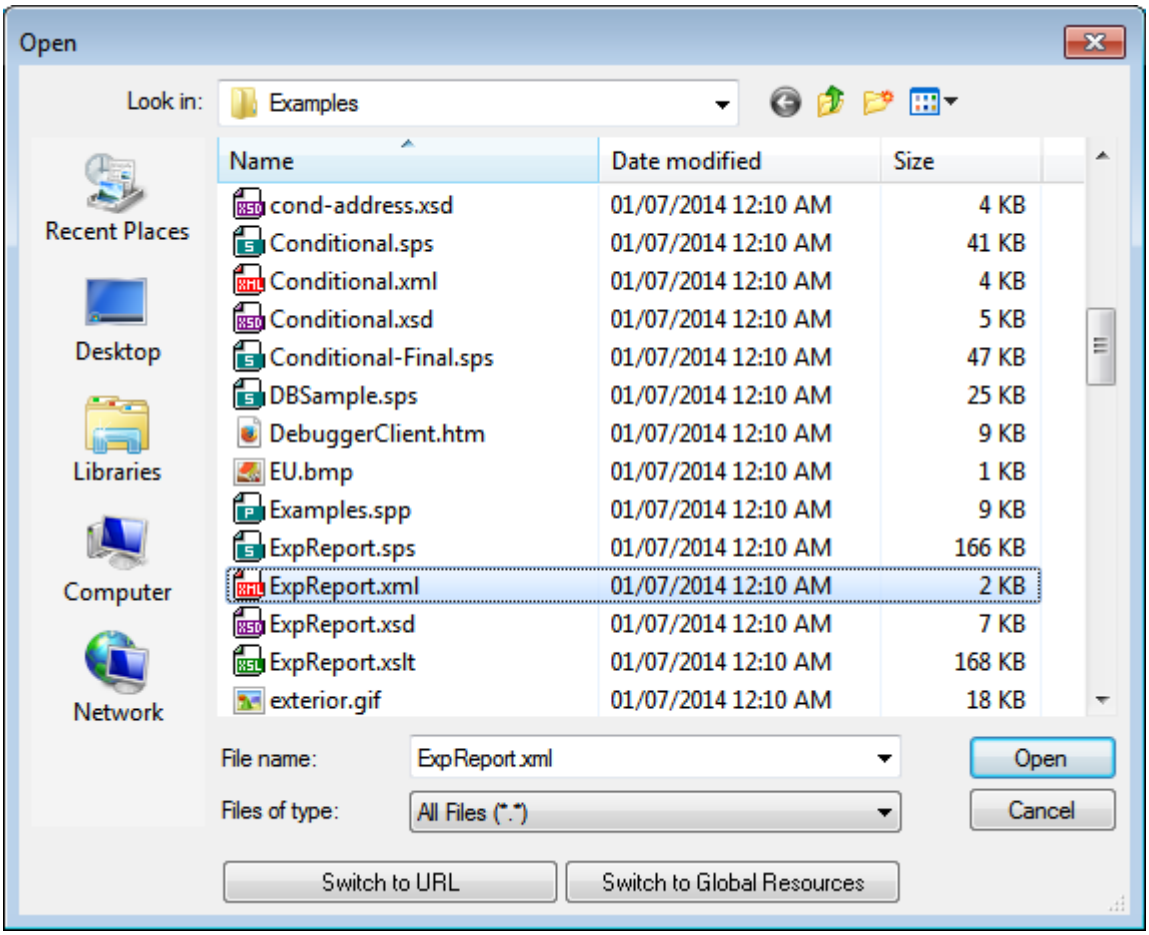
アイコン:	
ショートカット:	**Ctrl+O**

説明

開くコマンドにより、慣れ親しんだW indows の開くダイアログが表示され、XML に関するドキュメントや、テキストドキュメントを開くことができます。開くダイアログでは、2つ以上のファイルを開くことができます。ファイルの種類コンボボックスを使用することで、ダイアログボックスに表示されるファイルの型セクションを制限することができます(表示されるファイルの種類一覧は、オプションダイアログ(「ツール | オプション[1458]」)にて設定することができます)。XML ファイルが開かれると、整形式のチェックが行われます。ファイルが整形式で無い場合、整形式に関するエラーが表示されます。エラー箇所を修正して、メニューコマンドからXML | 整形式のチェック(F7)[1225]を選択します。 ファイルを開く再に自動検証[1455] を行うよう選択しており、ファイルが妥当でない場合、エラーメッセージが表示されます。エラー箇所を修正して、メニューコマンドから「 XML | XML の検証 (F8)[1227] 」を選択して、再度検証を行います。

▼ URL とグローバルリソースを使用してファイルを選択または保存する

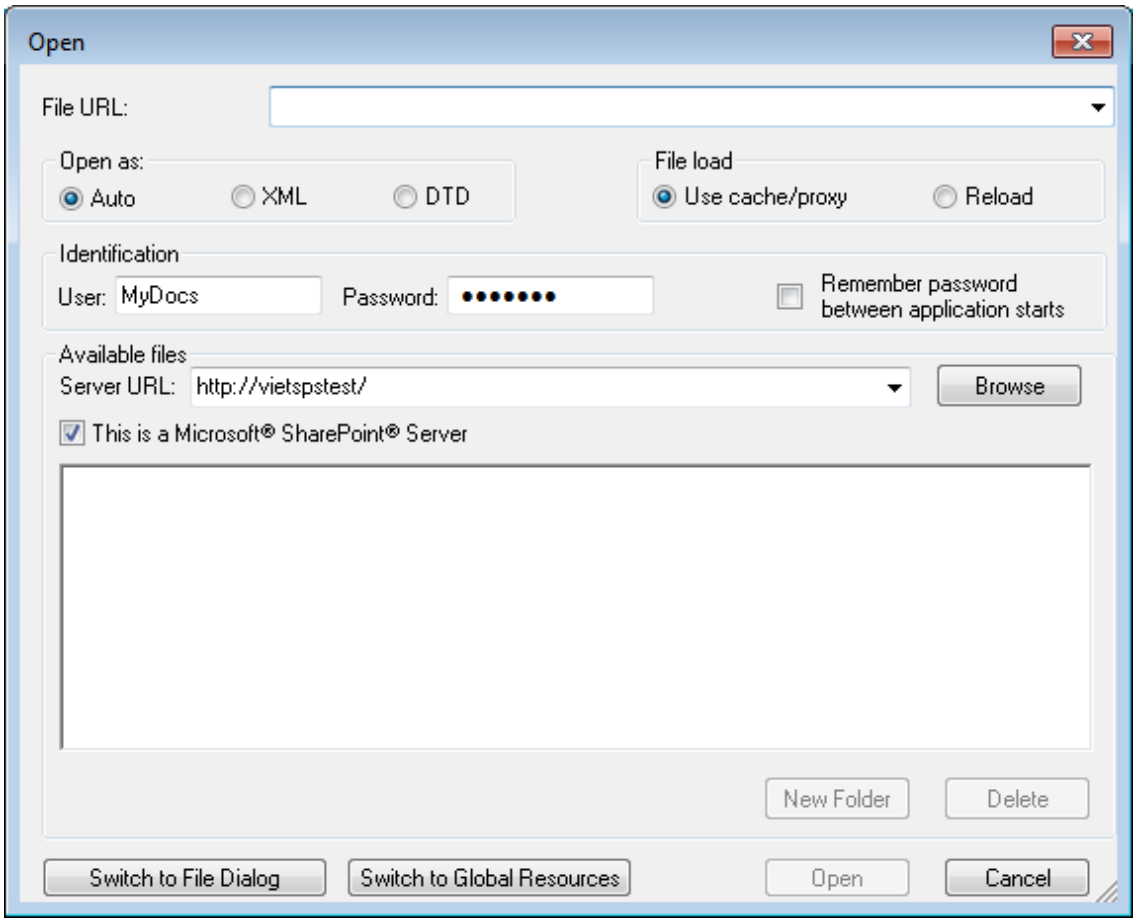
「ファイルを開く」および「ファイルの保存」ダイアログでは、URL またはグローバルリソースを使用して必要なファイルを選択またはファイルを保存することができます（*下のスクリーンショット参照*）。選択プロセスに移動するために「URL に切り替える」または「グローバルリソース」をクリックしてください。



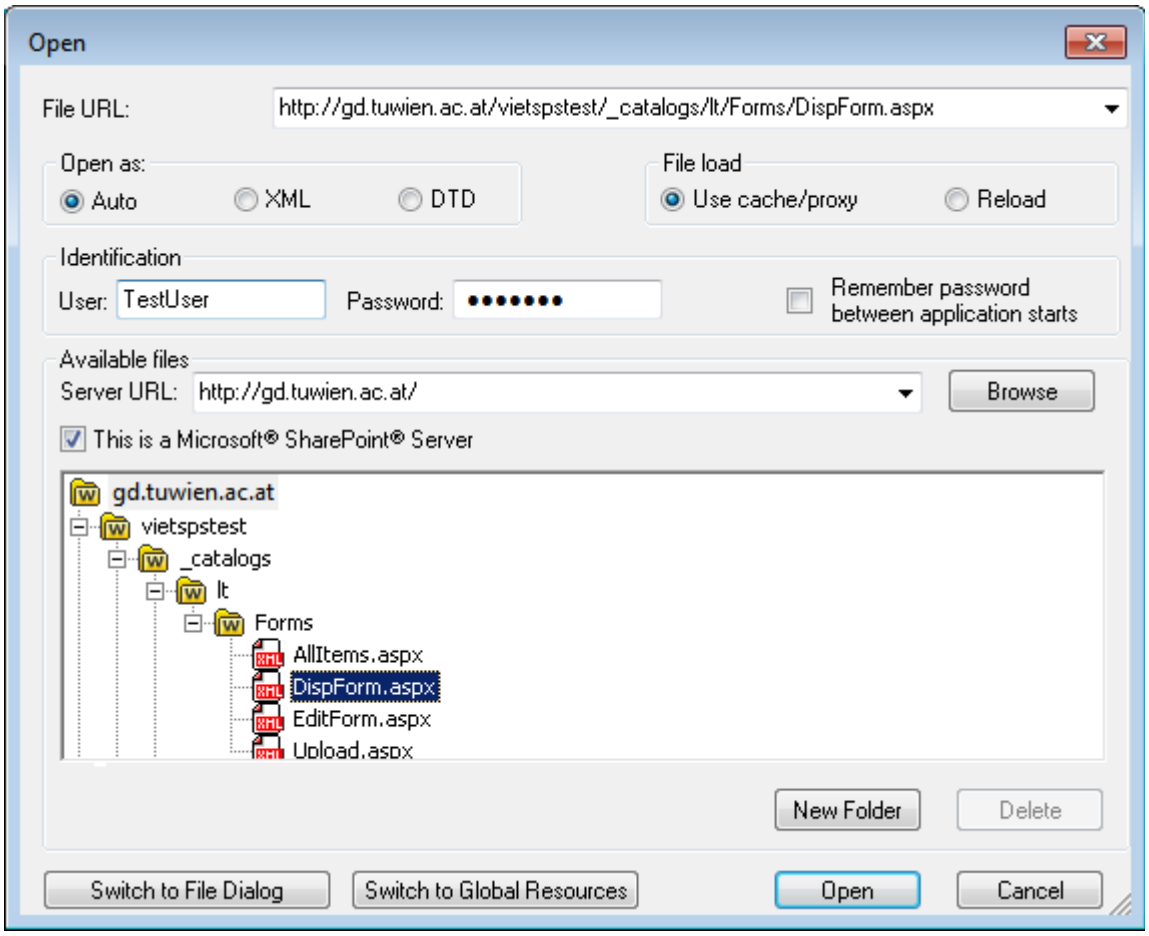
URL を使用してファイルを切り替える

URL から（開くまたは保存する）ファイルを選択するには以下を行ってください：

1. 「URL に切り替える」コマンドをクリックします。これにより、開くまたは保存ダイアログの URL モードにスイッチされます。（*下のスクリーンショットは開くダイアログを表示しています*）。



2. アクセスするURLをサーバー*URL*フィールドに入力します（上のスクリーンショット）。サーバーがMicrosoft® SharePoint® Serverの場合*Microsoft® SharePoint® Server*チェックボックスをチェックしてください。この種類のサーバー上にあるファイルと作業する場合、下のMicrosoft® SharePoint® Serverメモを参照してください。
3. サーバーがパスワードにより保護されている場合、*ユーザー*と*パスワード*フィールドに入力してください。
4. 「参照」をクリックして、サーバーのディレクトリ構造をナビゲートします。
5. フォルダーツリー内でロードするファイルを参照してクリックします。



ファイルURL がファイルURL フィールドに表示されます（*上のスクリーンショット参照*）。「開く」または「保存」ボタンを使用できるようになります。

6. 「開く」をクリックしてファイルをロードするか、「保存」をクリックして保存します。

以下の点に注意してください：

- WebDAV をサポートするサーバーとMicrosoft SharePoint Servers のみで参照機能を使用することができます。サポートされるプロトコールはFTP、HTTP およびHTTPS です。
- ファイルを開く際のロード処理を更に管理するには、ローカルのキャッシュまたはファイルが以前にロードされている場合スピードを向上するプロキシのサーバーからファイルをロードします。（ファイルが事前にロードされることにより更に処理スピードを向上することができます）。また、電子出版またはデータベースシステムなどの作業中のファイルを再ロードして、「再ロード」オプションを選択します

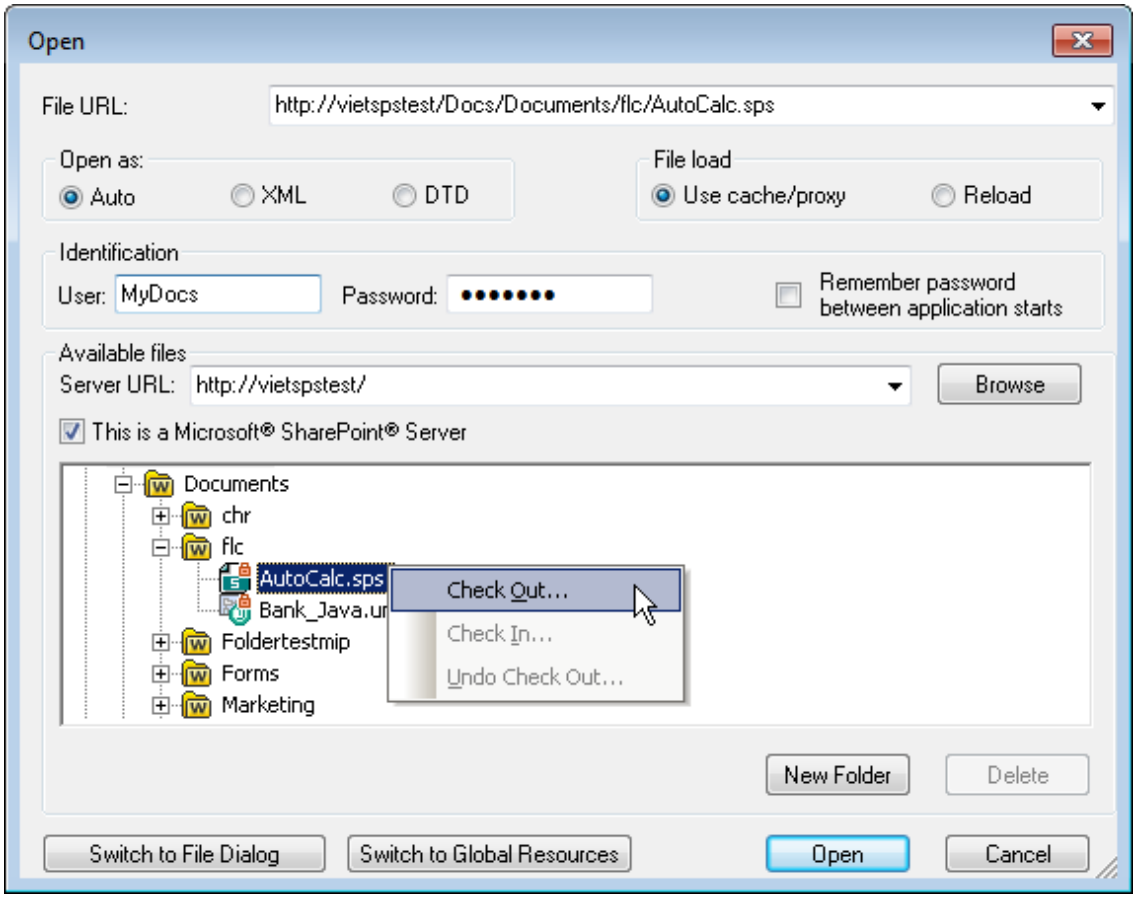
。

▼ Microsoft® SharePoint® Server メモ

Microsoft® SharePoint® Servers に関する以下の点に注意してください：

- 「利用可能なファイル」ペインに表示されるディレクトリ構造では、ファイルアイコンにはファイルのチェックイン/チェックアウトの状

態を表示するシンボルがあります（*下のスクリーンショット*）。



ファイルを右クリックすると、ファイルで使用することのできるコンテキストメニューがポップアップします（*上のスクリーンショット*）。

- 異なるファイルアイコンが以下に表示されています:

	チェックイン状態。
	チェックアウト可能。他のユーザーによりチェックアウト状態。
	チェックアウト不可能。ローカルに対してチェックアウト。

- ファイルをチェックアウトした後、Altova アプリケーションを使用して編集することができ、「**ファイル | 保存 (Ctrl+S)**」を使用して保存することができます。
- 編集されたファイルを「URL を開く」ダイアログのコンテキストメニューからチェックインすることができます（*上のスクリーンショット参照*）または、アプリケーションのメインウィンドウのファイルタブを右クリックすることによりポップアップするコンテキストメニューを使用して（*下のスクリーンショット*）。

- ファイルが他のユーザーによりチェックアウトされている場合、チェックアウトすることができません。
- 他のユーザーによりファイルがチェックアウトされている場合、チェックアウトを行うことはできません。これによりファイルを変更す

ることなくサーバーに戻すことができます。
- 変更することなしにファイルがサーバーへ戻されます。あるAltova 製品でファイルをチェックアウトした場合、他のAltova 製品で同じファイルをチェックアウトすることはできません。この時点で使用できるMicrosoft® SharePoint® Server をサポートするAltova アプリケーションのコマンドは、以下の通りです:「**チェックイン**」および「**チェックアウトを元に戻す**」です。

▼ グローバルリソースを使用してファイルを選択または保存する

グローバルリソースを使用してファイルを開くまたは保存する場合、「**グローバルリソース**」をクリックします。グローバルリソースを選択するためのダイアログが表示されます。これらのダイアログは以下のセクションで説明されています: グローバルリソースの使用[944]. グローバルリソースの説明に関してはこのドキュメントのグローバルリソース[934] セクションを参照してください。

28.1.3 再ロード

アイコン

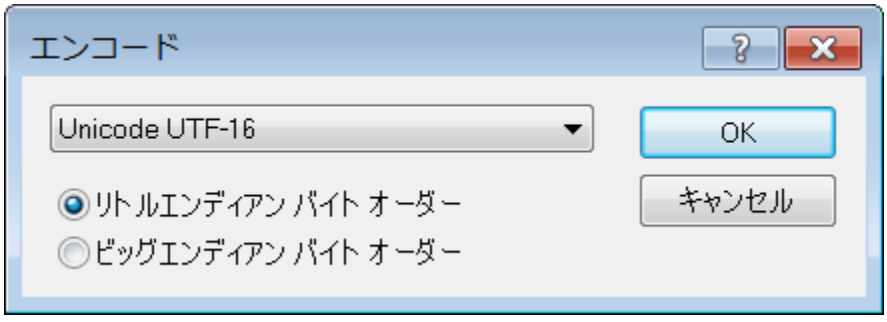
アイコン:	

説明

XMLSpy 外で変更された開かれているドキュメントを再ロードします。XMLSpy 外でドキュメントが変更されると、変更されたドキュメントの再ロードを問うプロンプトが表示されます。再ロードを選択すると、ファイルが最後に保存された時点からの変更が失われます。

28.1.4 エンコード

エンコードコマンドにより、(XML 以外も含む)現在アクティブなドキュメントにて使用されているエンコーディングを確認し、他のエンコーディングを選択して、次回の保存時に反映させることができます。



XMLドキュメントにおいて、それまで使われていたものとは異なるエンコーディングを指定すると、XML 宣言内のエンコーディング指定もそれに合わせて変更されます。2バイト文字ならびに4バイト文字エンコーディング(UTF-16、UCS-2、UCS-4)では、ファイルに使用されるバイトオーダーを指定することができます。XMLドキュメントのエンコーディングを変更するもう1つの方法は、ドキュメントのXML 宣言で使用されて

いるencoding 属性を直接編集することです。既存ならびに新規 XML ドキュメント、または非 XML ドキュメントのデフォルトエンコーディングは、オプションダイアログのエンコードセクション[1460]にて指定することができます。

メモ: ドキュメントを保存する際、XMLSpy はエンコーディングの指定を自動的にチェックして、ユーザーにより入力された エンコーディングが認識できない場合は、ダイアログを表示します。更に、選択された エンコーディングにより表現することのできない文字がドキュメントに含まれている場合、ファイルを保存する際に警告メッセージが表示されます。

28.1.5 閉じる、全て閉じる、非アクティブを全て閉じる

閉じる

閉じるコマンドにより、アクティブなドキュメントウィンドウが閉じられます。ファイルが変更されている場合(タイトルバーに表示されるファイル名の後にアスタリスク＊が表示されます)、まずファイルを保存するか尋ねられます。

全て閉じる

全て閉じるコマンドにより、開かれているドキュメントウィンドウが全て閉じられます。ドキュメントのどれかが変更されている場合、まずファイルを保存するか尋ねられます。

非アクティブを全て閉じる

非アクティブを全て閉じるコマンドにより、アクティブなドキュメントウィンドウ以外の、全ウィンドウが閉じられます。ドキュメントのどれかが変更されている場合、まずファイルを保存するか尋ねられます。

28.1.6 保存、名前を付けて保存、全て保存

アイコンとショートカット

コマンド	*アイコン*	*ショートカット*
保存		Ctrl+S
名前を付けて保存		

保存

保存 (Ctrl+S) コマンドにより、アクティブなドキュメントのコンテンツを開かれた元のファイルに保存しなおします。ドキュメントを保存する際、整形式が自動的にチェック[1225]されます。オプションダイアログ(ツール | オプション[1455])のファイルセクションにてオプションが指定されていれば、ファイルの検証も同時に行われます。XML 宣言のエンコード[1460]指定もチェックされ、ファイルの保存時にエンコーディングがドキュメントに対して適用されます。

名前をつけて保存

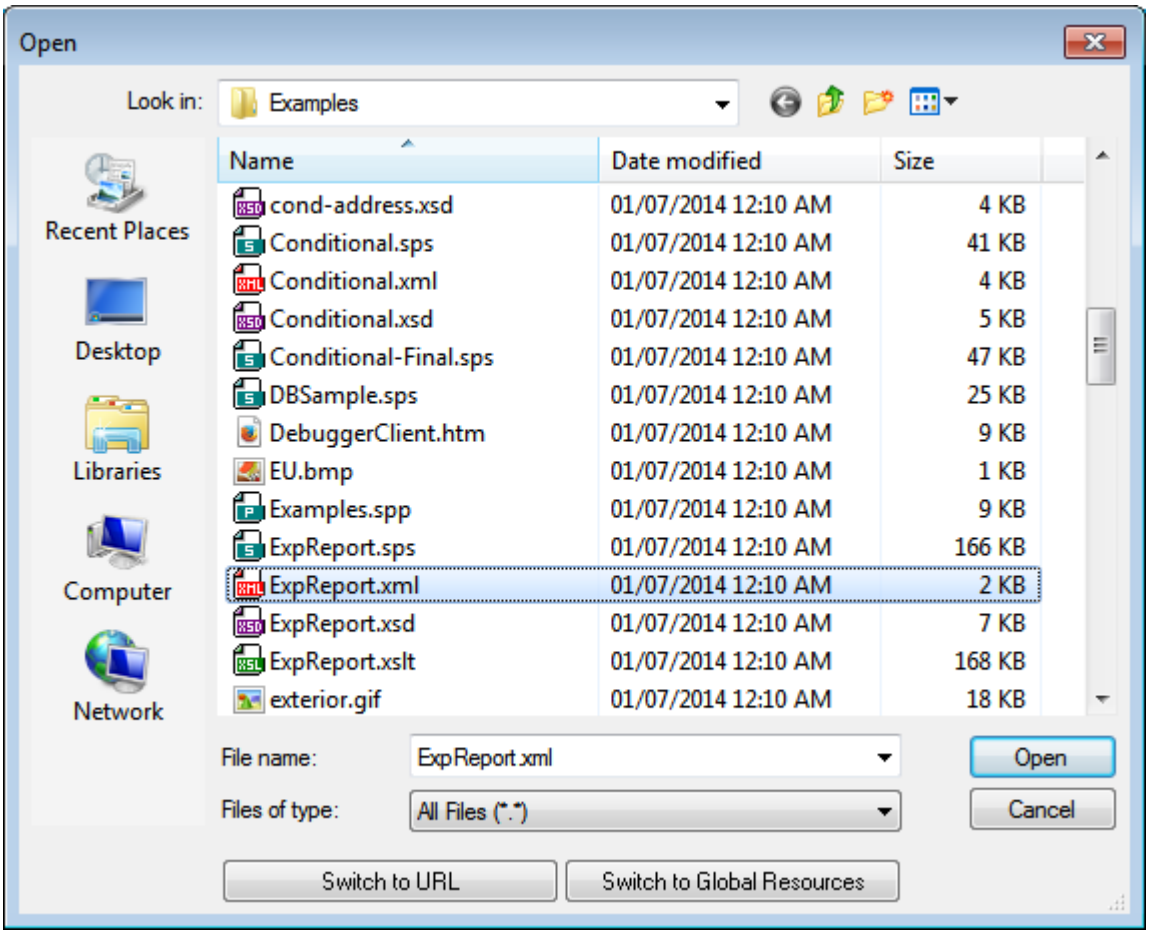
名前を付けて保存コマンドを選択すると、馴染みのある W indows 名前を付けて保存ダイアログボックスが表示され、アクティブなドキュメントの内容を保存するファイルの場所と名前を入力することになります。保存コマンドに対して行われるものと同じチェックがここでも行われます。

全て保存

全て保存 コマンドでは、開かれているドキュメントに対して行われた全ての変更を保存します。このコマンドは、複数のドキュメントを同時に保存する際に便利な機能になります。まだドキュメントが保存されていない場合（例えば、ファイルが新たに作成された後、まだ保存されていない場合）、そのドキュメントに対して名前を付けて保存ダイアログボックスが表示されます。

URL とグローバルリソースを使用してファイルを選択または保存する

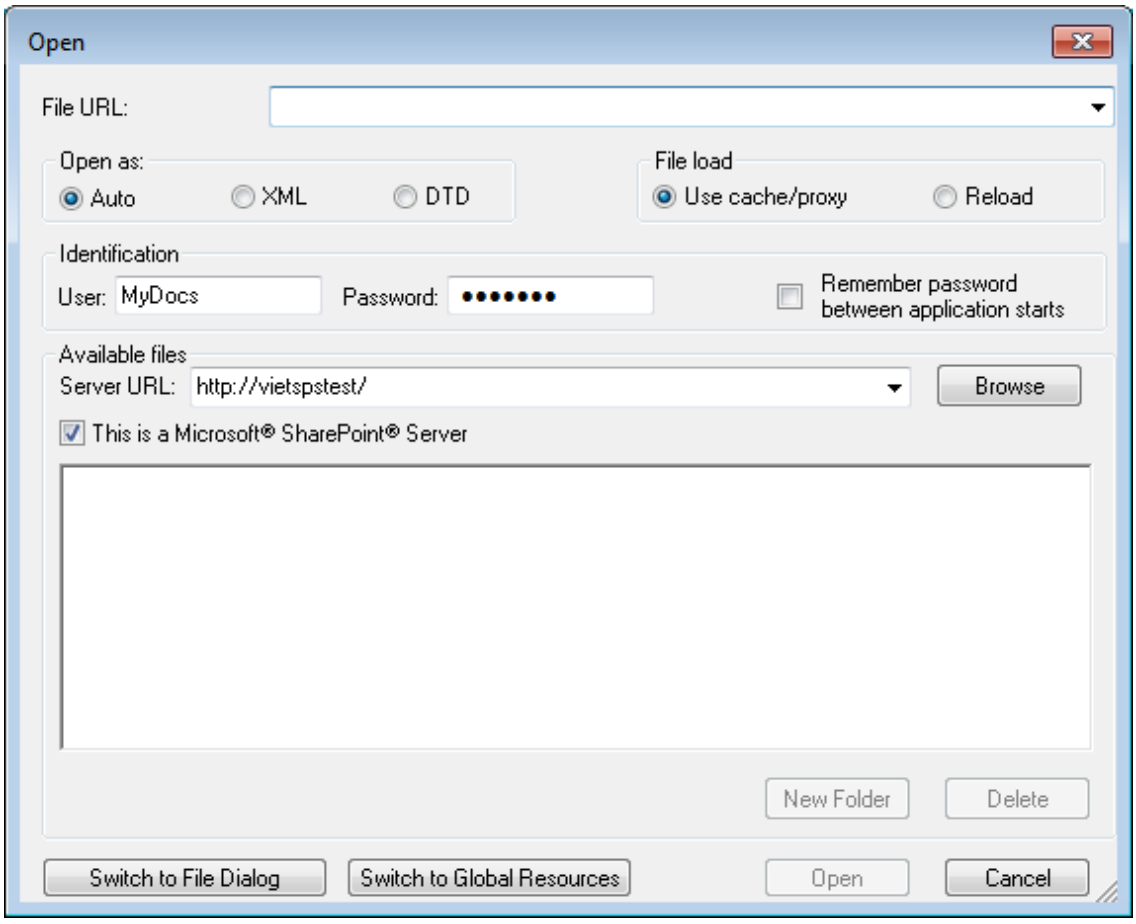
「ファイルを開く」および「ファイルの保存」ダイアログでは、URL またはグローバルリソースを使用して必要なファイルを選択またはファイルを保存することができます（*下のスクリーンショット参照*）。選択プロセスに移動するために「**URL に切り替える**」または「**グローバルリソース**」をクリックしてください。



URL を使用してファイルを切り替える

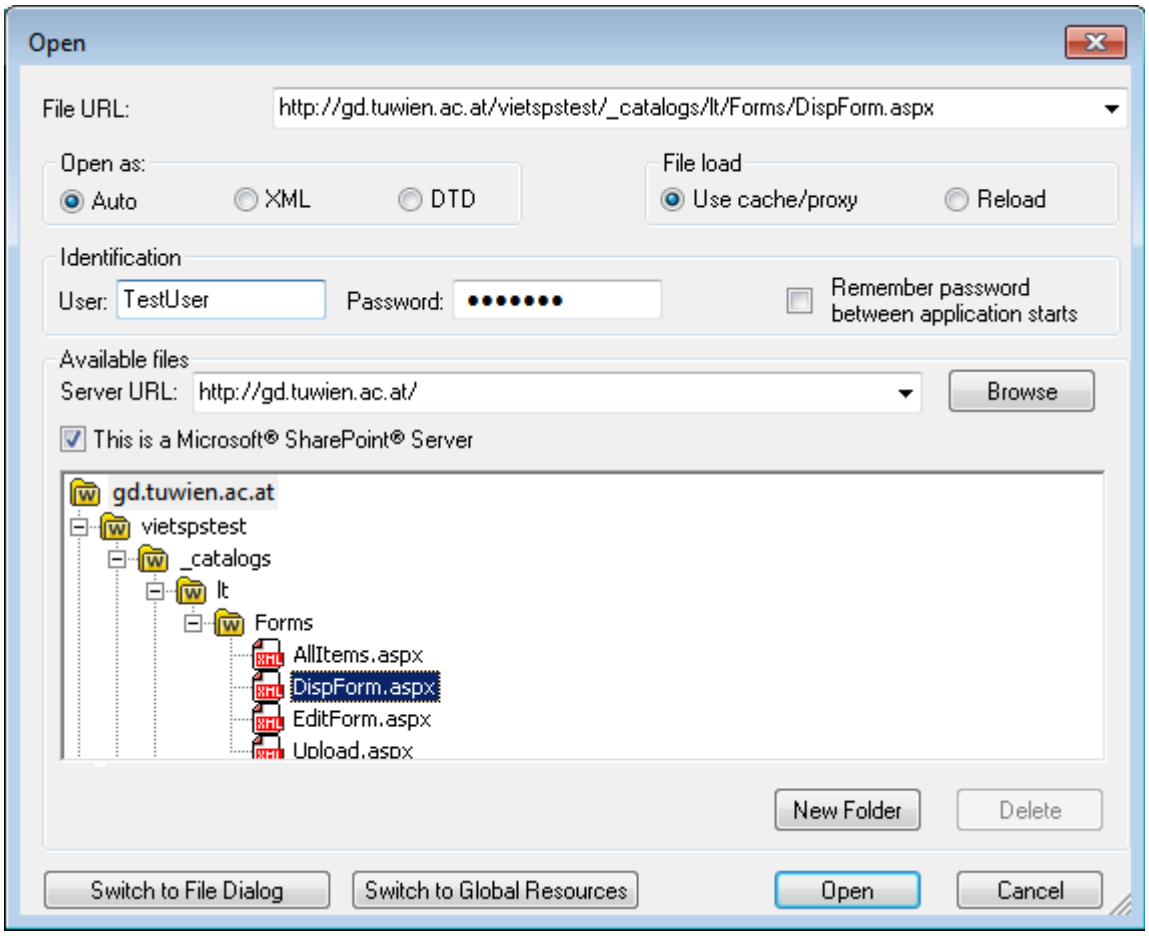
URL から（開くまたは保存する）ファイルを選択するには以下を行ってください：

1. 「**URL に切り替える**」コマンドをクリックします。これにより、開くまたは保存ダイアログの URL モードにスイッチされます。（*下のスクリーンショットは開くダイアログを表示しています*）。



2. アクセスするURLをサーバー*URL*フィールドに入力します（上のスクリーンショット）。サーバーがMicrosoft® SharePoint® Serverの場合*Microsoft® SharePoint® Server*チェックボックスをチェックしてください。この種類のサーバー上にあるファイルと作業する場合、下のMicrosoft® SharePoint® Server メモを参照してください。
3. サーバーがパスワードにより保護されている場合、*ユーザー*と*パスワード*フィールドに入力してください。
4. 「参照」をクリックして、サーバーのディレクトリ構造をナビゲートします。
5. フォルダーツリー内でロードするファイルを参照してクリックします。

(C) 2015-2021 Altova GmbH



ファイルURL がファイルURL フィールドに表示されます（*上のスクリーンショット参照*）。「開く」または「保存」ボタンを使用できるようになります。

6. 「開く」をクリックしてファイルをロードするか、「保存」をクリックして保存します。

以下の点に注意してください：

- WebDAV をサポートするサーバーとMicrosoft SharePoint Servers のみで参照機能を使用することができます。サポートされるプロトコールはFTP、HTTP およびHTTPS です。
- ファイルを開く際のロード処理を更に管理するには、ローカルのキャッシュまたはファイルが以前にロードされている場合スピードを向上するプロキシのサーバーからファイルをロードします。（ファイルが事前にロードされることにより更に処理スピードを向上することができます）。また、電子出版またはデータベースシステムなどの作業中のファイルを再ロードして、「再ロード」オプションを選択します

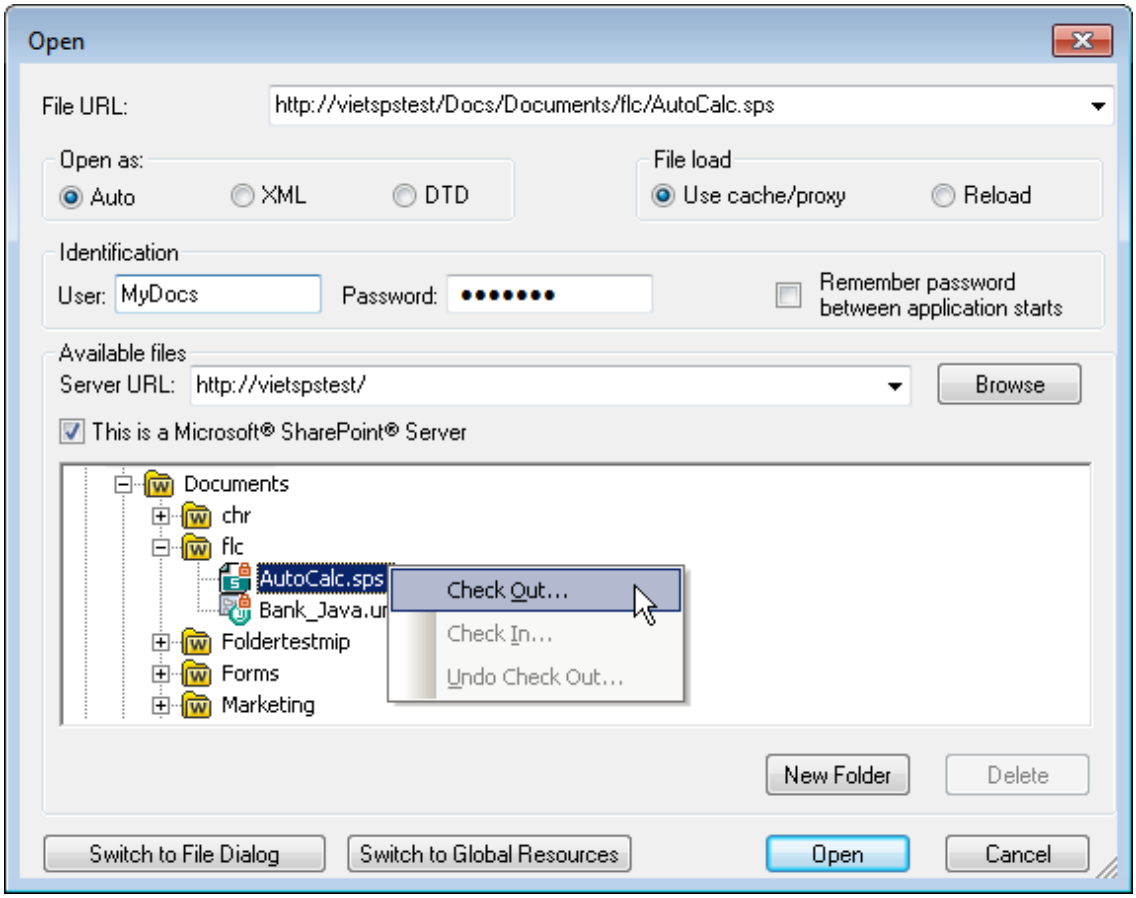
。

▼ Microsoft® SharePoint® Server メモ

Microsoft® SharePoint® Servers に関する以下の点に注意してください：

- 「利用可能なファイル」ペインに表示されるディレクトリ構造では、ファイルアイコンにはファイルのチェックイン/チェックアウトの状

態を表示するシンボルがあります（*下のスクリーンショット*）。



ファイルを右クリックすると、ファイルで使用することのできるコンテキストメニューがポップアップします（*上のスクリーンショット*）。

- 異なるファイルアイコンが以下に表示されています:

	チェックイン状態。
	チェックアウト可能。他のユーザーによりチェックアウト状態。
	チェックアウト不可能。ローカルに対してチェックアウト。

- ファイルをチェックアウトした後、Altova アプリケーションを使用して編集することができ、「**ファイル | 保存 (Ctrl+S)**」を使用して保存することができます。
- 編集されたファイルを「URL を開く」ダイアログのコンテキストメニューからチェックインすることができます（*上のスクリーンショット参照*）または、アプリケーションのメインウィンドウのファイルタブを右クリックすることによりポップアップするコンテキストメニューを使用して（*下のスクリーンショット*）。

- ファイルが他のユーザーによりチェックアウトされている場合、チェックアウトすることができません。
- 他のユーザーによりファイルがチェックアウトされている場合、チェックアウトを行うことはできません。これによりファイルを変更す

ることなくサーバーに戻すことができます。
- 変更することなしにファイルがサーバーへ戻されます。あるAltova 製品でファイルをチェックアウトした場合、他のAltova 製品で同じファイルをチェックアウトすることはできません。この時点で使用できるMicrosoft® SharePoint® Server をサポートするAltova アプリケーションのコマンドは、以下の通りです:「**チェックイン**」および「**チェックアウトを元に戻す**」です。

▼ グローバルリソースを使用してファイルを選択または保存する

グローバルリソースを使用してファイルを開くまたは保存する場合、「**グローバルリソース**」をクリックします。グローバルリソースを選択するためのダイアログが表示されます。これらのダイアログは以下のセクションで説明されています: グローバルリソースの使用 944. グローバルリソースの説明に関してはこのドキュメントの グローバルリソース 934 セクションを参照してください。

28.1.7 メールで送信

アイコン

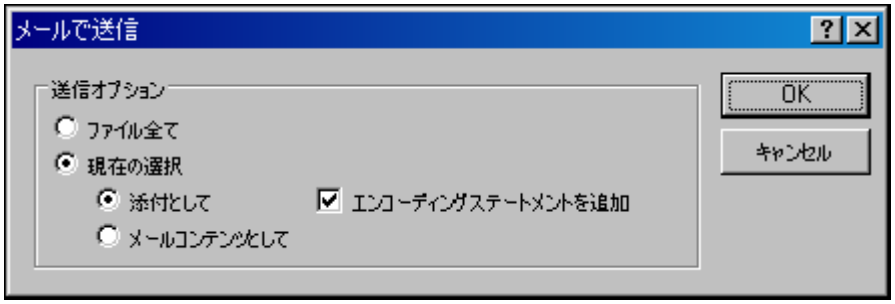
アイコン:	

説明

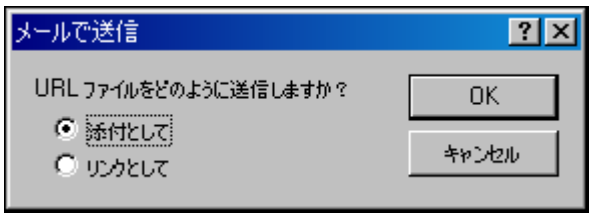
「**電子メールにて送信**」コマンドによりXML ドキュメント、または、XML ドキュメントからの選択を電子メールにて送信することができます。ドキュメント 、または、ドキュメントの選択の種類により、ドキュメントを添付、コンテンツ、または、リンクとして送信することができます。詳細に関しては以下を参照してください。

送信できる内容	送信の方法
アクティブなXML ドキュメント	電子メールの添付
アクティブなXML ドキュメント内の選択	電子メールの添付または電子メールの内容として送信
プロジェクトウィンドウ内の1つまたは複数のファイル	電子メールの添付
プロジェクトウィンドウ内の1つまたは複数のURL	電子メールの添付またはリンクとして送信

「**電子メールにて送信**」コマンドがアクティブなXML ドキュメント内で呼び出されると、「電子メールにて送信」ダイアログがポップアップし、スクリーンショット内に送信のオプションが表示されます(*下のスクリーンショット*) 。アクティブなファイル内でテキストが選択されておらず、「**電子メールにて送信**」コマンドが呼び出されると、*ファイル全てラジオボタンのみ*が有効化され、他のオプションは無効化されます(下のスクリーンショットを参照してください) 。



プロジェクトウィンドウから送信されるファイルは、電子メールの添付のみで送信されるため、電子メールで送信ダイアログはスキップされ、選択されたファイルが添付された電子メールが開かれます。プロジェクトウィンドウ内のURL は、添付またはリンクとして送信することができます(*下のスクリーンショットを参照*)。URL を送信する方法を選択し、「OK」をクリックします。



28.1.8 印刷

アイコンとショートカット

アイコン:	
ショートカット:	**Ctrl+P**

説明

印刷コマンドにより、印刷ダイアログボックスが表示され、プリンターのオプションを指定することができます。現在アクティブなドキュメントを、表示されているままに印刷することができます。

グリッドビューにて**印刷**コマンドをクリックすると、印刷オプションダイアログが表示され(以下のスクリーンショットを参照)、XML ドキュメントの印刷オプションを指定することができます。このダイアログにある**印刷**をクリックすることで、プリンターオプションの印刷ダイアログが表示されます。

グリッドビューから印刷に利用することができるオプションは、以下のとおりです:

- 型ペインでは、出力に表示させるアイテムを選択します。
- 印刷範囲オプションでは、現在選択している箇所か、ファイル全体を印刷するかを指定します。
- 展開オプションにより、ドキュメントをそのまま印刷するのか、または子孫要素が全て展開された状態で印刷するのかを指定します。
- コンテンツオプションでは、ノード全体の印刷を行うのか、ノード名だけを印刷するのかを指定します。
- コンテンツ幅がページより広い場合ペインでは、コンテンツがページの幅に収まらない場合の処理を選択します。ページを分割オプションでは、コンテンツを(水平方向ならびに垂直方向で)ページごとに分割することで、ドキュメント全体を通常のサイズで印刷します。印刷されたページをつなぎあわせることで、ポスターを作成することができます。最初のページオプションでは、印刷領域の左端、最初のページのみが印刷されます。水平方向に収まらないコンテンツは印刷されません。このオプションは、グリッドビューに表示されている重要な情報の殆どが左側に表示されているような状態で利用することができます。水平に縮小オプションを使用すると、印刷される内容が水平方向に収まるよう、大きさを(水平方向ならびに垂直方向に)縮小して印刷します。ドキュメントは複数ページになる場合もあります。両方縮小オプションを使うと、ドキュメントが1ページに収まるように縮小されます。
- 印刷ボタンをクリックすることで、選択されたオプションによる印刷が行われます。
- プレビューボタンにより、印刷プレビューウィンドウが表示され、印刷を実際に行う前に、最終的に印刷される内容を確認することができます。
- 印刷設定ボタンをクリックすると、プリンターの設定ダイアログが表示され、この印刷ジョブだけに対しての用紙サイズや印刷の向き、その他のプリンターオプションを設定することができます。印刷設定[1146] コマンドに関する記述も参照ください。

メモ: グリッドビューのカラム幅を変更することで、印刷出力を最適化することができます。

プログラムロゴ

ライセンスを購入されている場合、プログラムロゴ、著作権に関する情報、そして登録に関する詳細をXMLSpy にて印刷されないように指定することができます。オプションダイアログのビューセクション[1466]にてこのオプションをご利用になれます。

WSDL とXBRL デザイン

WSDL ならびにXBRL ドキュメントのグラフィカルなビューの印刷では、画面に表示されたままの印刷が行われます。

28.1.9 印刷プレビュー、印刷設定

印刷プレビュー

テキストビュー、Authentic View、とブラウザービュー内の**印刷プレビュー**コマンドをクリックすると、ボタンをクリックすると、現在アクティブなドキュメントの印刷プレビューが表示されます。グリッドビュー、スキーマビュー、WSDL ビュー、とXBRL ビューから印刷オプションを選択することのできる印刷ダイアログボックスが開かれ、印刷プレビューを表示するために**プレビュー**ボタンをクリックします。

拡大や**縮小**ボタンを使用することで、プレビューの表示を最適化することができます。ページ全体が表示されている状態であれば、**1ページ / 2ページ**ボタンにより、同時に表示されるページの数を切り替えることができます。**次のページ**ならびに**前のページ**ボタンにより、ページ間の移動を行うことができます。ツールバーには前ページの印刷やプレビューウィンドウを閉じるオプションも用意されています。

メモ: 印刷プレビューにて背景色やイメージを有効にするには、(i) Internet Explorer の**ツール**メニューにて、**インターネットオプション**を選択し、詳細設定タブをクリックし、(ii) 設定以下にある印刷にて、*背景の色とイメージを印刷する* チェックボックスにチェックを入れ、(iii)「**OK**」をクリックします。

印刷設定

印刷設定コマンドでは、プリンターに関する印刷設定ダイアログボックスが表示され、用紙のフォーマットや印刷の向きといった設定を行うことができます。これらの設定は、その後行われる印刷ジョブにも引き継がれます。

28.1.10 最近使用されたファイル、終了

最近使用されたファイル

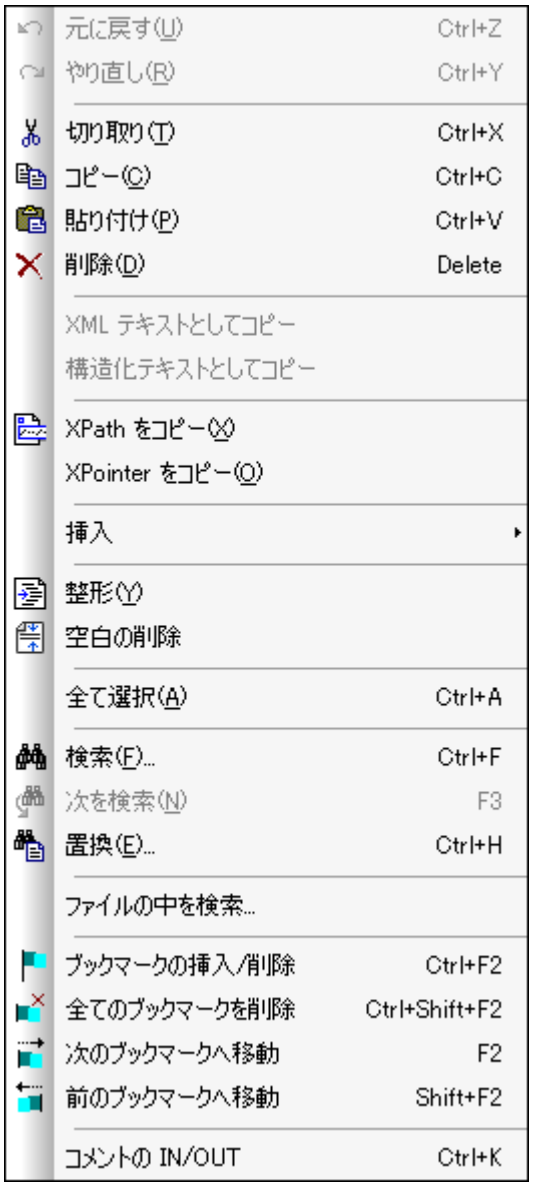
「**ファイル**」メニューの一番下に最近使用した順に9 つのファイルのリストが表示されます。名前をクリックしてファイルを開くことができます。キーボードを使用してファイルを開く場合、「**Alt+F**」を押して「**ファイル**」メニューを開き、開く番号を押します。

終了

XMLSpy を終了します。保存されていない変更のある開かれているファイルがある場合、これらの変更を保存するようにプロンプトされます。XMLSpy はプログラム設定への変更と最近使用されたファイルの情報も保存します。

28.2 編集メニュー

編集メニューには、XMLSpy にて開かれているドキュメントの編集を行うためのコマンドが用意されています。標準的な 元に戻す[1149]、やり直し[1149]、切り取り[1149]、コピー[1149]、貼り付け[1149]、削除[1149]、全て選択[1153]、検索[1159]、次を検索[1159] および 置換[1162] コマンドなどが用意されています。



XMLSpy では以下の様な特別なコマンドが用意されています:

- 選択された箇所を XML テキストとしてコピー[1150],
- 構造化されたテキストとしてコピー[1150]
- 選択されたアイテムの XPath セレクターをクリップボードにコピー[1154]
- ブックマークの挿入や削除やブックマークのナビゲーション

(C) 2015-2021 Altova GmbH

28.2.1 元に戻す、やり直し

アイコンとショートカット

コマンド	アイコン	ショートカット
元に戻す		Ctrl+Z
やり直し		Ctrl+Y

元に戻す

元に戻す コマンド では、回数に制限の無いアンドゥ(作業の取り消し)がサポートされます。この機能を使って行われた全ての操作を取り消すことができます。元に戻すコマンドの記録は、ドキュメントが保存されても蓄えられ、変更を保存する前の状態まで操作を取り消すことができます。**元に戻す** と **やり直し** コマンドを使用して履歴を戻り、また、先に進むことができます(*やり直し コマンドを参照してください*)。

やり直し

やり直し コマンド では、それまでに取り消されたコマンドをやり直し、それまで行われた作業の記録を辿ることができます。**元に戻す**や、**やり直し**コマンドを使用することで、これまでの記録を行き来することができます。

28.2.2 切り取り、コピー、貼り付け、削除

アイコンとショートカット

コマンド	アイコン	ショートカット
切り取り		Ctrl+X または Shift+Del
コピー		Ctrl+C
貼り付け		Ctrl+V
削除		Del

切り取り

切り取り コマンド は、選択されたテキストまたはイメージをクリップボードにコピーして、オリジナルの場所にあるオブジェクトを削除します。

コピー

コピー コマンド では、選択されたテキストやアイテムをクリップボードにコピーします。このコマンドを使用することで、XMLSpy 内部でデータの複製を行い、他のアプリケーションへデータを移動することができます。

メモ グリッドビューからテキスト形式の要素をクリップボードにコピーするには XML テキストとしてコピー[1150] と構造化テキストとしてコピー[1150] するという、2つの方法があります。「オプション | ツール」ダイアログの編集タブ[1461]により、コピーコマンドが行われた際に、どちらの処理が行われるかを選択することができます。

貼り付け

貼り付け コマンド により、クリップボードにコピーされたコンテンツを、現在カーソルがある位置に挿入することができます。

削除

削除 コマンド を使うことで、現在選択されているテキスト やアイテムを、クリップボードにコピーすること無く削除することができます。

28.2.3 XML テキストとしてコピー

XML テキストとしてコピーコマンドを使用すると、XML ソース XML テキストを扱うことができる外部アプリケーションに、簡単にデータを受け渡すことができます。グリッドビューにてドキュメントの編集を行なっている場合、表示されている要素を XML テキスト形式でクリップボードにコピーすることができます。

```
<row>
    <para align="left">
        <bold>Check the FAQ</bold>
    </para>
    <para>
        <link mode="internal">
            <link_section>support</link_section>
            <link_subsection>faq30</link_subsection>
            <link_text>XMLSPY 4.0 FAQ</link_text>
        </link>
        <link mode="internal">
            <link_section>support</link_section>
            <link_subsection>faq25</link_subsection>
            <link_text>XMLSPY 3.5 FAQ</link_text>
        </link>
    </para>
</row>
```

XML テキストとしてコピーコマンドでは、現在アクティブな設定を使用して、コピーの対象となるテキストが自動的に整形されます。これらの設定は、オプションダイアログ(「**ツール | オプション**」)のファイルセクションにあるファイルを保存セクションにて変更することができます。グリッドビューからは、**構造化テキストとしてコピー**[1150] コマンドを使用して、テーブルのグリッドビューの表示から XML データをコピーすることができます。

28.2.4 構造化テキストとしてコピー

構造化テキストとしてコピーコマンドは、要素を表示されているままにクリップボードへコピーします。このコマンドは、グリッドビューからテーブル形式のデータをコピーするのに便利な機能となります。

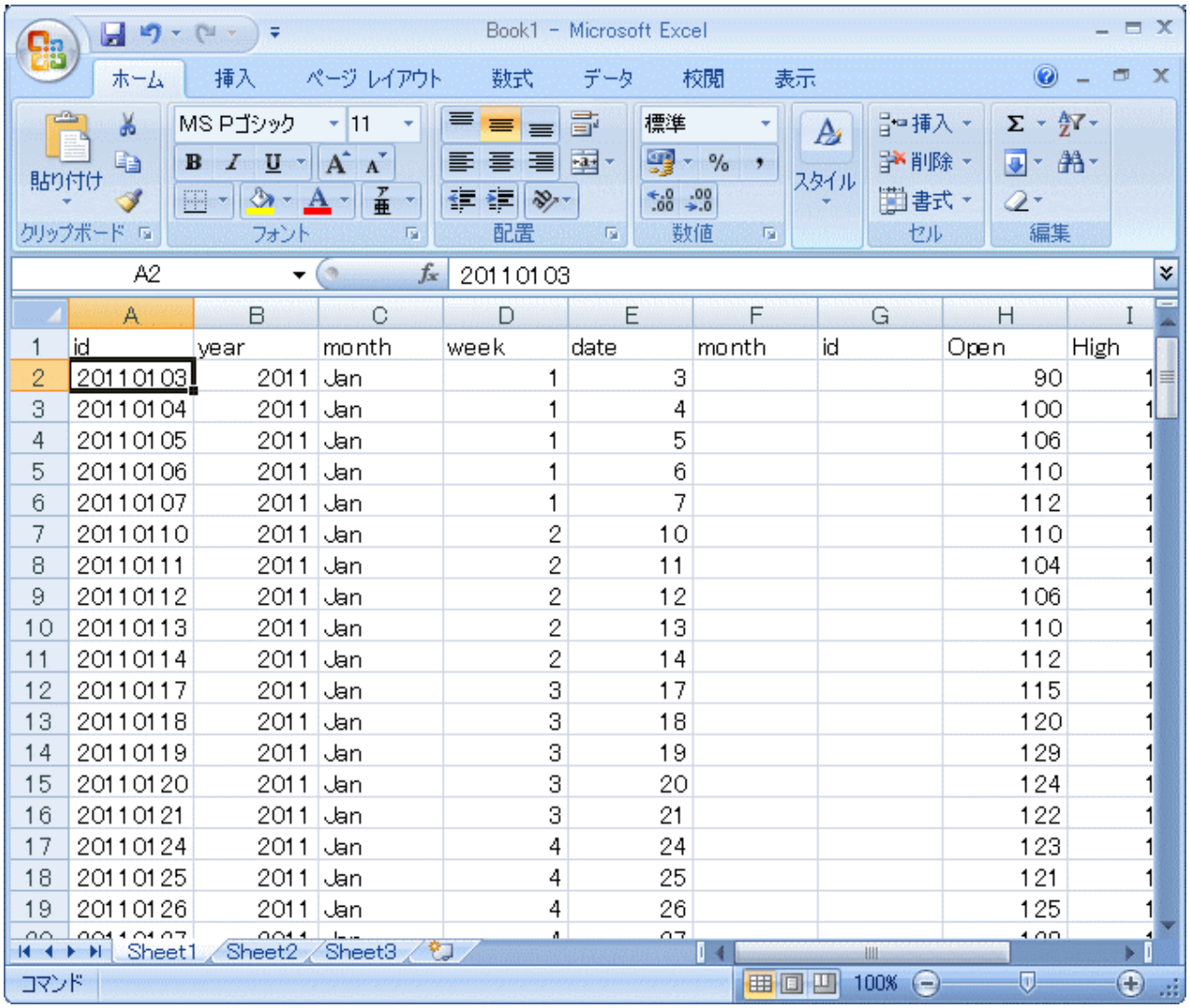
- テーブルビューから個別の行をコピーする[1150]
- テーブルビューからテーブル全体をコピーする[1152]
- 標準グリッドビューからコピーする[1152]

テーブルビューから個別の行をコピーする

下のスクリーンショット内では、グリッドビューのテーブルビュー 内の2 つの行が構成されたテキストとしてコピーされます。

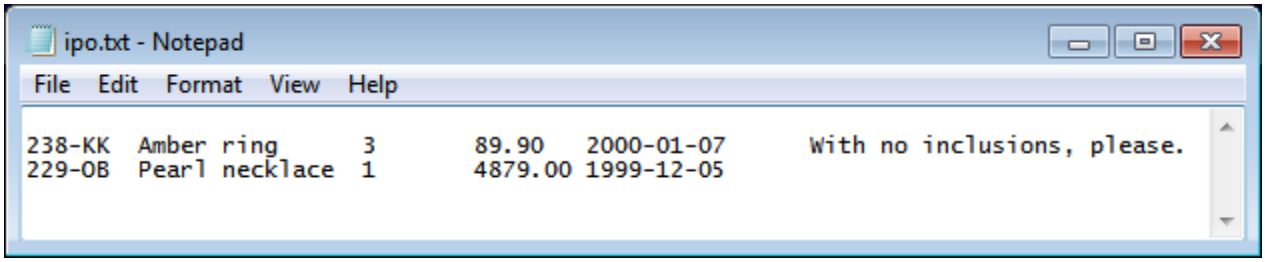
expense-item (3)	type	expto	Date	expense	description
1	Lodging	Sales	2003-01-01	122.11	
2	Lodging	Development	2003-01-02	122.12	Played penny arcade
3	Lodging	Marketing	2003-01-02	299.45	Treated Clients

次の2つの下のスクリーンショットは、グリッドビュー内の構成されたテキストとしてコピーされたデータを表示しています。上のスクリーンショットは、Microsoft Excel ドキュメントとNotepad ドキュメント内に貼り付けられたデータを表示しています。



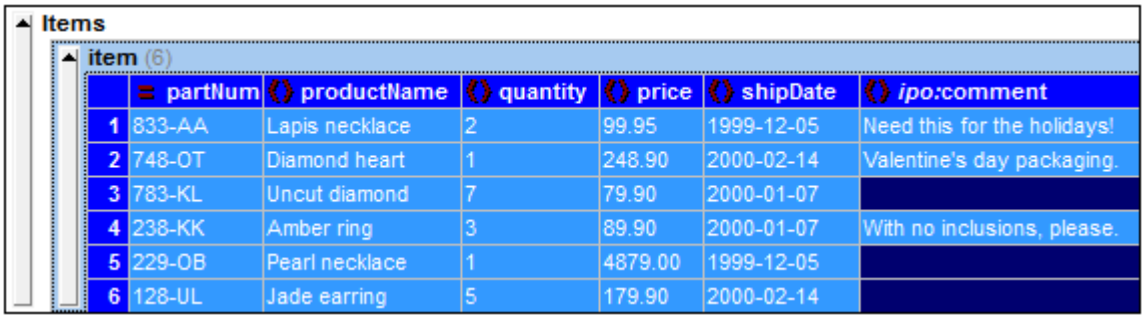
	A	B	C	D	E	F	G	H	I
1	id	year	month	week	date	month	id	Open	High
2	20110103	2011	Jan	1	3			90	1
3	20110104	2011	Jan	1	4			100	1
4	20110105	2011	Jan	1	5			106	1
5	20110106	2011	Jan	1	6			110	1
6	20110107	2011	Jan	1	7			112	1
7	20110110	2011	Jan	2	10			110	1
8	20110111	2011	Jan	2	11			104	1
9	20110112	2011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

Excel は自動的にセルテキストのそれぞれを、テキストの構文フォームをベースにフォーマットしますが、(*上のスクリーンショット*) Notepad はセルテキストを文字列として貼り付けます(*下のスクリーンショット*)。

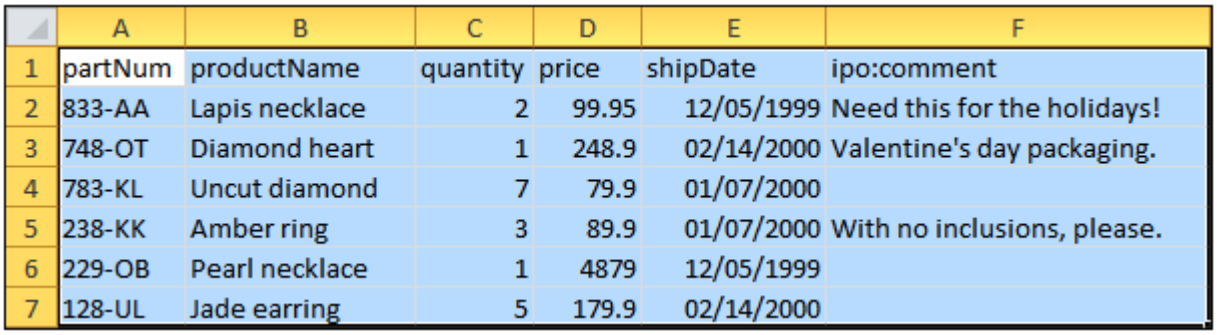


テーブルビューからテーブル全体をコピーする

下のスクリーンショットでは、(item (6) ノードエントリをクリックすることにより)テーブル全体が選択されています。



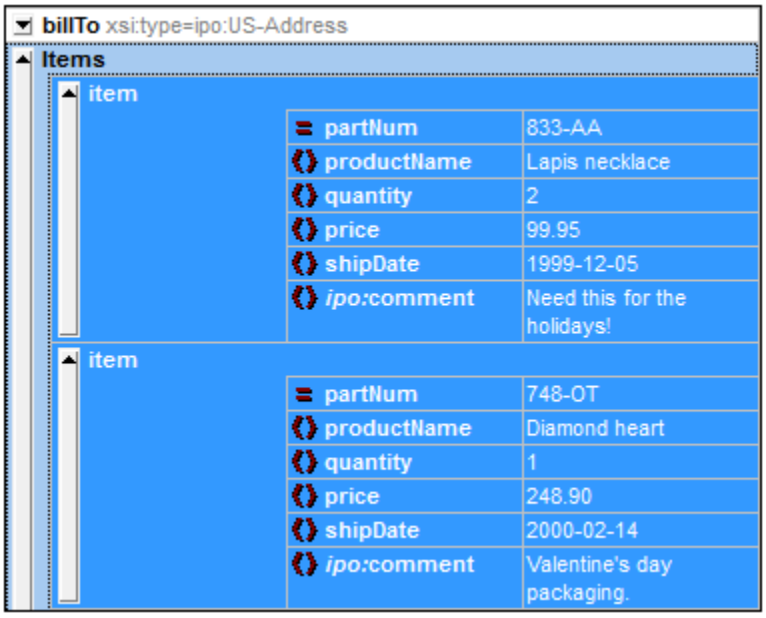
テーブルの列のタイトル以外の個別の行のための結果がコピーされます。下のスクリーンショットは、Microsoft Excel ワークシートにコピーして貼り付けられたデータを表示しています。XML ドキュメント内では、列ヘッダーは、要素または属性名であることに注意してください。



	A	B	C	D	E	F
1	partNum	productName	quantity	price	shipDate	ipo:comment
2	833-AA	Lapis necklace	2	99.95	12/05/1999	Need this for the holidays!
3	748-OT	Diamond heart	1	248.9	02/14/2000	Valentine's day packaging.
4	783-KL	Uncut diamond	7	79.9	01/07/2000	
5	238-KK	Amber ring	3	89.9	01/07/2000	With no inclusions, please.
6	229-OB	Pearl necklace	1	4879	12/05/1999	
7	128-UL	Jade earring	5	179.9	02/14/2000	

標準グリッドビューからコピーする

上に表示されている例はグリッドビューのテーブルビュー(グリッドビュー内でテーブルビューがオンになっている状態)からデータをコピーしています。標準のグリッドビューでは、(テーブルビューがオフに切り替えられている状態)、**構造化テキストとしてコピー** は、グリッドビューで表示されているとおり、選択されたデータをコピーします。しかしながら、構造は、テーブルではありません。例えば、下のスクリーンショットで通常のグリッドビュー内の2つの item 子を持つsアイテムノードを表示しています。



Microsoft Excel 内にコピーされたテキストが貼り付けられると、ワークシートは以下のスクリーンショットのようになります。

	A	B	C	D
1	Items			
2		item		
3			partNum	833-AA
4			productName	Lapis necklace
5			quantity	2
6			price	99.95
7			shipDate	12/05/1999
8			ipo:comment	Need this for the holidays!
9		item		
10			partNum	748-OT
11			productName	Diamond heart
12			quantity	1
13			price	248.9
14			shipDate	02/14/2000
15			ipo:comment	Valentine's day packaging.

XML データを構造化されたテキストとしてコピーするほか、構造化テキストとしてコピーコマンドを使用しても、データを XML テキストとしてグリッドビューからコピーすることができます[1150]。

28.2.5 XPath をコピー

XPath をコピーコマンドはテキストビューならびにグリッドビューで使用可能で、現在選択されているノードから XPath 条件式を作成し、クリップボードにコピーします。この機能により、条件式をドキュメント（例えばXSLTドキュメント）に貼り付けることができます。全ての条件式は、ドキュメントルートから開始されます。は、特定の要素ではなく、その階層レベルにある全ての要素が選択されます。テキストビューの場合、その単一要素が選択されます。例えば、2番目のCompany 要素から見て3番目のPerson という要素以下にあるLastName という名前の要素を選択した場合、作成されるXPath 条件式は以下のようになります
: `/Companies/Company[2]/Person[3]/LastName`

メモ: グリッドビューでは、コンテキストメニューからも XPath をコピーコマンドへアクセスすることができます。

28.2.6 XPointer をコピー

XPointer をコピーコマンドは、テキストビューならびにグリッドビューで使用可能で、現在選択されているノードのelement() スキーマ XPointer を作成し、クリップボードにコピーします。この機能により、XPointer をドキュメント（例えば、XML ドキュメント内部にある XInclude 要素内のxpointer 属性）に貼り付けることができます。

XPointer のelement() スキーマからは、element(/1/3) という形式が返され、この例ではドキュメント要素（ルート要素）にある3番目の要素が選択されます。以下の点に注意してください:

- element() スキーマでは属性を表現することができません。XMLSpy で属性が選択されている場合、以下のことが起こります：グリッドビューでは、XPointer をコピーコマンドが無効になります。テキストビューでは、その属性の親となる要素を元に XPointer が生成されます。
- 複数の要素を選択することはできません。グリッドビューで複数の要素が選択されている場合、XPointer をコピーコマンドは無効になります。テキストビューでは、選択された複数要素の親となる要素を元に XPointer が生成されます。

メモ: グリッドビューでは、コンテキストメニューからも XPointer をコピーコマンドにアクセスすることができます。

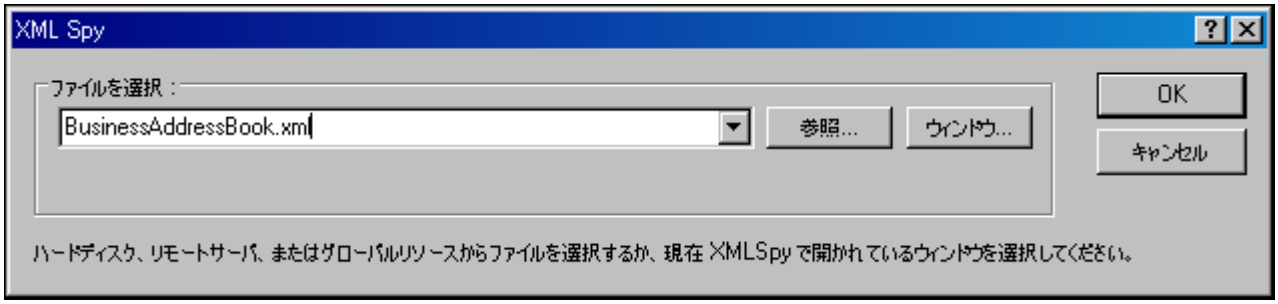
28.2.7 挿入

挿入コマンドを選択、またはマウスオーバーすると、以下に説明されるような3つのコマンドが表示されます:

- ファイルパスの挿入 1154
- XInclude の挿入 1155
- エンコードされた外部ファイルの挿入 1157

ファイルパスの挿入

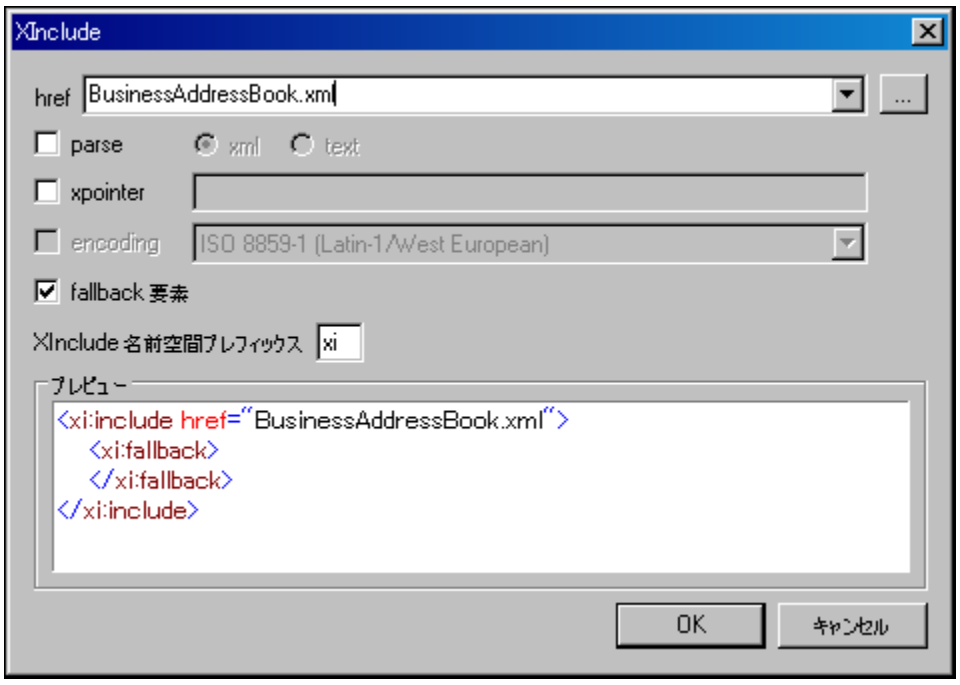
ファイルパスコマンドは、テキストビューまたはグリッドビューで表示されている、あらゆる種類のドキュメントで使用することができます。この機能を使用することで、カーソルがある位置にファイルのパスを挿入することができます。このコマンドをクリックすると、以下のスクリーンショットに示されるダイアログが表示され、目的のファイルを選択することができます。



目的のファイルは、以下にある方法から選択することができます: (i) ファイル、URL、またはグローバルリソースを参照(参照 1132 ボタンを使用)、(ii) ウィンドウから既に開かれているファイルを選択(ウィンドウボタンを使用)。選択をが終わると、「OK」をクリックします。選択されたファイルのパスが、アクティブなドキュメントのカーソル位置に挿入されます。

XInclude の挿入

XInclude コマンドは、テキストビューまたはグリッドビューで使用することができ、新たな XInclude 要素を、テキストビューにてカーソルが選択している箇所、またはテキストビューやグリッドビューにて選択されているアイテムの直前に挿入することができます。グリッドビューにて選択されているのが属性の場合、XInclude 要素は、属性の後か、属性の親要素が持っている最初の子要素の前に挿入されます。このコマンドを選択すると、XInclude ダイアログが表示されます(以下のスクリーンショットを参照)。



インクルードされる XML ファイルを href テキストボックスに入力します(または、テキストボックスの右側にある ... (参照) ボタンをクリックすることで、ファイルの参照を行うこともできます)。ファイル名が href 属性の値として、XML ドキュメント内に入力されます。XInclude (xi:include) 要素の parse、xpointer や encoding 属性、そして xi:include の fallback 子要素も、このダイアログにて挿入することができます。対応するチェックボックスにチェックを入れ、必要な値を選択/入力してください。fallback 要素の場合は、チェックボックスにチェックを入れることで、空要素を挿入することができます。fallback 要素のコンテンツは、編集ビューにて追加する必要があります。

parse 属性により、インクルードされるドキュメントが XML またはテキストとして解析(パース)されるかが決定されます(デフォルトでは XML

という値がセットされており、指定を行う必要はありません)。xpointer 属性は、href 属性により指定されるドキュメント内フラグメント(つまり、インクルードされるフラグメント)を特定します。encoding 属性ではインクルードされるドキュメントのエンコーディングが指定され、XMLSpy により、インクルードするドキュメントの内容がインクルード元のドキュメントエンコーディングに合う形で変換されます。インクルードされるドキュメントが見つからない場合は、xi:include 要素がfallback 子要素により置換えられます。

XInclude により2つのXMLドキュメントをインクルードするXMLドキュメントのサンプルを以下に示します:

```
<?xml version="1.0" encoding="UTF-16"?>
<AddressBook xsi:schemaLocation="http://www.altova.com/sv/myaddresses AddressBook.xsd"
       xmlns="http://www.altova.com/stylevision/tutorials/myaddresses"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="BusinessAddressBook.xml"/>
       <xi:include href="PersonalAddressBook.xml"/>
</AddressBook>
```

XMLドキュメントがパースされると、上の例で2つあるXInclude 要素が、href 属性により指定されたファイルの内容に置換えられます。

xml:base

XMLSpy のXML バリデーターによりXMLドキュメントが読み取られ、XInclude 名前空間内のinclude 要素(これ以降はxi:include と表記)が発見されると、この要素(xi:include)を、xi:include 要素のhref 属性により指定されたXMLドキュメントに置換えます。インクルードされたXMLドキュメントのドキュメント要素(ルート要素)や、XPointer により指定された要素にもxml:base 属性は含まれ、インクルードされたベースURI を保持します。(インクルードされたXML やソーフラグメントを含む)結果 XMLドキュメントが、スキーマに従う形で妥当でなければならない場合、インクルードされたドキュメントのドキュメント要素(ルート要素)やソーフラグメントのトップレベル要素を、xml:base 属性を許可するコンテンツモデルにより作成する必要があります。スキーマにより、xml:base 属性がこの要素に対して認められない場合、ドキュメントは妥当でないと評価されます。XMLSpy のスキーマビューにて要素コンテンツモデルのxml:base 属性を定義する方法は、ドキュメンテーションのスキーマビューセクション、xml: プレフィックス属性 228 にて記述されます。

XPointers

XMLSpy ではXInclude 内におけるXPointer がサポートされます。関連するW 3C 勧告は、XPointer Framework ならびにXPointer element() スキーマの2つです。XInclude 要素内のXPointer により、XMLドキュメント全体の代わりに、XMLドキュメント内部でインクルードする箇所を特定することができます。XPointer は、以下にあるように、XInclude 要素内部で使用されます::

```
<xi:include href="PersonalAddressBook.xml" xpointer="element(usa)"/>
<xi:include href="BusinessAddressBook.xml" xpointer="element(/1/1)"/>
<xi:include href="BobsAddressBook.xml" xpointer="element(usa/3/1)"/>
<xi:include href="PatsAddressBook.xml" xpointer="element(usa)element(/1/1)"/>
```

XPointer のelement() スキーマでは、NCName や子シーケンスを使用することができます。

- 上の例にある最初のxi:include 要素のxpointer 属性ではusa というNCName を伴った要素スキーマが使用されています。XPointer Framework に従うと、このNCName によりusa というID を持つ要素が特定されます。
- 2つ目のxi:include では、element(/1/1) という値がxpointer 属性で使用されています。最初のステップでは、ドキュメントにある最初の子要素(つまりドキュメントが整形式であれば、ドキュメント要素)が特定されます。次のステップでは、最初のステップで特定された要素から見た最初の子要素が特定され、結果としてドキュメント要素にある最初の子要素がインクルードされます。
- 3番目のxi:include にあるxpointer 属性では、NCName と子シーケンスの両方が使用されています。XPointer により、usa というID を持つ要素の3番目の子要素、その子要素から見た最初の子要素がインクルードされます。.
- 最初に使用したXPointer が期待通りに動作するか分からない場合、4番目の例として示されているxpointer="element(usa)element(/1/1)" という書式を使用することもできます。この方法では、たとえusa というID を持つ要素が存在しない場合でも、ドキュメント要素の最初の子要素が選択されます。これ以外にもバックアップとなる式を追加することができます。各 XPointer は、空白スペース以外で分けることはできず、以下のような書式に従う必要があります: xpointer="element(usa)element(addresses/1)

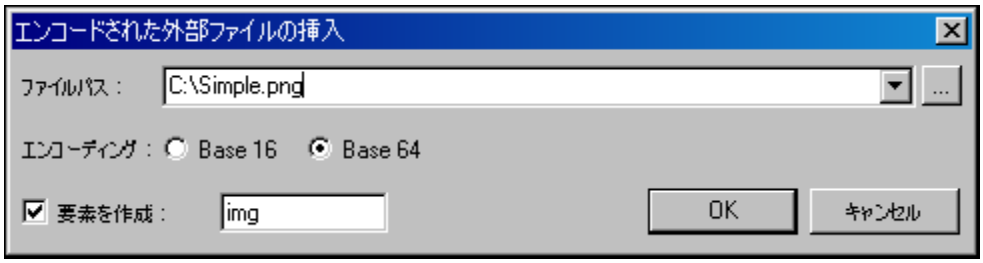
element(/1/1)"。

メモ: element() スキーマはqualified name でサポートされておらず、名前空間バインディングコンテキストは、element() スキーマにて使用されません。

エンコードされた外部ファイルの挿入

エンコードされた外部ファイルコマンドは、テキストビューならびにグリッドビューで使用することができます。このコマンドにより、Base-16 またはBase-64 テキスト形式でエンコードされた外部ファイルを、XML ドキュメント内部の任意の箇所に追加することができます。この機能により外部ファイルをXML ドキュメント内に埋め込むことができます。

「**挿入 | エンコードされた外部ファイル**」コマンドを選択すると、エンコードされた外部ファイルの挿入を行うためのダイアログが表示されます（以下のスクリーンショットを参照）。



埋込みを行う外部ファイルを参照または直接入力します。Base-16 またはBase-64 のエンコーディングを選択してください。エンコードされたテキストを、要素の中に含めて表示する場合、要素を作成チェックボックスにチェックを入れて、作成する要素名をテキストボックスに入力します。要素を作成チェックボックスにチェックが入っていない場合、エンコードされたテキストは、カーソルがある位置に直接挿入されます。

「OK」をクリックすると、選択したファイルをエンコードしたテキストが、（もしそう指定されていれば）要素に含まれる形でカーソル位置に挿入されます。

```
<img ext="png" encoding="xs:base64Binary">
  iVBORw0KGgoAAAANSUhEUgAAABAAAAAQAQMAAAAlPW0iAAAABlBMVEUAAAD/
  //+l2Z/dAAAAM0lEQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83NDDeNGe4U
  g9C9zwz3gVLMDA/A6P9/AFGGFyjOXZtQAAAAAElFTkSuQmCC
</img>
```

上の表示は、PNG イメージファイルをテキストにエンコードしたものです。img 要素がエンコードされたテキストの回りに作成されます。

28.2.8 整形

アイコン

アイコン:	

説明

整形コマンドにより、テキストビューに表示されているXML ドキュメントを再フォーマットすることができます。「**ツール | オプション**」にて表示される[オプションダイアログのビュータブ](#)1466 で、*インデントを使用* にチェックが入っているかどうかにより、2つの書式オプションを利用することができます:

- *インデントを使用* がチェックされている：構造が表示されるような形式でドキュメントが再フォーマットされ、各階層レベルがインデントされます。テキストビュー設定 ダイアログ[1363] (**ビュー | テキストビュー設定**)によりドキュメントの構造を明確に表示することができます。
- *インデントを使用* がチェックされていない：各行が左寄せになるように再フォーマットされます。

整形を使用する方法

インデントを使用して、XMLドキュメントにおける構造を表示するには、以下の操作を行います:

1. オプションダイアログのビュータブ[1466] (**ツール | オプション**) にて、*インデントを使用* チェックボックスにチェックを入れます。
2. テキストビュー設定 ダイアログ[1363] (**ビュー | テキストビュー設定**) から、整形におけるインデントにて使用されるタブの大きさを指定します。
3. オプションダイアログのファイルタブ[1455] (**ツール | オプション**) にて、フォーマット（インデント）を行いたくない要素を入力します。
4. **整形** コマンド（このコマンド）をクリックします。

全ての行が左寄せされた状態にドキュメントを再フォーマットするには、*インデントを使用* チェックボックスのチェックを外してください。

以下の点に注意してください

- このコマンドを使用するには、XMLドキュメントが整形式である必要があります。
- 整形により、ドキュメントが保存される際に、空白やタブが追加されます。
- 整形式が有効になっており（「ツール | オプション | ビュー | インデントを使用」）、テキストビューからグリッドビューへの切り替え、またはその逆が行われると、ドキュメントは自動的に整形されます。XML テキストの整形コマンドを選択する必要はありません。
- 空白の削除[1158] コマンドを使用することで、整形により作成された全ての空白（改行とインデント）を削除することができます。

28.2.9 空白の削除

アイコン

アイコン:	

説明

空白の削除 コマンドにより、ドキュメント上にある全ての空白が削除されます。この機能によりファイルの大きさを小さくすることができます。整形[1157] コマンド[1157] コマンドを使用することで、ドキュメントに改行やインデントを追加することができます。

28.2.10 全て選択

全て選択 (**Ctrl+A**) コマンドにより、ドキュメント全体のコンテンツが選択されます。

28.2.11 検索、次を検索

アイコンとショートカット

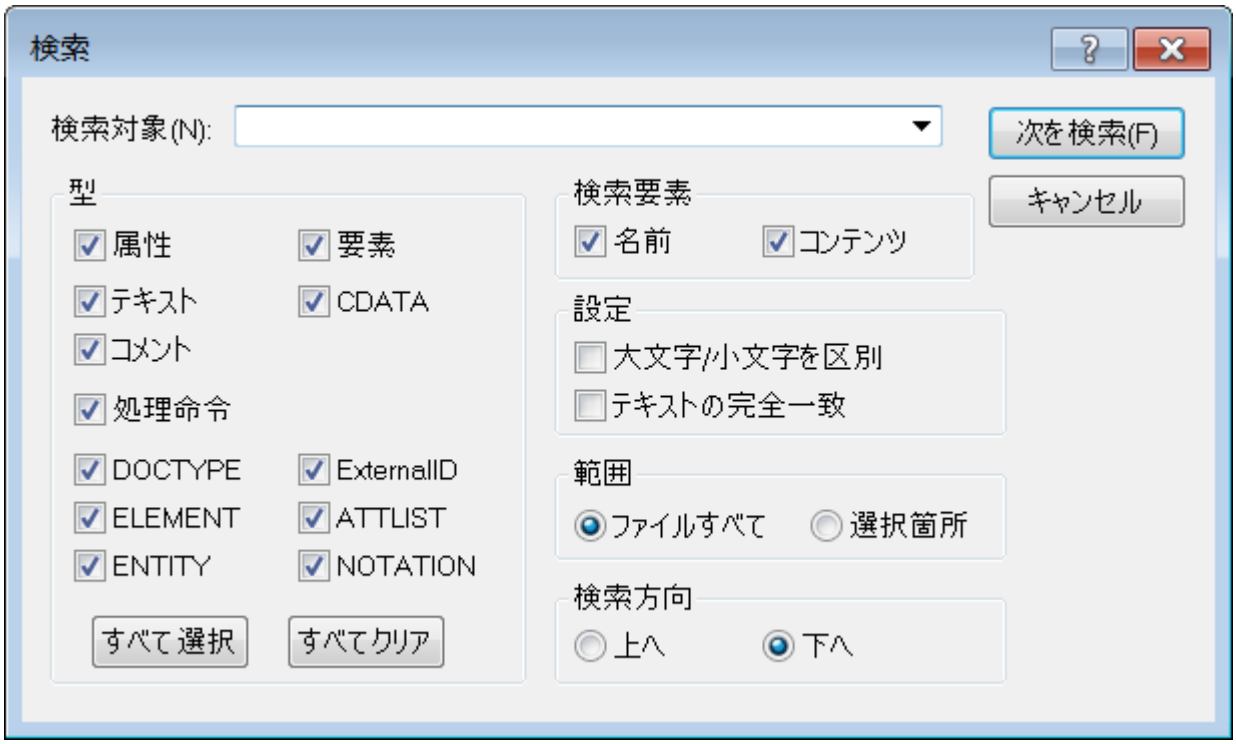
コマンド	アイコン	ショートカット
検索		Ctrl+F
次を検索		F3

検索

検索 コマンド により、検索ダイアログが表示され、検索を行ないたい文字列の入力や、検索に伴う各種オプションを指定することができます。使用しているビューに従って、検索ダイアログには異なるオプションが表示されます。テキストの検索を行うには、検索対象テキストボックスにその文字列を入力するか、コンボボックスを使用して、最後に使用された10個ある検索文字列から選択を行い、検索に必要なオプションを指定します。

検索 と **次を検索** コマンドをプロジェクトウィンドウ内でプロジェクトが選択されるとファイルとフォルダーを検索するために使用することができます。

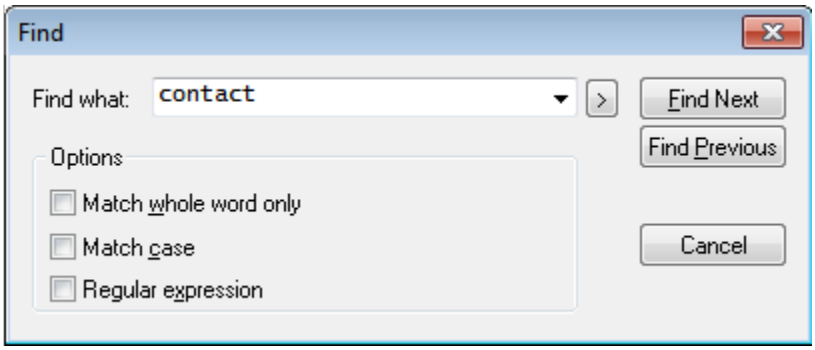
グリッドビュー
グリッドビューでは、次のダイアログボックスが表示されます。必要なオプションを選択、または、ラジオボタンを選択します。使用することのできるオプションについては下で説明されています。



- 型ペインでは、検索対象に含めるXMLドキュメントのノードやコンポーネントを指定することができます。この機能により、特定の型のノードを検索から除外することができます。「全て選択」ボタンにより、ペイン内にある「全てのチェック」ボックスが有効になり、「全てクリア」ボタンにより、全てのチェックボックスが無効になります。

- 検索対象ペインでは、入力された検索文字列に対して、ノード名、ノードのコンテンツ、またはそれら両方を検索対象に含めるか指定することができます。
- 設定ペインでは、検索において大文字と小文字の区別を行うか、または検索対象が入力テキストに完全に一致するかを指定することができます。
- 範囲ペインでは検索の範囲を指定することができます(ファイル全体、または、選択されたテキスト)。
- 検索方向オプションでは、検索が行われる向きを指定することができます。

テキストビュー
編集することのできるテキストビューの検索機能に関しては、以下のセクションを参照してください: テキストビュー[140]。(XPath/XQuery ウィンドウ[113] などの) 編集することのできないテキストビューの検索機能については下で説明されています。

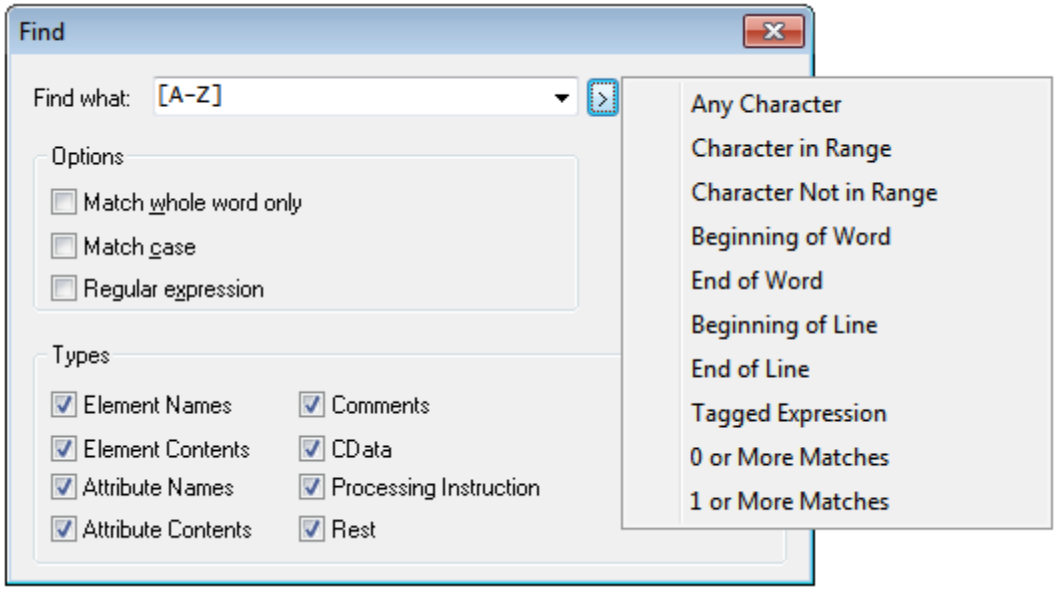


- 単語の完全マッチ: テキスト内にある正確な単語だけが検索のマッチ対象となります。例えば、単語の完全マッチオプションを有効にして、fit という単語で検索を行うと、fit という文字列だけが検索結果として表示され、fitness という単語はマッチしません。
- 大文字・小文字を区別: 大文字と小文字を区別した検索を行います(Address と address は別物として扱われます)。
- 正規表現: テキストボックスに入力された文字列を正規表現として、検索を行います。正規表現については、正規表現[1160] を参照ください。

以下の点に注意してください:

- 検索ダイアログはモードレスになっており、テキストビューを使用中も、ダイアログを開いたままにしておくことができます。ダイアログボックスが開いている間、Enter を押すと、ダイアログボックスが閉じられます。ダイアログボックスを開く前にテキストがマークされると、マークされたテキストは自動的にテキストの検索ボックスに挿入されます。
- 検索ダイアログが閉じられた後、F3 を押下することで現在の検索を継続して行うことができます。Shift+F3 により、反対方向へ検索が行われます。
- 検索対象コンボボックスの右側にある「展開」ボタンをクリックすると正規表現[1160] を使用するためのウィンドウが表示されます。

正規表現
正規表現を使用することで、検索の条件をさらに絞り込むことができます。表示されるポップアップリストから正規表現を構築することができます。リストへのアクセスを行うには、検索文字列を入力するフィールドの右にある > ボタンをクリックします。



表示されているアイテムをクリックすることで、それに対応した正規表現が入力フィールドに入力されます。正規表現の構文文字とその説明を以下に示します。

.	全ての文字列にマッチします。これは1つの文字に対応したプレースホルダーです。
\(	マッチのタグ付けを行うための開始マーク。.
\)	タグ付けの終了マーク
\n	ここでのn は1 から9 までの数値を表わしており、上でタグ付けされた範囲を置換えにて使用することができます。例えば、検索文字列がFred¥([1-9])XXX で、置換え文字列がSam¥2YYY である場合、Fred2XXX という文字列がSam2YYY という文字列に置換えられます。
\<	単語の開始位置にマッチします。
\>	単語の終了位置にマッチします。
\x	通常正規表現にて特別な意味を持つx という文字を、通常の文字として扱います。例えば、¥[という文字列は、文字セットの開始ではなく[として認識されます。
[...]	文字のセットを表します。例えば、[abc] という表現により、a、b、c 全ての文字がマッチするようになります。それ以外にも[a-z] という形で範囲を指定することにより、この例では小文字全てをマッチの対象とすることができます。
[^...]	指定された文字セット以外のものにマッチします。例えば、[^A-Za-z] という表現では、アルファベット以外の文字列がマッチの対象になります。
^	行頭を表します(上で述べられている文字のセットで使用される場合を除く)。
$	行末を表します。例えばA+$ という表現で、行末に1つ以上連続して並んでいるA を検索します。
*	0 回以上の繰り返しにマッチします。例えば、Sa*m という表現は、Sm、Sam、Saam、Saaam というた文字列にマッチします。
+	例えば、Sa+m という表現は、Sam、Saam、Saaam というた文字列にマッチします。

メモ 置換 フィールド[1162]にて正規表現はサポートされていません。

次を検索

次を検索 コマンドは、最後に実行された検索コマンドを再度実行し、目的のテキスト内で次に出現する検索結果を表示します。

28.2.12 置換

アイコンとショートカット

コマンド	*アイコン*	*ショートカット*
置換		Ctrl+H

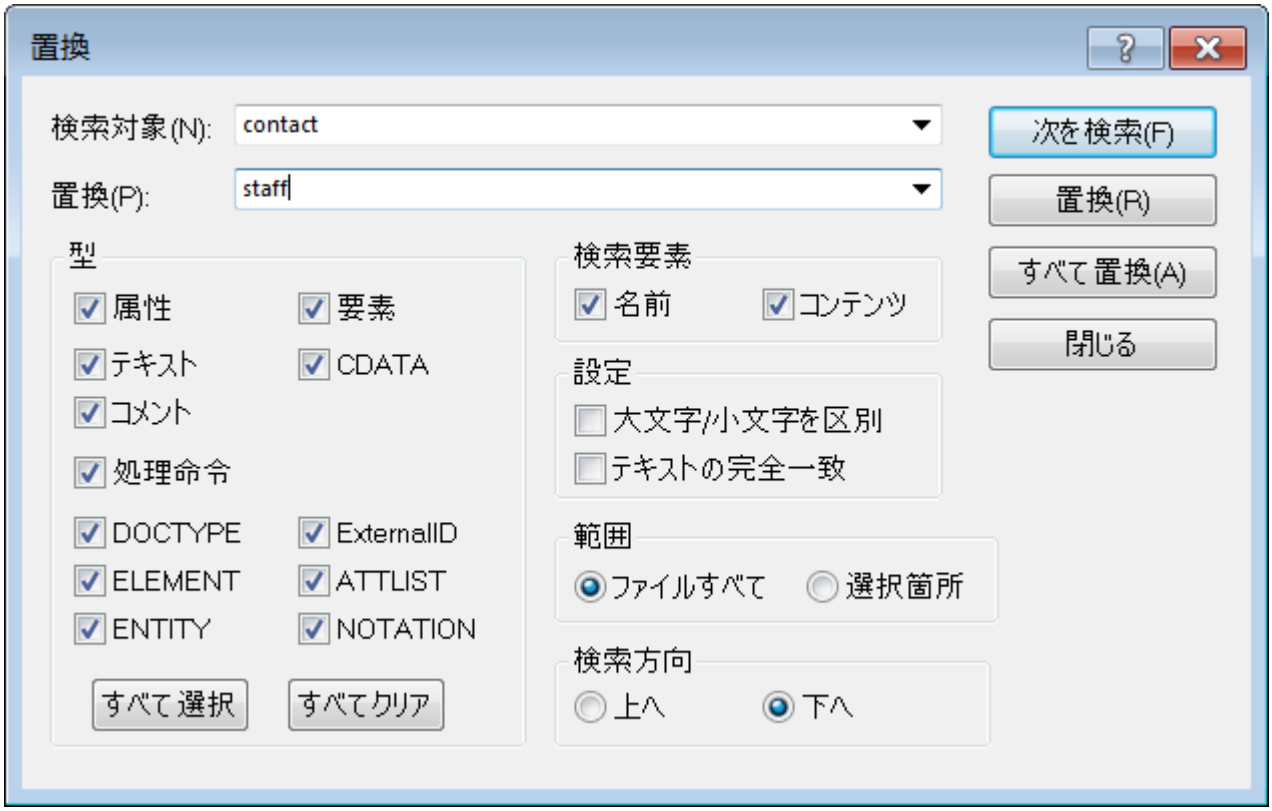
説明

置換コマンドでは、テキスト内のある文字列を別の文字列に置き換えることができます。検索[1159] コマンドにて使用されるオプションをここでも使用することができます。使用しているビューの種類により、置換ダイアログに表示されるオプションも異なるものになります。各アイテムを個々に置き換えるか、**全て置換**ボタンを使用することで、検索と置換処理をグローバルに行うことができます。

グリッドビュー

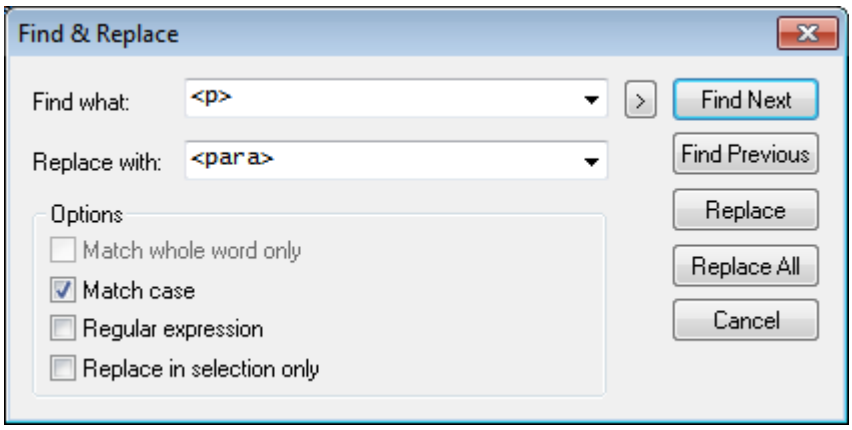
グリッドビューでは、**置換** コマンドを選択すると、置換えダイアログが開かれます（*下のスクリーンショット*）。スクリーンショットは、検索[1159] コマンドで説明されている異なる検索オプションを表示しています。

(C) 2015-2021 Altova GmbH

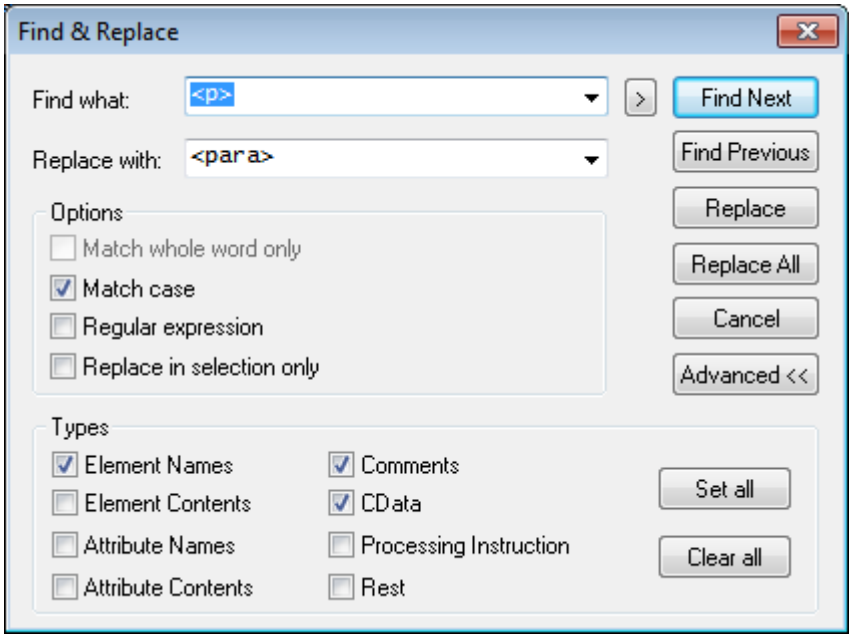


テキストビュー

編集することのできるテキストビューの置換の機能に関しては、以下のセクションを参照してください: テキストビュー[140]。(XPath/XQuery ウィンドウ[113] などの) 編集することのできないテキストビューの置換機能は下で説明されています。テキストビューでは、置換の選択は、を開きます。コマンド検索と置換ダイアログ (*下のスクリーンショット*)。検索オプションは、検索[1159] コマンドの検索のオプションと同じです。選択された範囲内での置換オプションは選択範囲内でのみ検索と置換をおこないます。



詳細ボタンをクリックすることで、ノードの型の検索のスコープを選択することのできる、型のオプションを指定することができます(次を参照してください: 検索[1159] コマンド)。

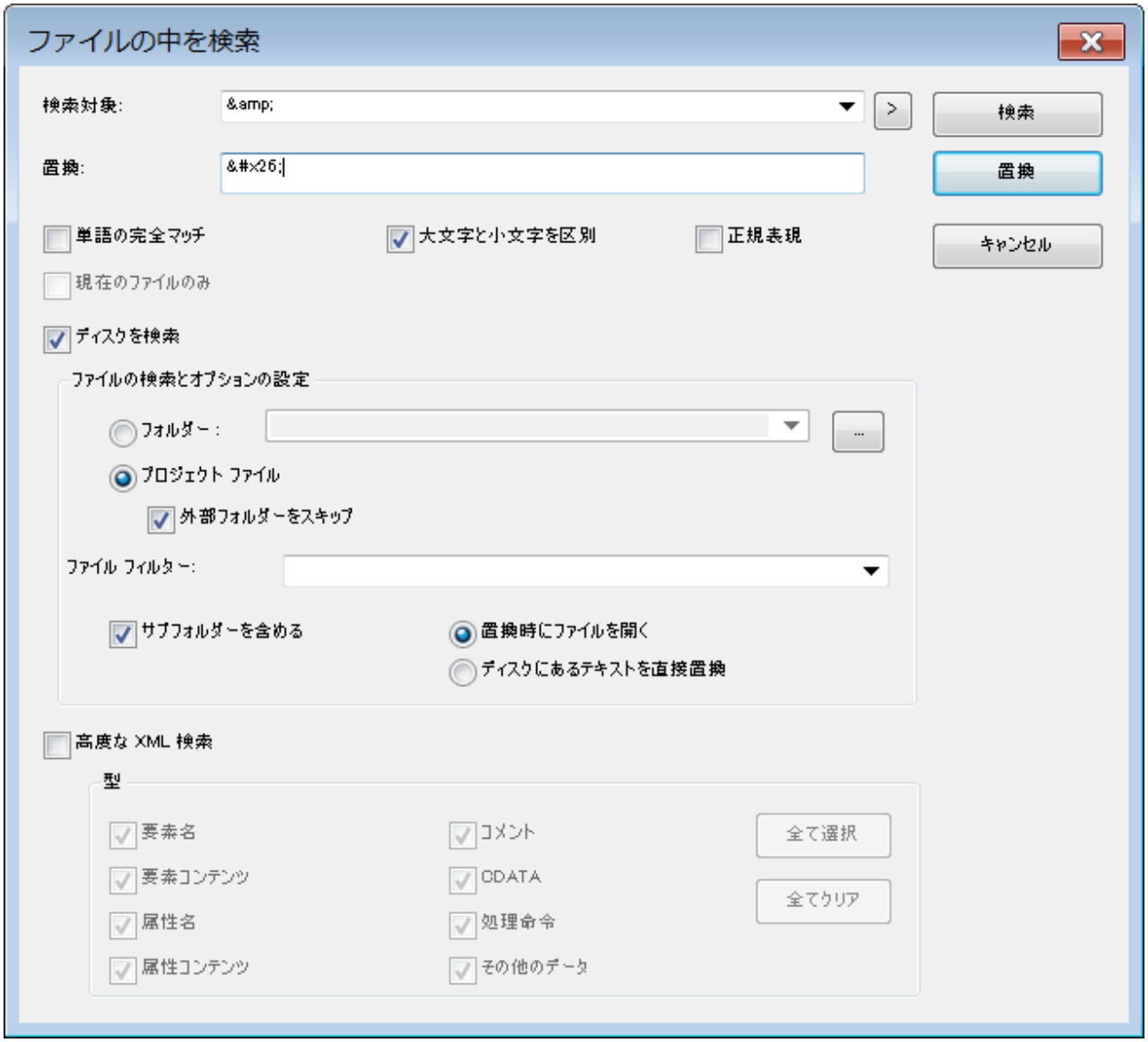


メモ **全て置換**コマンドを使用すると、各置換は、単一の**オペレーション**として認識されます。従って、**全て置換**処理の取り消しは、1ステップずつ行うことになります。

28.2.13 ファイル内の検索

ファイルの中を検索コマンドは、数多くのファイルに対して検索と置換を素早く行うことができるパワフルな機能です。コマンドをクリックすることで、ファイルの中を検索ダイアログが表示されます(以下の*スクリーンショット*を参照)。**ファイルの中を検索**コマンドは、指定された位置全ての検索ならびに置換処理を一度に行うという点で、**検索**コマンドとは違っています。検索結果のレポートが[ファイル内の検索 出力 ウィンドウ](#)[115] に表示されます。しかし**検索**コマンドの場合は、ユーザーが検索文字列を入力し、検索により見つかったアイテムに対して個々に処理を行うことになります。

(C) 2015-2021 Altova GmbH



検索条件

検索には大きく分けて、(i) 何を検索するのか、そして(ii) どこを検索するのか、という2つの条件があります。検索の対象となるテキストの入力方法については、検索[1159] コマンドの記述を参照ください。検索対象テキストボックスにて入力されたテキストが正規表現の場合、正規表現チェックボックスにチェックを入れる必要があります。正規表現に対する入力ヘルパーは、(> ボタン) ボタンからアクセスすることができます。正規表現を使った検索については検索[1160] コマンドにある記述を参照ください。

XML ドキュメント内にある検索対象のノードを指定するには、高度な XML 検索チェックボックスにチェックを入れて、目的のノード型のチェックボックスにチェックを入れてください。

現在のファイルのみ チェックボックス、または *ディスクを検索* チェックボックスにチェックを入れることで、どのファイルを検索対象にするか指定することができます。ディスクの検索を選択した場合、検索を行うフォルダーや プロジェクト[1168] を(ディスクを検索チェックボックスにチェックを入れた後に)選択することができます。プロジェクトフォルダーが選択されている場合、プロジェクトに追加されている外部フォルダーを検索の対象から

除外することができます。検索の対象となるファイルは、(xml* や xsl* といったかたちで) 拡張子とアスタリスクによりフィルタリングすることができます。2つ以上あるファイル拡張子は、コンマやセミコロンで分けることができます(例: xml*;xsl*)。アスタリスクはワイルドカードとして使用することができます。

検索対象に指定された箇所にある検索文字列のインスタンスは、ファイル内の検索 出力バー[115] に表示されます。表示されているアイテムをクリックすることで、そのファイルがテキストビューで表示され、アイテムがハイライトされます。

置換

置換ボタンをクリックすると、全ての検索結果が置換文字列に置き換えられることに注意してください。ファイル内の検索ダイアログにて、*置換時にファイルを開く* が選択されている場合、ファイルがテキストビューで開かれ、*ディスクにあるテキストを直接置換* が選択されている場合、ファイルを開くこと無しに置換が行われます。置換が行われた文字列はファイル内の検索 出力バー[115] に表示されます。表示されているアイテムをクリックすることで、そのファイルがテキストビューで表示され、アイテムがハイライトされます。

メモ: 正規表現は置換フィールドにてサポートされません。

28.2.14 ブックマーク コマンド

アイコンとショートカット

コマンド	*アイコン*	*ショートカット*
ブックマークを挿入/削除する		**Ctrl+F2**
全てのブックマークを削除する		**Ctrl+Shift+F2**
次のブックマークへ移動する		**F2**
前のブックマークへ移動する		**Shift+F2**

ブックマークを挿入/削除する

ブックマークを挿入/削除する コマンドにより、現在のカーソル位置にブックマークを挿入したり、カーソルのある行に既にブックマークが割り当てられている場合にはブックマークを削除することができます。このコマンドはテキストビューでのみ使用することができます。

ブックマークのラインは、以下の方法の内のひとつにより表示されます:

- ブックマーク余白 が有効化されている場合、テキスト左端にあるブックマーク余白に青色のマークが表示されます。
- ブックマーク余白が有効化されていない場合、カーソルが位置している行全体がハイライトされます。

F2 キーにより、ドキュメント内にあるブックマークへ移動することができます。

全てのブックマークを削除する

全てのブックマークを削除する コマンドにより、現在定義されている全てのブックマークを削除することができます。このコマンドはテキストビューでのみ使用することができます。**取り消し**コマンドを使っても、このコマンドを取り消すことができないことに注意してください。

次のブックマークへ移動する

次のブックマークへ移動する コマンドを使用すると、テキストカーソルを次のブックマーク行の行頭へ移動することができます。このコマンドはテキストビューでのみ使用することができます。

前のブックマークへ移動する

前のブックマークへ移動する コマンドを使用すると、テキストカーソルを前のブックマーク行の行頭へ移動することができます。このコマンドはテキストビューでのみ使用することができます。

28.2.15 コメントの IN/OUT

コメントの IN/OUT コマンドはテキストビューで使用することができ、XML テキストフラグメントのコメントアウトやアンコメントを行うことができます。XML ドキュメント内のテキストは、XML のコメント開始ならびにコメント終了タグ（それぞれ <!-- と --> ）によりコメントとして認識されます。XMLSpy では、**コメントの IN/OUT** メニューコマンドを使用することで、これらコメントのタグを簡単に挿入することができます。

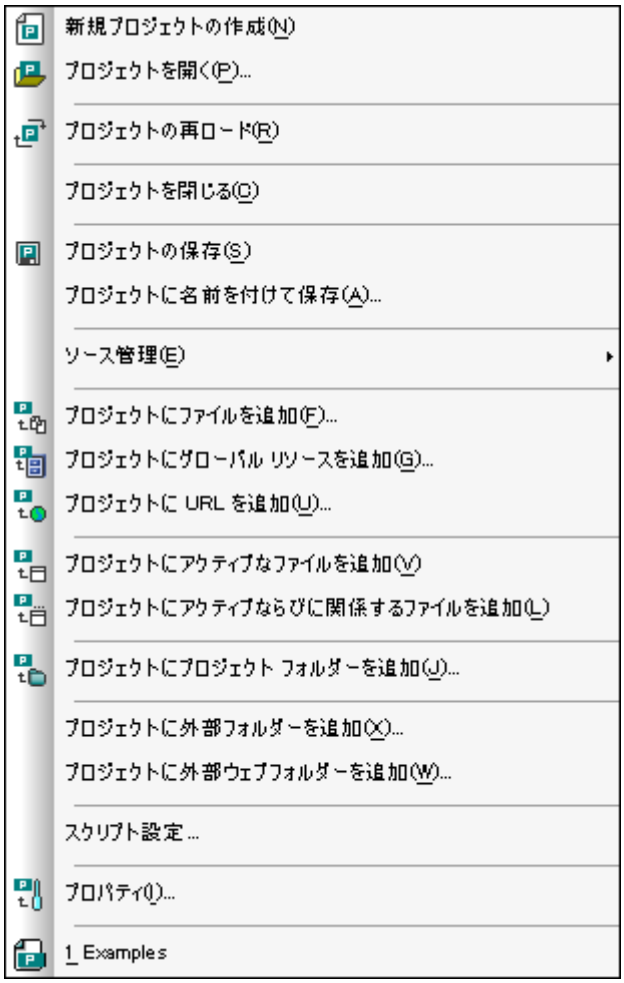
テキストブロックをコメントアウトするには、コメントアウトするテキストを選択し、**編集**メニュー、または選択されたテキストを右クリックすることで表示されるコンテキストメニューから**コメントの IN/OUT** コマンドを選択します。コメントアウトされたテキストはグレーアウトされます（以下のスクリーンショットを参照）。

```
<Department>
    <Name>Administration</Name>
    <Person>
    <Person>
    <Person>
    <!--<Person>
        <First
        <Last></Last>
        <PhoneExt></PhoneExt>
        <EMail></EMail>
        <LeaveTotal></LeaveTotal>
        <LeaveUsed></LeaveUsed>
        <LeaveLeft></LeaveLeft>
    </Person>-->
</Department>
```

コメントされたテキストのブロックをアンコメントするには、**コメントタグを除いて**、コメントアウトされたブロックを選択し、**編集**メニュー、または選択されたテキストを右クリックすることで表示されるコンテキストメニューから、**コメントの IN/OUT** コマンドを選択します。コメントタグが削除され、グレーアウトされていたテキストが通常の色に戻ります。

28.3 プロジェクトメニュー

XMLSpy では、広く使用されているツリービューにより表示されるXML プロジェクトを使用することで、複数のファイルやURL を管理することができます。ファイル[1185] とURL[1185] は拡張子などでフォルダー[1186] 内にグループ化することができ、バッチ処理などを簡単に行うことができるようになります。



メモ: プロジェクトに関連したコマンドは、プロジェクトウィンドウにあるアイテムを右クリックすることで表示されるコンテキストメニューからアクセスすることができます。

絶対パスと相対パス

各プロジェクトは、.spp という拡張子のついたプロジェクトファイルとして保存されます。これらのファイルの内容はXML ドキュメント形式で保存されており、通常のXML ファイルとしても編集することができます。プロジェクトファイルでは、同じ位置より先高い場所に対しては絶対パスが、現在のフォルダーやサブフォルダーに対しては相対パスが使用されます。例えば、ディレクトリ構造が以下の様な状況で、

```
|-Folder1
|      |
|      |-Folder2
|             |
|             |-Folder3
```

```
|                    |
|                    |-Folder4
```

.spp ファイルがFolder3 内部にある場合、Folder1 ならびにFolder2 内部のファイルに対する参照は、以下のように記述されます:

```
c:¥Folder1¥NameOfFile.ext
c:¥Folder1¥Folder2¥NameOfFile.ext
```

Folder3 ならびにFolder4 への参照は、以下のように記述されます:

```
.¥NameOfFile.ext
.¥Folder4¥NameOfFile.ext
```

相対パスの使用を確実なものにするには、作業しているディスク(ドライブ)のルートディレクトリに.spp ファイルを配置してください。

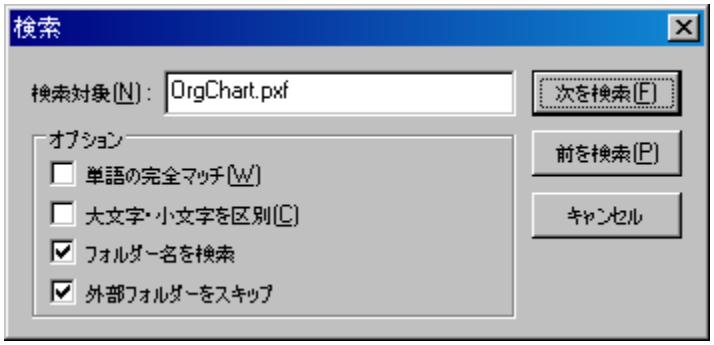
ドラッグアンドドロップ

プロジェクトウィンドウでは、フォルダーをドラッグすることで、別のフォルダーや、同じフォルダー内における別の位置へ、そのフォルダーを移動することができます。ファイルをドラッグすることで別のフォルダーへの移動を行うことはできますが、同一フォルダー内における移動はできません(フォルダー内部のファイルはアルファベット順に表示されます)。更に、W indows ファイルエクスプローラーからプロジェクトウィンドウへ、ファイルのドラッグを行うこともできます。

プロジェクト内の検索

ファイルの名前、またはファイル名の一部からプロジェクト内部にあるファイルやフォルダーを検索することができます。検索が行われると、該当するファイルやフォルダーが1つずつハイライトされた状態で表示されます。

検索を行うには、検索を行うプロジェクトフォルダーをプロジェクトウィンドウにて選択し、「**編集 | 検索**」メニューコマンドを選択するか、Ctrl+F キーを押下します。検索ダイアログが表示されるので、検索を行う文字列を入力し、目的に従うかたちで検索オプションを指定します(以下のスクリーンショットを参照)。検索オプションについては以下を参照ください。



以下の検索オプションが利用できます:

- 単語の完全マッチ: 入力された検索語とファイルまたはフォルダー名が完全に一致するかチェックされ、厳格な検索を行うことができます。ドット(".")の前後に入力された文字列も単語として認識され、例えばファイルの拡張子だけを入力して検索を行うことができます。
- 大文字小文字を区別: 検索語として入力された文字列の大文字と小文字を区別するかどうか指定することができます。
- フォルダー名を検索: フォルダー名を検索の対象に含めます。選択されていない場合、ファイル名だけが検索されます。
- 外部フォルダー[1186] を検索の対象に加える、または対象から外す事ができます。外部フォルダーはシステムまたはネットワーク上に実在しているフォルダーで、プロジェクト内部で論理的に作成されたプロジェクトフォルダーとは異なるものです。

検索が正常に行われた場合、最初にマッチしたアイテムがプロジェクトウィンドウにてハイライトされます。検索ダイアログにある**次を検索**ならびに**前を検索**ボタンをクリックすることで、その他のマッチを参照することができます。

プロジェクトの更新

外部 フォルダーに変更が加えられた場合、この変更は、プロジェクトが更新されるまで、プロジェクト ウィンドウ内で反映されません。

コンテキスト メニュー内のグローバル リソース

プロジェクト ウィンドウのフォルダーを右クリックすると、コンテキスト メニューが表示され、その中にあるグローバル リソース ファイルを追加メニュー アイテムから、グローバル リソース[934] を追加することができます。メニューコマンドにより、グローバル リソース ファイルを追加ダイアログが表示され、現在アクティブなグローバル リソース XML ファイルにある全てのファイル型ならびにフォルダー型のグローバル リソースが表示されます。目的のグローバル リソースを選択すると、選択されたプロジェクト フォルダーに追加されます。

プロジェクトならびにソース管理プロバイダー

XMLSpy プロジェクトをソース管理リポジトリに追加する場合、プロジェクト内にあるファイルを、プロジェクト ファイルが収められている場所以下に配置するようにしてください。

プロジェクト ファイルが位置している場所が、ソース管理リポジトリ内部にあるプロジェクトのルート ディレクトリになります。これ(プロジェクト ルート ディレクトリ)より上の階層にあるファイルで XMLSpy プロジェクトに加えられたファイルは、(仮にファイルの配置ができたとしても)リポジトリ内部にて予期しない場所に配置される可能性があります。

例えば、上に示したディレクトリ構造で、プロジェクト ファイルが Folder3 に保存され、ソース管理にも配置されると:

- Folder1 に追加されたファイルはソース管理にて配置されない場合があります。
- Folder2 に追加されたファイルは、(プロジェクト フォルダーではなく)リポジトリのルート ディレクトリに追加されますが、それでもソース管理以下に配置されます。
- Folder3 ならびに Folder4 に配置されているファイルは、ソース管理以下においても期待されている通りの場所に配置されます。

28.3.1 新規プロジェクト

「新規プロジェクトの作成」コマンドを選択すると、XMLSpy にて新たなプロジェクトが作成されます。既に他のプロジェクトを開いている場合、そのプロジェクトに属している全てのドキュメントを閉じるかどうか選択することができます。

28.3.2 プロジェクトを開く

「プロジェクトを開く...」コマンドでは、XMLSpy における既存のプロジェクトを開きます。既に別のプロジェクトを開いている場合、まずはそのプロジェクトが閉じられます。

28.3.3 プロジェクトの再ロード

プロジェクトの再ロードコマンドは、現在のプロジェクトをディスクから再度ロードします。複数のユーザーが作業をしているような環境では、他のユーザーがプロジェクトに加えた変更を反映するために、プロジェクトの再ロードが必要になる時があります。

メモ: プロジェクトファイル(.spp ファイル)の内容はXMLドキュメントで、その他の一般的なXMLファイルと同様の方法で編集することができます。

28.3.4 プロジェクトを閉じる

プロジェクトを閉じるコマンドにより、現在アクティブなプロジェクトが閉じられます。プロジェクトに変更が加えられている場合、閉じる前にプロジェクトを保存するか選択することができます。プロジェクトに変更が加えられている場合は、プロジェクトウィンドウに表示されているプロジェクト名に、アスタリスクが表示されます。

28.3.5 プロジェクトの保存、名前を付けて保存

プロジェクトの保存コマンドにより、現在のプロジェクトが保存されます。プロジェクトウィンドウをアクティブにして、保存 アイコンをクリックすることでも、プロジェクトの保存を行うことができます。

プロジェクトに名前を付けて保存コマンドは、現在のプロジェクトを新たに指定した別の名前で保存します。

28.3.6 ソース管理

Altova アプリケーションでは、Microsoft SourceSafe や、それに準拠したリポジトリがサポートされます。これらシステムのインストール方法については、サポートされているソース管理システム[983]を参照ください。このセクションでは、XMLSpy 内部からソース管理システムへのアクセスを行う「**プロジェクト | ソース管理**」のサブメニューコマンドについて説明します。

ソース管理機能の概要

アプリケーションプロジェクト内部にあるファイルをソース管理へ配置するメカニズムは以下のようになっています:

1. XMLSpy にて、ソース管理へ配置するファイルが含まれているアプリケーションプロジェクトフォルダーを作成します。通常アプリケーションフォルダーは、プロジェクトファイルが含まれているローカルのフォルダーに対応しています。ローカルフォルダーへのパスは、ローカルパスと呼ばれます。
2. (ソース管理 またはレポジトリとして参照される)ソース管理 システムのデータベース内では、ソース管理下に置かれるファイルを含むフォルダーが作成されます。

3. 「プロジェクト | ソース管理 | ソース管理に追加[1177]」コマンドにより、アプリケーションプロジェクトファイルがソース管理へ追加されます。
4. チェックインやチェックアウト、ソース管理からファイルを削除といった、ソース管理に関するアクションは、「プロジェクト | ソース管理[1171] サブメニュー[1171] コマンドを使用することで行うことができます。このセクションのサブセクションでは、このサブメニューにあるコマンドに関する記述を行います。

メモ: 現在使用しているソース管理プロバイダーを変更するには、(i) ソース管理オプション(ツール | オプション | ソース管理[1489])から行う方法と、(ii) ソース管理の変更ダイアログ(プロジェクト | ソース管理 | ソース管理の変更[1184])にて行う方法の2つが用意されています。

メモ: ソース管理プロジェクトは、アプリケーションプロジェクトとは異なるものです。ソース管理プロジェクトはディレクトリ構造に依存しているのに対して、XMLSpy プロジェクトはディレクトリ構造から独立した論理的な構造となっています。

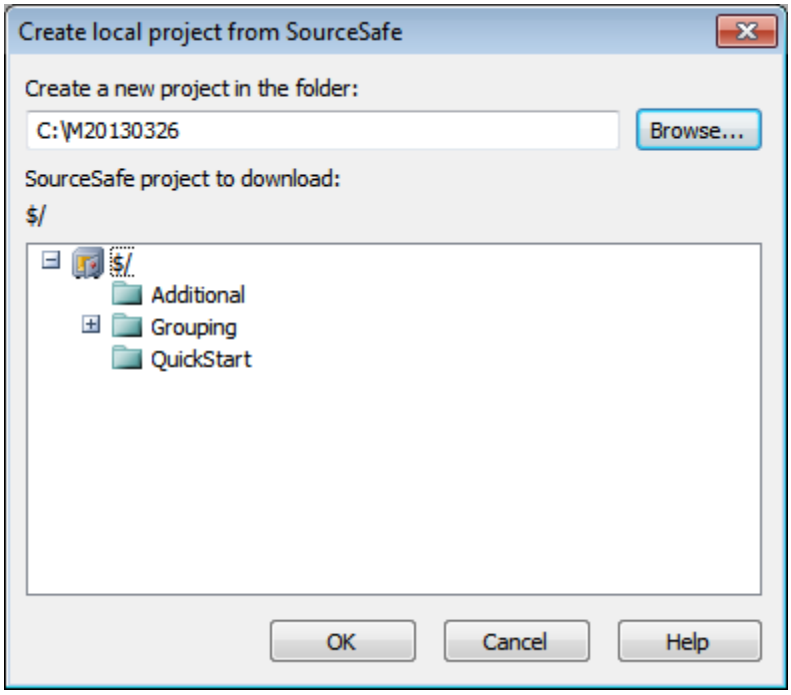
詳細に関しては次のセクションを参照してください: ソース管理[980]。

28.3.6.1 ソース管理から開く

ソース管理から開く コマンドにより新規アプリケーションプロジェクトがソース管理下のプロジェクトより作成されます。

新規プロジェクトは以下のとおり作成することができます:

1. 使用されるソース管理システムにより、場合によっては、ソース管理から新規のプロジェクトが作成される前に、プロジェクトからファイルがチェックアウトされないように注意してください。
2. 全てのプロジェクトはアプリケーション内に存在する必要はありません。
3. コマンド「プロジェクト | ソース管理 | ソース管理から開く」を選択します。
4. 現在設定されているソース管理システムは、検証と接続ダイアログをポップアップします。ローカルフォルダーに対応するレポジトリ内のバインドされたフォルダーであるレポジトリに接続を作成します。
5. ポップアップするダイアログ内で、(接続が作成された)レポジトリ内のバインドされたフォルダーのコンテンツがコピーされるローカルフォルダーを参照します(下のスクリーンショット)。下のスクリーンショットでは、バインドされたフォルダーは `MyProject` と呼ばれ、`$` サインにより表示されています。ローカルフォルダーは `C:\M20130326` です。



6. 「OK」をクリックします。バインドされたフォルダー(MyProject) のコンテンツは、次のローカルフォルダーにコピーされC:\M20130326.、新規のプロジェクトとして作成される、プロジェクトファイル(.spp ファイル) を選択するようにプロンプトするダイアログがポップアップします。
7. ローカルフォルダーにコピーされる .spp ファイルを選択します。サンプルでは、C:\M20130326 フォルダーにある MyProject.spp です。アプリケーション内にMyProject という名前の新規のプロジェクト が作成され、プロジェクトウィンドウ内に表示されます。プロジェクトのファイルは以下のフォルダー内にあります: C:¥M20130326。

ソース管理のシンボル

フォルダーやファイルにはシンボルが表示されており、それぞれ以下に類似した意味を持ちます。

	チェックイン状態。チェックアウト可能。
	他のユーザーによりチェックアウトされた状態。チェックアウト不可能。
	ローカルにチェックアウト済み。編集してチェックイン可能。

28.3.6.2 ソース管理を有効にする

ソース管理を有効にするコマンドにより、アプリケーションプロジェクトのソース管理を有効化、または無効化することができます。このオプションを選択することで、プロジェクト全体のソース管理を有効化/無効化することができます。ソース管理が有効化されると、さまざまなファイルのチェックインとアウトの状況が抽出され、プロジェクトウィンドウ内に表示されます。

	チェックイン状態。チェックアウト可能。
	他のユーザーによりチェックアウトされた状態。チェックアウト不可能。

	ローカルにチェックアウト済み。編集してチェックイン可能。

28.3.6.3 最新バージョンの取得

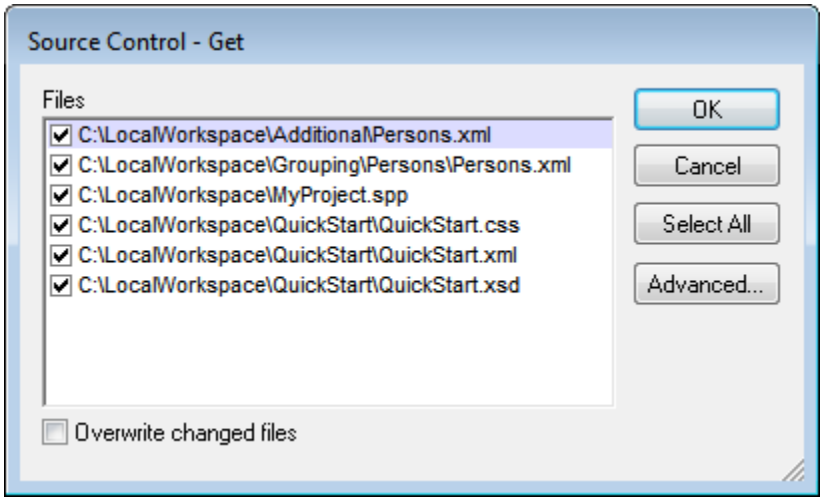
(**プロジェクト | ソース管理** メニュー内の) **最新バージョンの取得**コマンドにより、現在作業中のディレクトリにて選択されたファイルの最新版をソース管理から取得します。ファイルは読み取り専用として取得され、チェックアウトはされません。このコマンドは**Get** 1174 コマンド 1174 と同じように作動しますが、Get ダイアログは表示しません。
選択されたファイルが現在チェックアウトされている場合、ソース管理システムがどのようにこのような状況を扱うかにより取られるアクションは異なります。通常、ソース管理システムは、チェックアウトされたファイルを置き換える、マージする、またはそのままにするかを問います。

メモ: このコマンドは、フォルダーに対して使用されると、再帰的です。これは、フォルダー階層構造内の下の全てのファイルに影響を与えます。

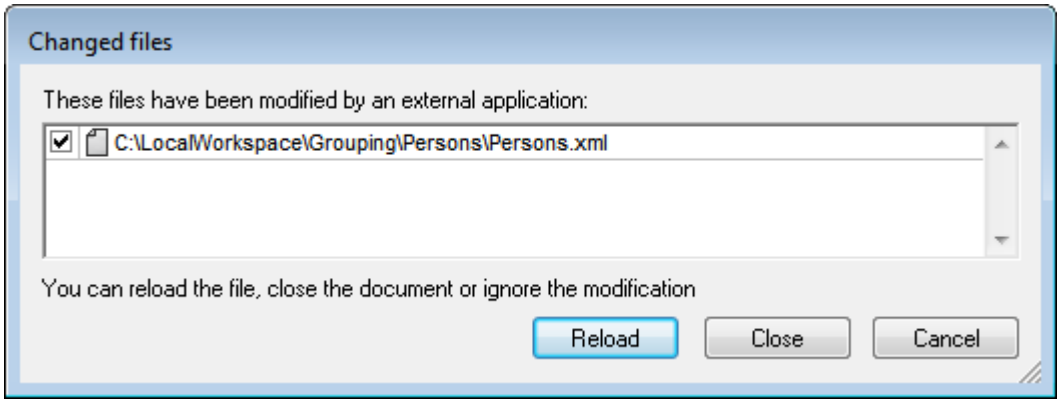
28.3.6.4 フォルダーの取得

(**プロジェクト | ソース管理** メニュー内の) **取得** コマンドはレポジトリからのファイルを読み取り専用 ファイル。として取得します（編集するにはファイルをチェックアウトする必要があります）。取得ダイアログは、「**取得**」コマンドが実行される オブジェクト 内のファイル(プロジェクトまたはフォルダー)をリストします（*下のスクリーンショットを参照*）。チェックして取得するファイルを選択することができます。

メモ: ソース管理システムが許可している場合、「**フォルダーの取得**」コマンドによりレポジトリ内の個別のサブフォルダーを選択することができます。

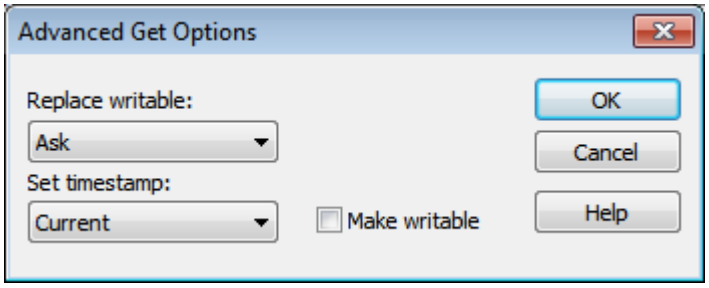


取得ダイアログの下にあるこのオプションをチェックして、変更された、または、チェックアウトされたファイルを上書きすることを選択することができます。「OK」をクリックすると、ファイルが上書きされます。上書きされたファイルが開かれていると、ファイルを再ロード(**再ロード** ボタン)、ファイルを閉める(**閉じる**)、または、ファイルの現在に表示を保持する(**キャンセル**)オプションを問うダイアログがポップアップします（*下のスクリーンショット*）。



高度な取得オプション

高度な取得オプションは取得ダイアログ内の「高度な」ボタンを使用してアクセスすることができますダイアログ（*下のスクリーンショット*）（*このセクションの最初のスクリーンショットを参照*）。



ここでは以下に関する設定をおこなうことができます：（i）チェックアウトされた書き込むことのできるファイルを置き換える。（ii）タイムスタンプ。（iii）取得されたファイルの読み取り専用プロパティを変更して書き込み可能にする。

28.3.6.5 チェックアウト、チェックイン

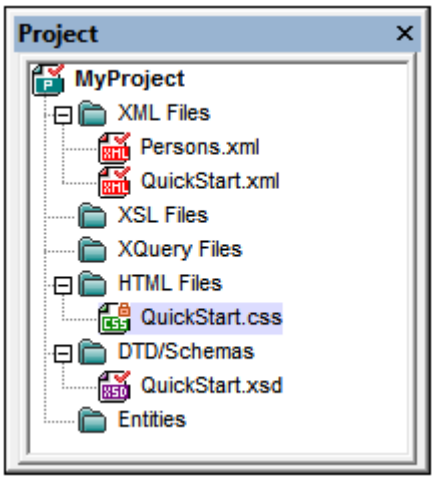
プロジェクトファイルがソース管理下に置かれると、（プロジェクトウィンドウ内で）ファイルを選択して、「**プロジェクト | ソース管理**」メニュー: 「**チェックアウト**」と「**チェックイン**」内の対応するコマンドをチェックして、チェックアウトするか、チェックインするかを選択することができます。

ファイルがチェックアウトされると、ローカルフォルダー内にレポジトリからのコピーが置かれます。チェックアウトされたファイルは編集することができます。ソース管理下にあるファイルがチェックアウトされていない場合、このファイルを編集することはできません。ファイルの編集後、レポジトリに編集を保存することができます。ファイルが保存されない場合でも、レポジトリに変更をチェックインすると変更は保存されます。ファイルがチェックアウトされているかは、アイコン内のチェック、または、ロック済みのシンボルにより表示されています。

ファイルとプロジェクトフォルダーには、下に示されるような特定のシンボルが表示されます。

	チェックイン状態。チェックアウト可能。
	他のユーザーによりチェックアウトされた状態。チェックアウト不可能。
	ローカルにチェックアウト済み。編集してチェックイン可能。

プロジェクト、または、フォルダーをプロジェクト内から選択するには、選択内のファイルを選択します。複数のオブジェクト（ファイルとフォルダー）を選択するには、オブジェクトをクリックして、Ctrl キーを押します。下のスクリーンショットはチェックアウトされたプロジェクトを示しています。ファイル `QuickStart.css` が次にチェックインされています。



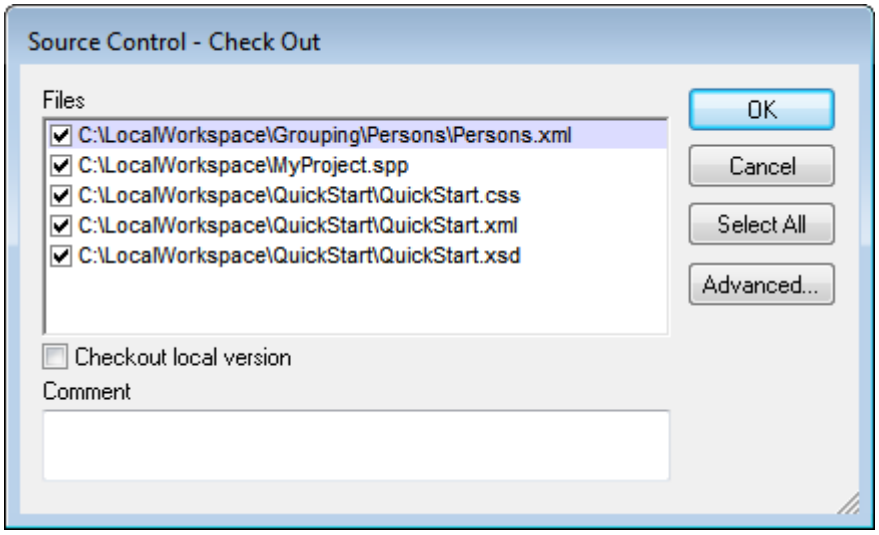
編集の変更を保存、または、拒否する

ファイルをチェックインする場合、ファイルをチェックインしたままにすることができます。ファイルをチェックアウトしたままにして、レポジトリに編集の変更を保存することを意味します。この機能は、定期的にレポジトリに編集の変更を保存して、編集を続ける場合とても役に立ちます。

ファイルをチェックアウトして、編集を行い、ファイルに加えられた変更を拒否する場合、コマンド「**プロジェクト | ソース管理 | チェックアウトの取り消し**」を選択して、レポジトリ内に保存されたドキュメントにバージョンを戻すことができます。

チェックアウト

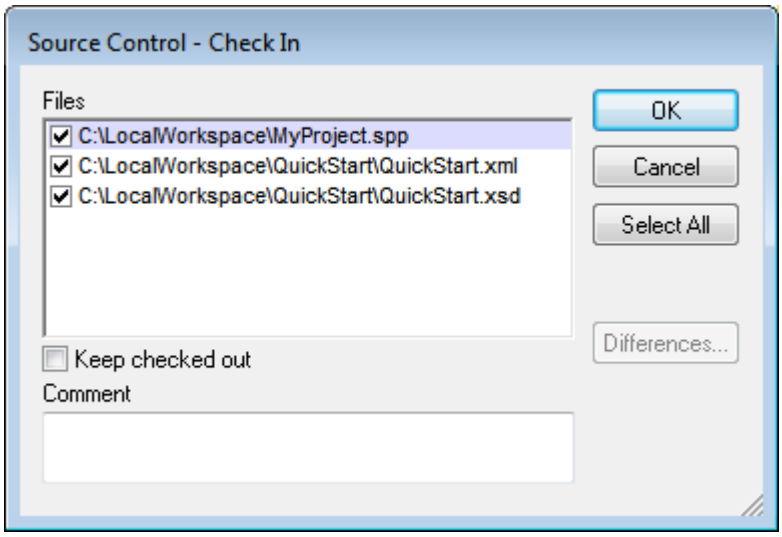
チェックアウト ダイアログ（*下のスクリーンショット*）にて以下を行うことができます: (i) ファイルをチェックアウトする、および、(ii) レポジトリバージョンを選択する、または、ローカル バージョンがチェックアウトされるかを選択する。



(C) 2015-2021 Altova GmbH

チェックイン

チェックインダイアログ（*下のスクリーンショット*）にて以下を行うことができます: (i)ファイルをチェックインする、および、 (ii) 希望する場合、ファイルをチェックアウトしたままにすることができます。



メモ: 両方のダイアログ（チェックアウトとチェックイン）内で、選択されたオブジェクト（プロジェクトまたはプロジェクトフォルダー）が複数のファイルファイルを含む場合、複数のファイルが表示されます。

28.3.6.6 チェックアウトの取り消し

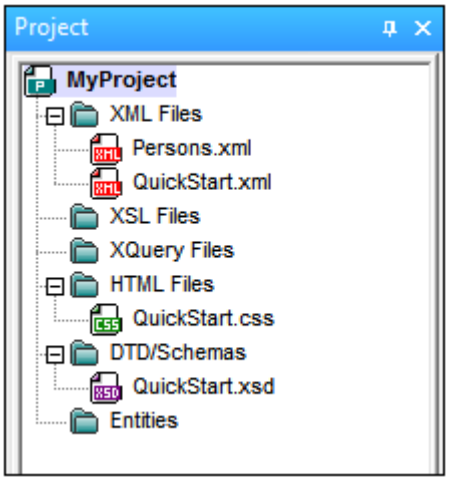
チェックアウトされたファイルが存在し、編集の変更が加えられており、変更を拒否する場合、コマンド「**プロジェクト | ソース管理 | チェックアウトの取り消し**」を使用してレポジトリ内で保存されているドキュメントのバージョンに戻り、ます。

ファイルとプロジェクトフォルダーには、下に示されるような特定のシンボルが表示されます。

	チェックイン状態。チェックアウト可能。
	他のユーザーによりチェックアウトされた状態。チェックアウト不可能。
	ローカルにチェックアウト済み。編集してチェックイン可能。

28.3.6.7 ソース管理へ追加

ソース管理にプロジェクトが追加されると、ファイルを単一に、または、グループとしてソース管理に追加することができます。プロジェクトウィンドウ内でファイルを選択し、**プロジェクト | ソース管理 | ソース管理に追加** をクリックします。複数のファイルを選択するには、追加するファイルをクリック中に**Ctrl** キーを押します。（緑）プロジェクトフォルダー上でコマンドを実行するには、全てのファイルをソース管理のフォルダーとサブフォルダー内に追加します（*下のスクリーンショットを参照*）。



ソース管理にファイルが追加されると、(プロジェクトフォルダー階層構造ではなく)ローカルフォルダー階層構造がレポジトリ内でレプリケートされます。ファイルがサブフォルダー内のローカルフォルダー内でX レベル下にあると、ファイルの親フォルダー他の祖先フォルダーは自動的にレポジトリに追加されます。

ソース管理にプロジェクトから最初のファイルが追加されると、正確なバインドレポジトリ内に作成され、プロジェクトファイル(.spp ファイル)が自動的に追加されます。詳細に関しては、ソース管理へ追加[988]のセクションを参照してください。

ソース管理シンボル

ファイルとプロジェクトフォルダーには、下に示されるような特定のシンボルが表示されます。

	チェックイン状態。チェックアウト可能。
	他のユーザーによりチェックアウトされた状態。チェックアウト不可能。
	ローカルにチェックアウト済み。編集してチェックイン可能。

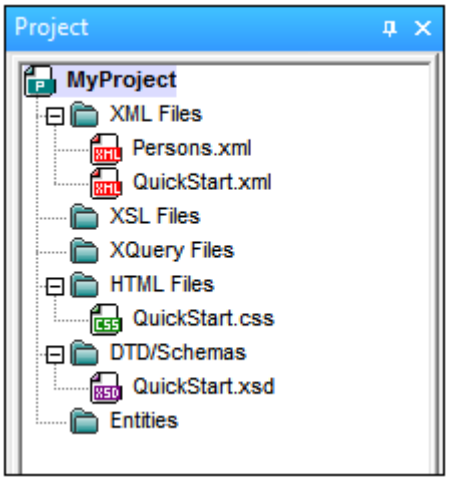
28.3.6.8 ソース管理から削除

ソース管理ファイルからファイルを削除するためには、ファイルを選択して、コマンド「**プロジェクト | ソース管理 | ソース管理から削除する**」をクリックします。以下も削除することができます: (i) フォルダー上の実行コマンドによるプロジェクトフォルダー内のファイルの削除、(ii) **Ctrl** キーを押して複数のファイルを選択しての削除(iii) プロジェクト上の実行コマンドによるプロジェクト全体の削除。

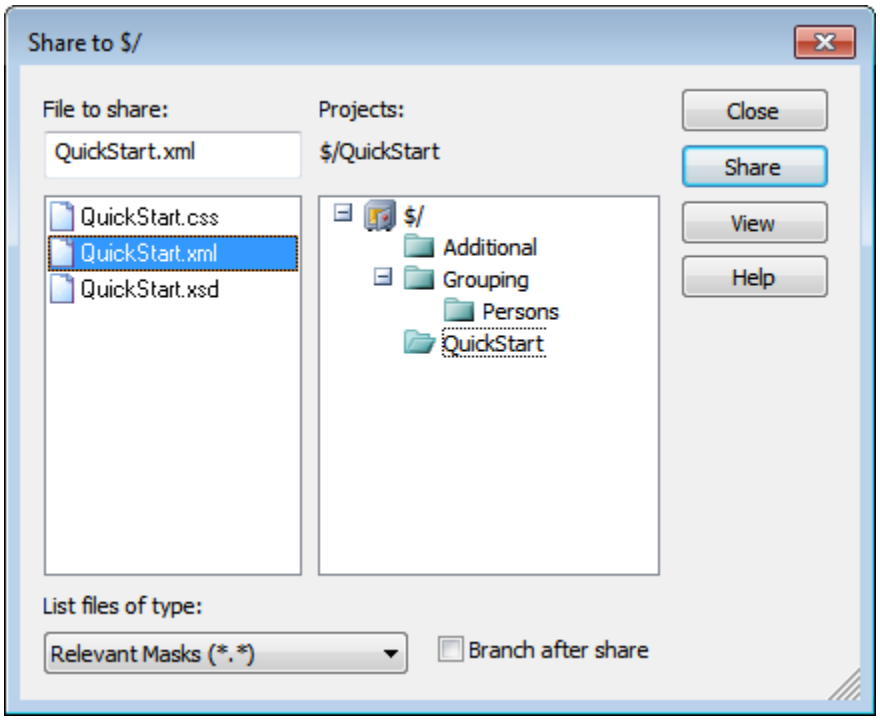
28.3.6.9 ソース管理から共有

ソース管理システムが共有するために使用されている場合、「**ソース管理から共有**」コマンドがサポートされます。複数のローカルの場所でファイルを共有することができます。これらのローカルファイルに加えられた変更は、他の「共有された」バージョンで反映されます。

アプリケーションのプロジェクトウィンドウから最初にプロジェクトを選択します(*下のスクリーンショット内でハイライトされています*)。「**ソース管理から共有**」をクリックします。

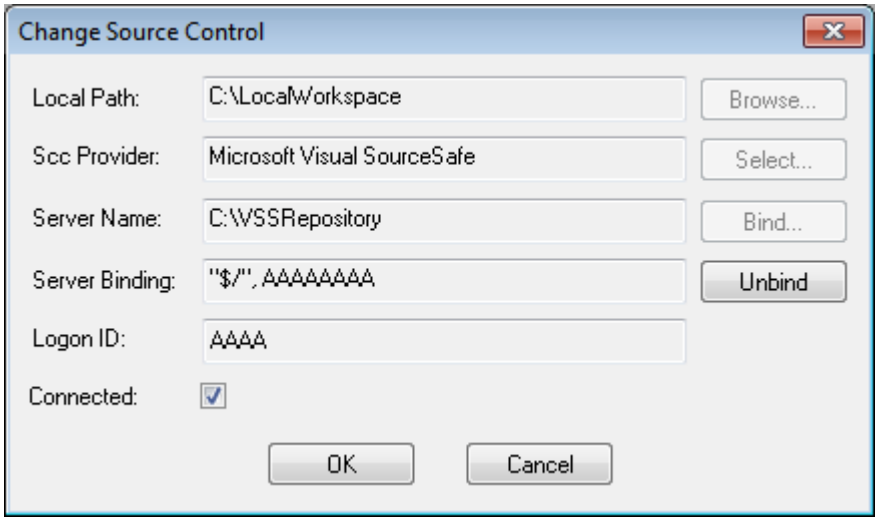


共有するフォルダーダイアログ（*下のスクリーンショット*）がポップアップします。



共有するファイルを選択するには、ファイルが存在するフォルダーが存在する右側のペイン内のプロジェクトツリーを選択します。フォルダー内のファイルは、左側のペインに表示されます。共有するファイルを選択します（複数のファイルは **Ctrl** キーを押して共有するファイルをクリックします）。選択されたファイルは 共有するファイル テキスト ボックス内に（左上）表示されます。「**共有**」をクリックして、選択されたファイルをローカルの共有フォルダーにコピーするために、「**閉じる**」をクリックします。

共有フォルダーは、共有するフォルダーダイアログの名前内に記録されます。上のスクリーンショット内では、（$ サインがローカルフォルダーがバウンドされるレポジトリ内のフォルダーであるため）ローカルフォルダーです。ローカル パスとサーバーのバインドを変更し、ソース管理の変更 ダイアログ内で共有フォルダーを確認し設定することができます（*下のスクリーンショットを参照してください*）。

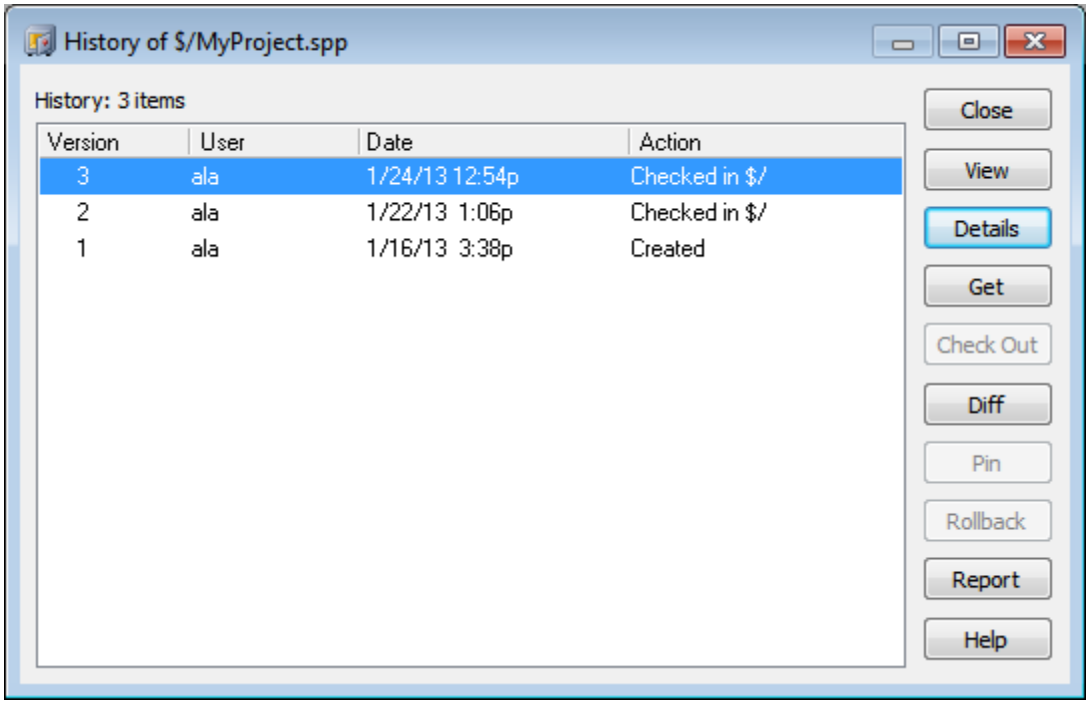


ソース管理システムの詳細に関しては、次を参照してください: ソース管理システムのユーザードキュメント。

28.3.6.10 履歴の表示

履歴の表示コマンドでは、ソース管理以下にあるファイルの履歴を表示して、プロジェクトファイル(.spp ファイル)に対して施されたそれまでの変更情報を確認し、差分の表示や、過去バージョンのファイルを取得することができます。以下の操作を行って、ファイルの履歴を確認することができます:

下のスクリーンショットは、Visual SourceSafe ソース管理のシステムの履歴ダイアログを表示しています。MyProject.spp ファイルの異なるバージョンを表示しています。



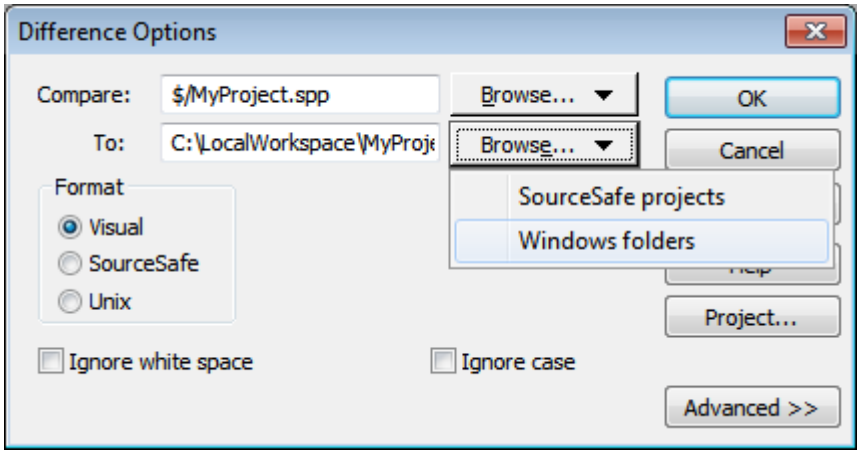
このダイアログボックスでは、目的のファイルを比較したり、特定のバージョンを取得するためのオプションが提供されます。リストにあるエントリーをダブルクリックすることで、そのファイルに関する履歴の詳細情報ダイアログボックスが表示されます。ダイアログ内に表示されるボタンでは、以下の機能が提供されます:

閉じる:このダイアログボックスを閉じます。
表示:ファイルを表示するためのアプリケーションを選択するためのダイアログボックスが表示されます。
詳細情報:現在アクティブになっているファイルのプロパティ[1183]を確認するためのダイアログボックスが表示されます。
取得:バージョンリストに表示されているファイルのバージョンを取得し、現在作業中のディレクトリに配置します。
チェックアウト:前バージョンのファイルをチェックアウトすることができます。
差分:差分表示オプション[1181]ダイアログボックスを開き、ファイルバージョンにおける差分を表示する際に使用される表示オプションを設定することができます。
ピン設定:ファイルのバージョンにピンを打つことで、特定のファイルバージョンを差分に使用することができます。
ロールバック:選択されたバージョンのファイルへロールバックします。
レポート:プリンター、ファイル、またはクリップボードへ送ることができる履歴レポートを作成します。
ヘルプ:ソース管理プロバイダープラグインのオンラインヘルプを開きます。

28.3.6.11 差分の表示

「**差異の表示**」コマンドは、プロジェクトウィンドウ内でファイルが選択されると有効化されます。プロジェクトファイル(.spp ファイル)を選択するには、プロジェクトウィンドウ内でプロジェクトのタイトルを選択します。「**差異の表示**」コマンドは、ソース管理システムの差異ツールを開始し、Altova アプリケーションからファイル間の差異が直接チェックできるようになります。

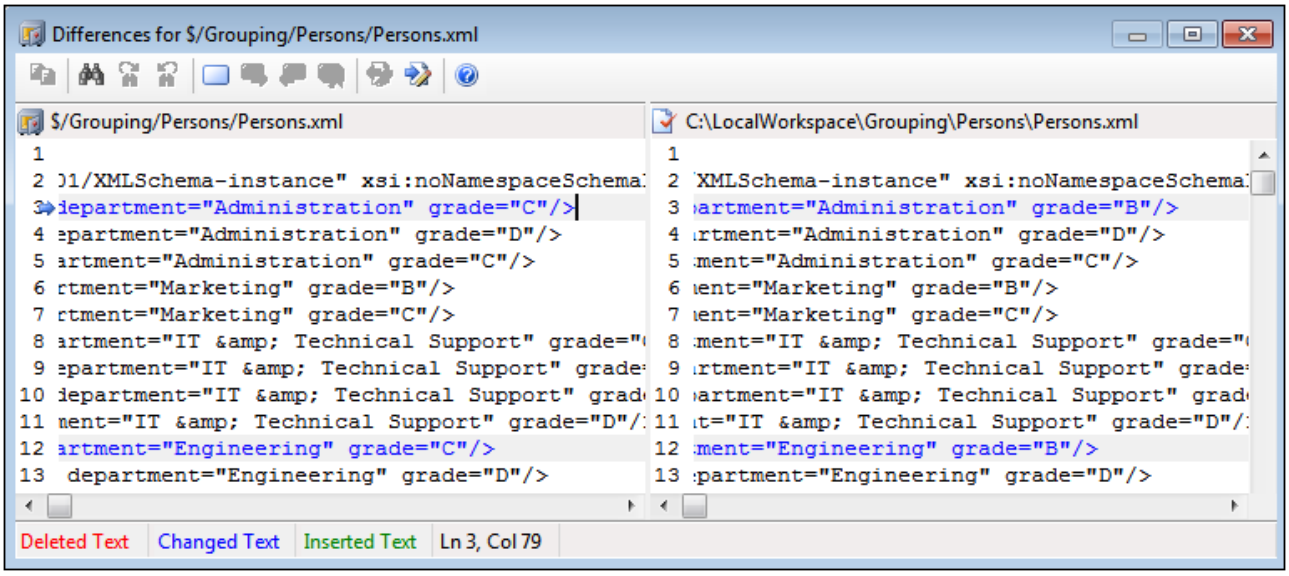
下のスクリーンショットは、Visual SourceSafe ソース管理システムの差異ツールを表示しています。



レポジトリとローカルバージョンがそれぞれ比較とテキストフィールドにデフォルトで表示されます。他のファイルの参照の方法は、以下のとおりです:

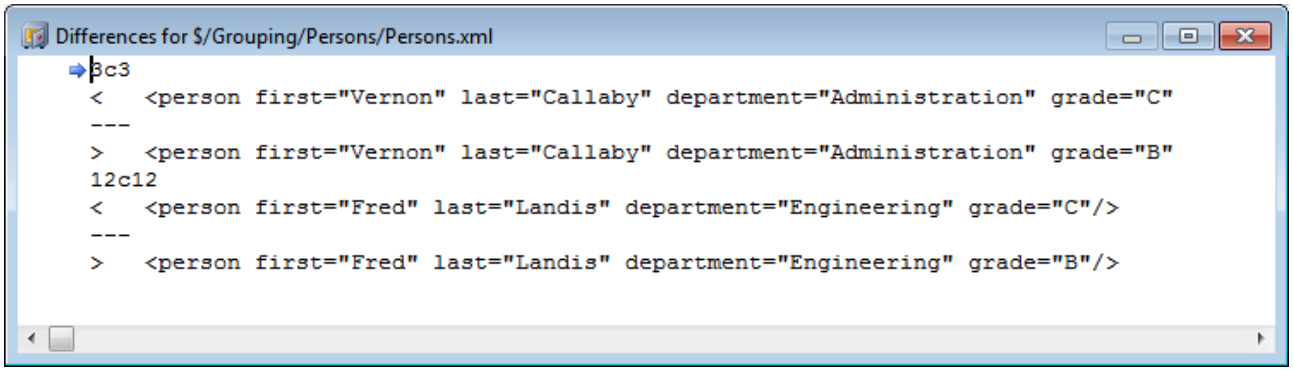
1. 「参照」ボタンドロップダウンリストから、(レポジトリフィールドを参照するために) ソースセーフプロジェクト または(ローカルフォルダーを選択するために)Windows フォルダーを選択します。
2. ファイルを参照し、選択します。

適切なエントリーを選択し、「OK」により確定します。差異の結果が個別のウィンドウに表示されます。下のスクリーンショットは、2つのフォーマットのチェックインの結果を表示しています。



上のスクリーンショットは、Visual SourceSafe 差異の結果をVisual フォーマットで表示して折、したのスクリーンショットは、Unix フォーマットでの結果を表示しています(*上のオプション ダイアログを参照してください*)。2つの差異の種類が存在し、それぞれがC からB. へのグレードの変更を表示しています。

(C) 2015-2021 Altova GmbH

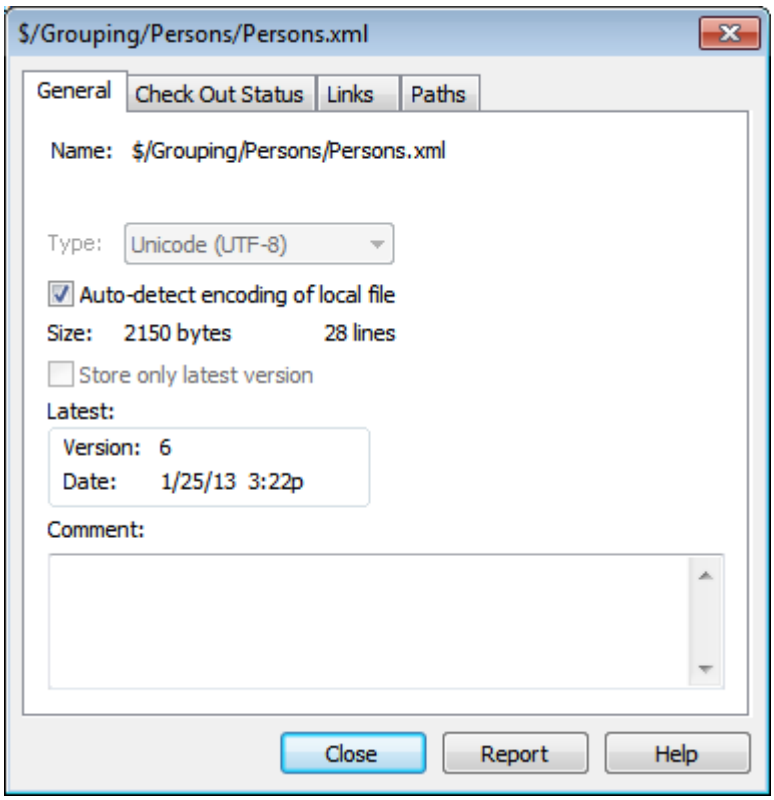


ソース管理システムの差異の処理に関しては、次を参照してください: 製品のユーザードキュメント。

28.3.6.12 プロパティの表示

プロパティの表示コマンドにより、現在選択されているファイルのプロパティが表示されます(以下のスクリーンショットを参照)。表示されるプロパティは、使用しているソース管理プロバイダーにより変化します。下のスクリーンショットは、Visual SourceSafe がアクティブなソース管理システムであることを示しています。

このコマンドは、単一のファイルのために有効化されていることに注意してください。



このコマンドは単一のファイルに対してしか使用できないことに注意してください。

28.3.6.13 ステータスを最新の状態に更新

「**ステータスを最新の状態に更新**」コマンドにより、プロジェクトファイル全てのステータスが、現在のステータスに関係なく最新の状態に更新されます。

28.3.6.14 ソース管理マネジャー

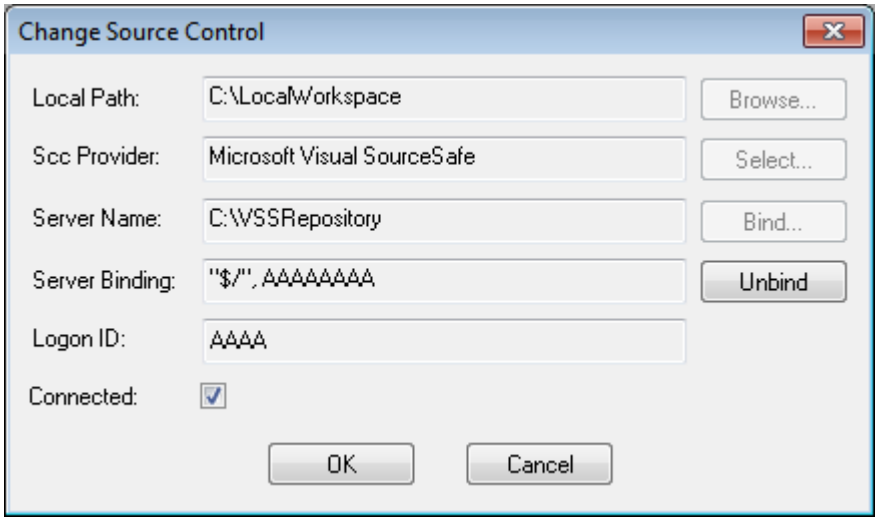
「**ソース管理**」コマンドにより、ソース管理ソフトウェアがネイティブのユーザーインターフェースにて起動されます。

28.3.6.15 ソース管理の変更

アクティブな アプリケーションプロジェクト がソース管理データベースに接続するために現在の バインド が使用されるため、現在の バインド は正確である必要があります。アプリケーションプロジェクトファイル（.spp ファイル）がローカル パスフォルダー内に存在し、レポジトリ上の バインドされたフォルダーがプロジェクトのファイルが保管されているデータベースである必要があることを意味します。通常、バインドされたフォルダーとそのサブ構造は、ローカルワークプレースとそのサブ構造に対応しています。

ソース管理の変更 ダイアログ内でソース内で管理システム（*SCC プロバイダー*）、ローカルフォルダー（*ローカル パス*）、および レポジトリ バインド（*サーバー名* と *サーバーの バインド*）.を変更することができます（*下のスクリーンショット*）。

現在の バインドの バインドを解除した後のみに設定を変更することができます。「**バインドの解除**」ボタンを使用して現在の バインドを解除します。この後、全ての設定を編集することができます。



ソース管理の変更設定 は、以下のとおりです:

1. 「**参照**」ボタンを使用してローカルフォルダーを参照し、インストールされたソース管理 システムから「**選択**」ボタンを選択します。
2. レポジトリデータベースにローカルフォルダーを バインドすることができます。「**バインド**」ボタンをクリックしてこれを行います。ソース管理システムの接続 ダイアログが表示されます。
3. *ログイン ID* を入力すると、ソース管理システムに パスされます。それ以外の場合、接続ダイアログ内に詳細を入力します。

4. ローカルフォルダーにバインドするレポジトリ内のデータベースを選択します。この設定は、1つ以上のダイアログを完成する必要がある可能性があります。
5. 設定が作成された後、「OK」ボタンをクリックしてください。

28.3.7 プロジェクトにファイルを追加

「**プロジェクト | プロジェクトにファイルを追加**」コマンドにより、ファイルを現在のプロジェクトへ追加することができます。このコマンドを使用して、ファイルをプロジェクト内部にある任意のフォルダーに追加することができます。開くダイアログでは、単一のファイルを選択したり、**Ctrl** を押下したままクリックすることで複数のファイルを選択することができます。プロジェクトにファイルを追加している場合は、プロジェクトプロパティ[1194] ダイアログボックスで定義されているファイル拡張子をベースに、適切なフォルダーにファイルが配置されます。

28.3.8 プロジェクトにグローバルリソースを追加

「**プロジェクト | プロジェクトにグローバルリソースを追加**」コマンドにより、グローバルリソースを追加ダイアログが表示され、プロジェクトに追加するファイルまたはフォルダーのグローバルリソースを選択することができます。ファイル型のグローバルリソースが選択されると、プロジェクトプロパティ[1194] ダイアログボックスで定義されているファイル拡張子をベースに、ファイルが適切なフォルダーに追加されます。フォルダー型のグローバルリソースが選択されると、そのフォルダーがファイルを開くダイアログにて開かれ、ファイルを選択するよう促されます。選択されたファイルは、プロジェクトプロパティ[1194] ダイアログボックスで定義されているファイル拡張子をベースに、適切なフォルダーに追加されます。グローバルリソースに関する詳細は、ドキュメンテーションのグローバルリソースのセクションを参照ください。

28.3.9 プロジェクトに URL を追加

「**プロジェクト | プロジェクトにURL を追加**」コマンドでは、現在のプロジェクトにURL を追加することができます。プロジェクトにURL を追加することで、URL のターゲットオブジェクトをプロジェクトに追加することができます。URL やURL オブジェクトが含まれるフォルダーに対してバッチ処理が行われると、XMLSpy によりURL からドキュメントが取得され、指定されたオペレーションが行われます。

28.3.10 プロジェクトにアクティブなファイルを追加

「**プロジェクト | プロジェクトにアクティブなファイルを追加**」コマンドにより、アクティブなファイルを現在のプロジェクトに追加することができます。ハードディスクやURL からファイルを開くと、このコマンドによりそのファイルを現在のプロジェクトに追加することができます。

28.3.11 プロジェクトにアクティブならびに関係するファイルを追加

「**プロジェクト | プロジェクトにアクティブならびに関係するファイルを追加**」コマンドを選択すると、現在アクティブな XML ドキュメントと、それに関係する全てのファイルがプロジェクトに追加されます。DTD やスキーマをベースにした XML ドキュメントを処理している場合、このコマンドを選択することで XML ドキュメントだけでなく、関連する全てのファイル(例えば、DTD ならびに DTD が参照する全ての外部エンティティ)が現在のプロジェクトに追加されます。

メモ: :処理命令により参照されるファイル(例えば XSLT ファイル)は、関係するファイルとしては認識されません。

28.3.12 プロジェクトにアクティブならびに関係するフォルダーを追加

プロジェクト | プロジェクトにプロジェクトフォルダーを追加する コマンドは、新規のフォルダーを現在のプロジェクトに追加します。このコマンドを使用して、新規のフォルダーをプロジェクトフォルダーの現在のプロジェクトまたはサブフォルダーに追加することができます。プロジェクトウィンドウ内のフォルダーを右クリックして、コンテキストメニューからこのコマンドにアクセスすることができます。

メモ: プロジェクトフォルダーをドラッグして、プロジェクトフォルダー、または、プロジェクト内の他の場所にドロップすることができます。更に、フォルダーを Windows (ファイル) Explorer からドロップして、プロジェクトフォルダーにドロップすることもできます。
メモ: プロジェクトフォルダーは緑で示され、外部フォルダー[1186] は黄色で示されています。

28.3.13 プロジェクトに外部フォルダーを追加

「**プロジェクト | プロジェクトに外部フォルダーを追加**」コマンドにより、外部フォルダーを現在のプロジェクトへ追加することができます。このコマンドによりローカルまたはネットワークフォルダーを現在のプロジェクトへ追加することができます。プロジェクトウィンドウにあるフォルダーを右クリックすることで表示されるコンテキストメニューからも、このコマンドへアクセスすることができます。

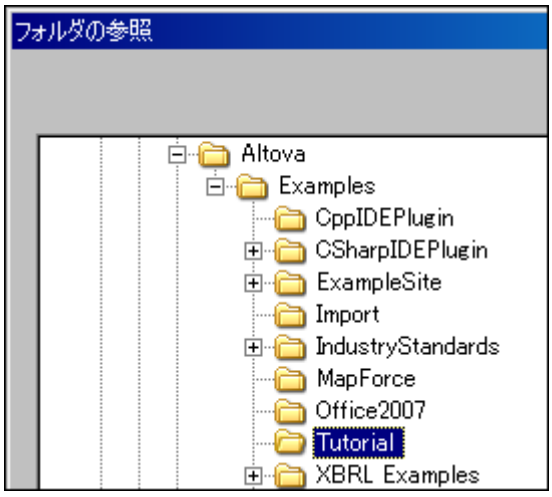
メモ: 外部フォルダーは黄色で、プロジェクトフォルダー[1186] は緑で表示されます。

メモ: 外部フォルダー内部に収められているファイルを、ソース管理内に収めることはできません。

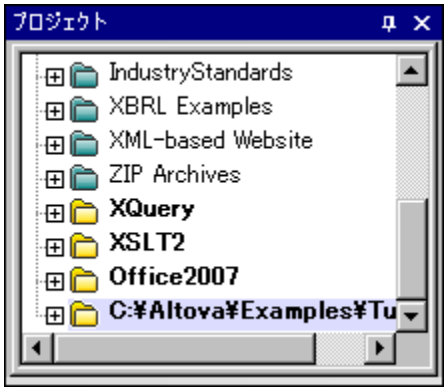
外部フォルダーをプロジェクトへ追加する

外部フォルダーをプロジェクトへ追加するには、以下の操作を行ってください:

1. メニューオプションから「**プロジェクト | プロジェクトに外部フォルダーを追加**」を選択します。
2. フォルダーの参照ダイアログボックスにて追加するフォルダーを選択し、「**OK**」により確定します。



選択されたフォルダーがプロジェクトウィンドウに表示されます。

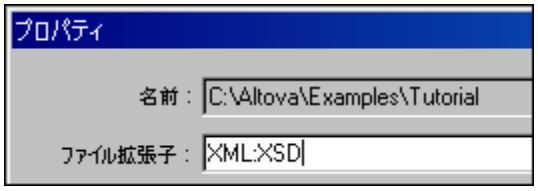


3. プラスアイコン（＋）をクリックして、フォルダーのコンテンツを確認します。

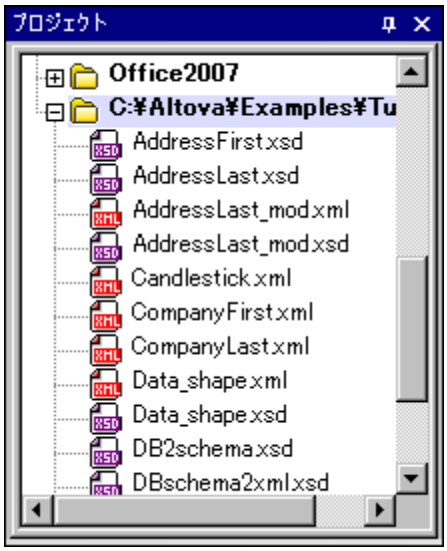
フォルダーのコンテンツをフィルタリング

フォルダー内部にあるコンテンツをフィルタリングするには、以下の操作を行ってください:

1. ローカルフォルダーを右クリックして、表示されるメニューオプションから「**プロパティ**」を選択します。プロパティダイアログボックスが表示されます。



2. ファイル拡張子フィールドをクリックし、表示したいファイル拡張子を入力します。セミコロンにより、各ファイルタイプを分けることができます（例えば、この例ではXMLとスキーマXSD が分けられています）。
3. 「**OK**」をクリックして確認します。



プロジェクトウィンドウにはXMLとXSDファイルだけが表示されるようになります。

外部フォルダーの検証

外部フォルダーの整形式チェックや検証を行うには、以下の操作を行ってください:

1. 外部フォルダーからチェックを行うファイルを選択します。
2. フォルダーを選択して、メニューコマンド[XML | 整形式をチェック]または[XML | XMLの検証]をクリックします(それぞれホットキー[F7]または[F8]が対応します)。フォルダーの下の可視化されているファイルがチェックされます。ファイルが形式に誤りがある、または、無効な場合このファイルはメインウィンドウで開かれ編集することが可能になります。
3. エラーを正して再チェックのために検証プロセスを実行します。

プロジェクトフォルダーの更新

ローカルやネットワーク上にファイルが追加されたり、削除されることもあります。フォルダーの表示を更新するには、外部フォルダーを右クリックして、ポップアップメニューオプションから「**最新の状態に更新**」を選択します。

外部フォルダーやファイルの削除

外部フォルダーを選択して「**削除**」キーを押下することで、プロジェクトウィンドウからフォルダーを削除することができます。ほかにも、外部フォルダーを右クリックして「**削除**」コマンドを選択することもできます。これらの操作によりプロジェクトウィンドウからフォルダーが削除されますが、ハードディスクやネットワーク上にあるフォルダーが削除されることはありません。

外部フォルダーにあるファイルを削除するにはハードディスクやネットワークから物理的に削除する必要があります。プロジェクトになされた変更を確認するには、外部フォルダーを右クリックして、「**最新の状態に更新**」を選択してください。

メモ: ドラッグアンドドロップにより、外部フォルダーを他のプロジェクトフォルダーや、プロジェクト内の別の場所へ移動することができます(但しほかの外部フォルダーへは不可)。更に、Windows(ファイル)エクスプローラーから、外部フォルダーを(外部フォルダー以外の)プロジェクトフォルダーへドラッグすることでも、フォルダーを追加することができます。

(C) 2015-2021 Altova GmbH

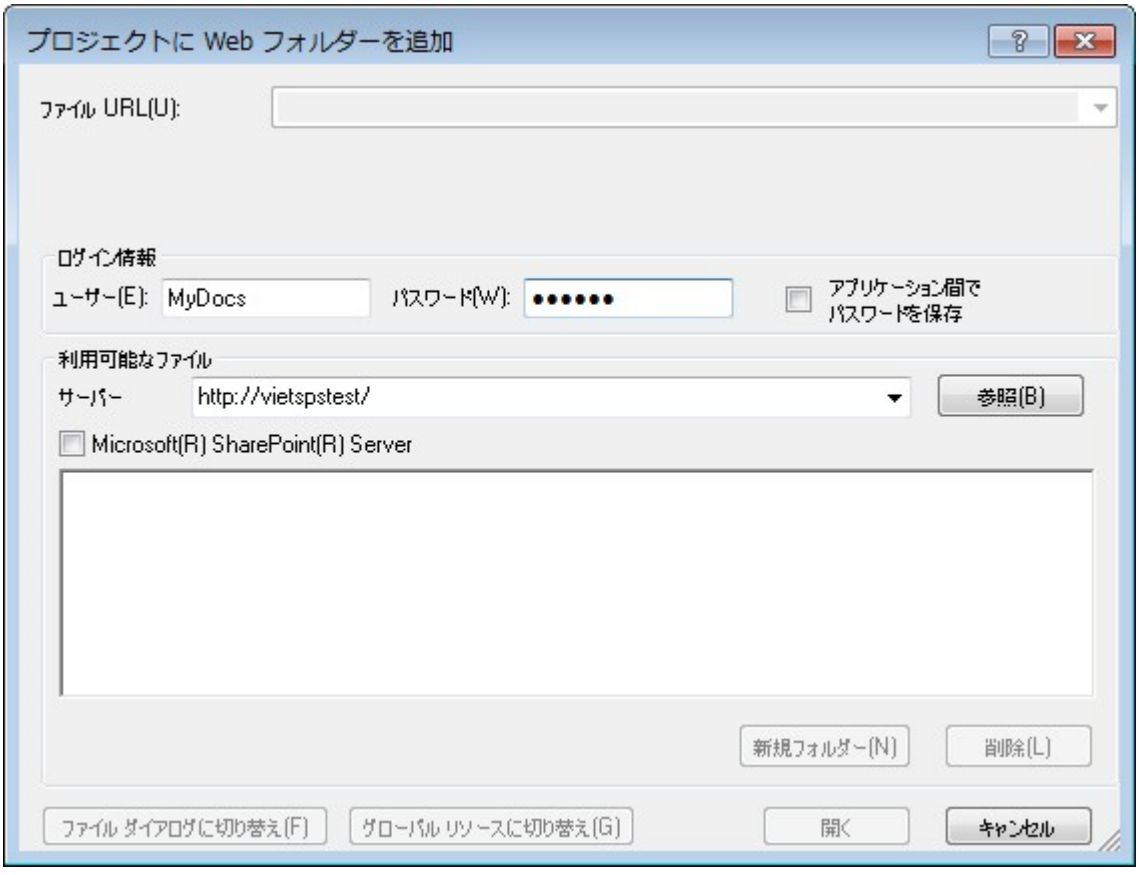
28.3.14 プロジェクトに外部ウェブフォルダーを追加

このコマンドは現在のプロジェクトに新規の外部 W eb フォルダーを追加します。プロジェクトウィンドウ内のフォルダーを右クリックすることによりコンテキストメニューからこのコマンドにアクセスすることができます。外部フォルダーに含まれるファイルは、ソース管理の下に置くことはできません。

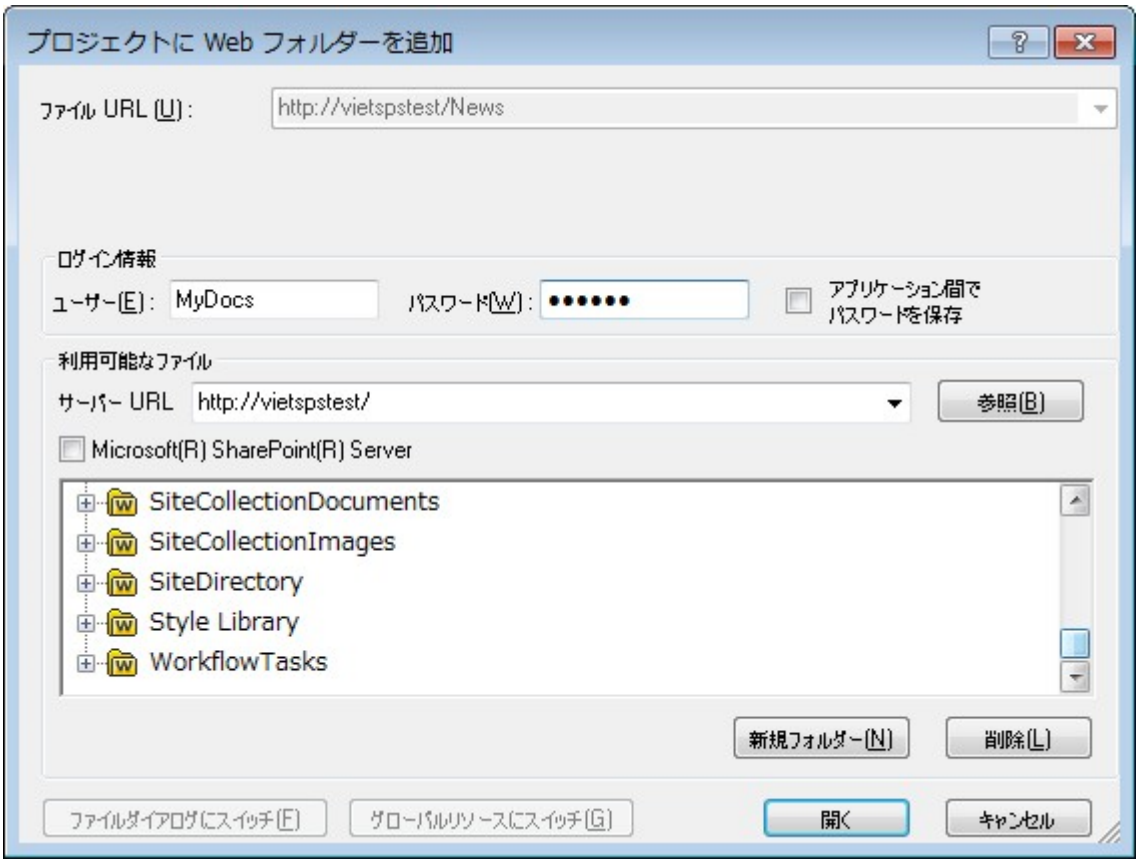
外部 W eb フォルダーをプロジェクトに追加する

外部 W eb フォルダーをプロジェクトに追加するには、以下を行います:

1. メニューオプション「**プロジェクト | 外部 W eb フォルダーをプロジェクトに追加する**」を選択します。プロジェクトにW eb フォルダーを追加するダイアログボックスが開かれます（*下のスクリーンショット*）。

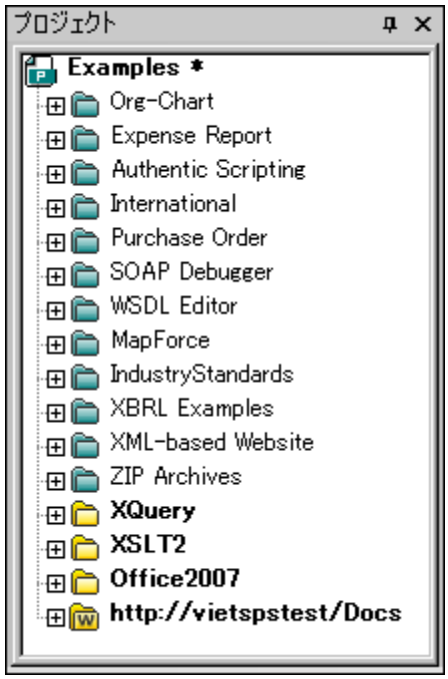


2. サーバーURL フィールドをクリックして、サーバーURL を入力します。サーバーがMicrosoft® SharePoint® Server の場合、このオプションをチェックします。サーバーのこの型のファイルと作業するための詳細に関しては、*Microsoft® SharePoint® Server* 下のセクションのフォルダーを参照してください。
3. サーバーがパスワードにより保護されている場合、ユーザーID とパスワードをユーザーとパスワードフィールドに入力してください。
4. **参照** をクリックして、サーバーに接続し、使用することのできるフォルダーを確認してください。

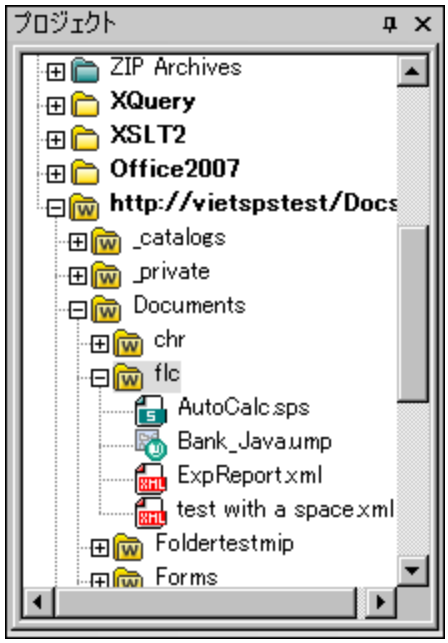


5. プロジェクトウィンドウに追加したいフォルダーをクリックします。この操作により、「開く」ボタンが有効になります。フォルダーの URL がファイル URL フィールドに表示されます。
6. 「開く」ボタンをクリックして、プロジェクトにフォルダーを追加します。

(C) 2015-2021 Altova GmbH



7. プラスアイコンをクリックして、フォルダー内部にあるコンテンツを確認することができます。



フォルダーのコンテンツをフィルタリング

フォルダーのコンテンツをフィルタリングするには、フォルダーを右クリックして、コンテキストメニューから「プロパティ」を選択します。表示されるプロパティダイアログにて、ファイル拡張子 フィールドをクリックし、表示させたいファイル種類の拡張子（例えば、XML やXSD ファイル）を入力し

ます。セミコロンによりファイルの種類を分けることができます(例: xml; xsd; sps)。プロジェクトウィンドウには、指定された拡張子を持つファイルだけが表示されるようになります。

フォルダーの検証と整形式のチェック

フォルダーに収められているファイルの整形式をチェック、または検証を行うには、フォルダーを選択して、[XML | **整形式をチェック**]または[XML | XML **の検証**]アイコンをクリックします(それぞれホットキー[**F7**]または[**F8**]が対応します)。フォルダーに表示されている全てのファイルがチェックされます。ファイルに不備があったり妥当でない場合、そのファイルがメインウィンドウで開かれ、編集を行うことができるようになります。エラーの修正を行い、再度チェックを行ってください。**Ctrl** キーを押下しながらファイルをクリックすることで、個別にファイルを選択することができ、「F7」または「F8」を押下することで、選択されたファイルがチェックされます。

プロジェクトフォルダーのコンテンツを更新

ファイルの追加など、ウェブフォルダー内のファイルは常に更新される可能性があります。フォルダーのビューを更新するには、外部フォルダーを右クリックして、コンテキストメニューから**最新の状態に更新**オプションを選択します。

フォルダーとファイルを削除

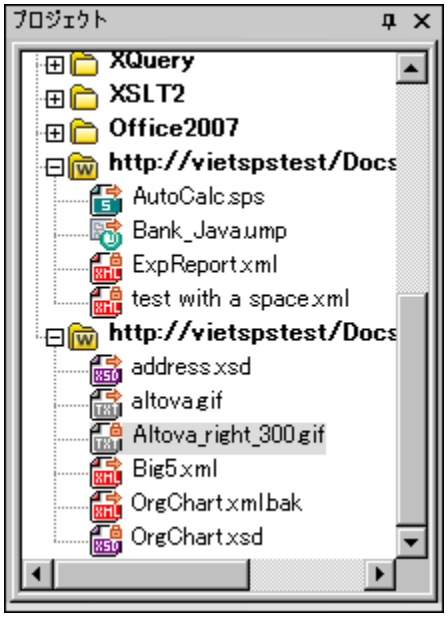
プロジェクトに追加されたのはウェブフォルダーだけであるため、プロジェクトから削除されるのもウェブフォルダーだけになります(つまりその中にあるファイルは削除されません)。ウェブフォルダーを削除するには、(i) フォルダーを右クリックして、「**削除**」を選択するか、(ii) フォルダーを選択して、「**削除**」キーを押下します。これでプロジェクトビューからフォルダーが削除されますが、ウェブサーバーにおけるファイル(やフォルダー)の削除は行われません。

メモ: 単一のファイルを右クリックして、「**削除**」キーを押しても、プロジェクトウィンドウからファイルを削除することはできません。ファイルを削除するにはサーバーから物理的にファイルを削除し、外部フォルダーのコンテンツを更新して確認する必要があります。

Microsoft® SharePoint® Server 上のフォルダー

Microsoft(R) SharePoint(R) Server 上にあるフォルダーがプロジェクトに追加されると、プロジェクトウィンドウ内に表示されているファイルのコンテキストメニューから、フォルダー内のファイルをチェックインまたはチェックアウトすることができるようになります。これらのコマンドにアクセスするには、目的のファイルを右クリックし、目的のコマンド(**チェックアウト**、**チェックイン**、**チェックアウトの取り消し**)を選択します。

ユーザーID ならびにパスワードは[プロジェクト内部にある各フォルダーのプロパティ](1194)に保存することができ、サーバーにアクセスするたびに、認証を行う必要が無くなります。



プロジェクトウィンドウ(上のスクリーンショット)では、ファイルのチェックイン/チェックアウトステータスを表すシンボルがファイルアイコンに表示されます。ファイルアイコンの説明を以下に示します:

	チェックイン状態。チェックアウト可能。
	他のユーザーによりチェックアウトされた状態。チェックアウト不可能。
	ローカルにチェックアウト済み。編集してチェックイン可能。

以下の点に注意してください:

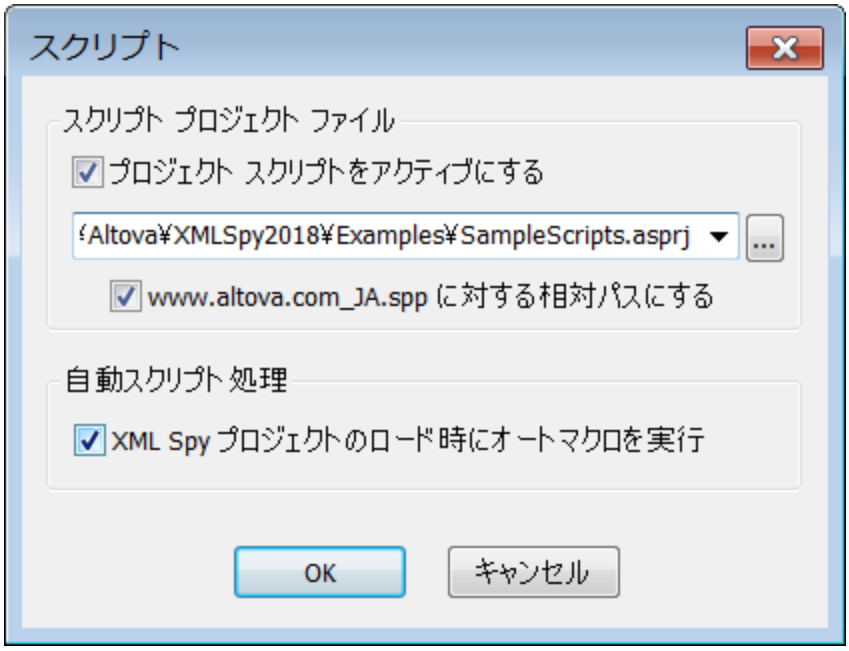
- ファイルのチェックアウトを行うと、Altova アプリケーション上でファイルの編集を行い、「**ファイル | 保存 (Ctrl+S)**」により保存を行うことができます。
- 編集されたファイルのチェックインを行うには、プロジェクトウィンドウからアクセスすることができるコンテキストメニューか、アプリケーションのメインウィンドウにあるファイルのタブを右クリックすることで表示されるコンテキストメニューから、チェックインコマンドを選択します。

- 他のユーザーによりファイルがチェックアウトされている場合、そのファイルをチェックアウトすることはできません。
- (あなた自身により)ファイルがローカルにチェックアウトされている場合、コンテキストメニューからチェックアウトの取り消しを選択することで、チェックアウトを取り消すことができます。これでファイルが変更されないままサーバー側に返されることになります。
- ある Altova アプリケーションでファイルをチェックアウトしている場合、他の Altova アプリケーションにてそのファイルをチェックアウトすることはできません。ファイルは既にチェックアウトされている状態であると認識されます。Microsoft(R) SharePoint(R) Server をサポートしている Altova 製品が、この状態で行えるコマンドは、「**チェックインとチェックアウトの取り消し**」だけです。

28.3.15 スクリプトの設定

スクリプトプロジェクトは、XMLSpy プロジェクトに以下のとおり割り当てられます:

1. XMLSpy GUI 内で、必要な *アプリケーションプロジェクト*を開きます。
2. メニューコマンド「**プロジェクト | スクリプト設定**」を選択すると、*スクリプト* ダイアログが開かれます（*下のスクリーンショット*）。



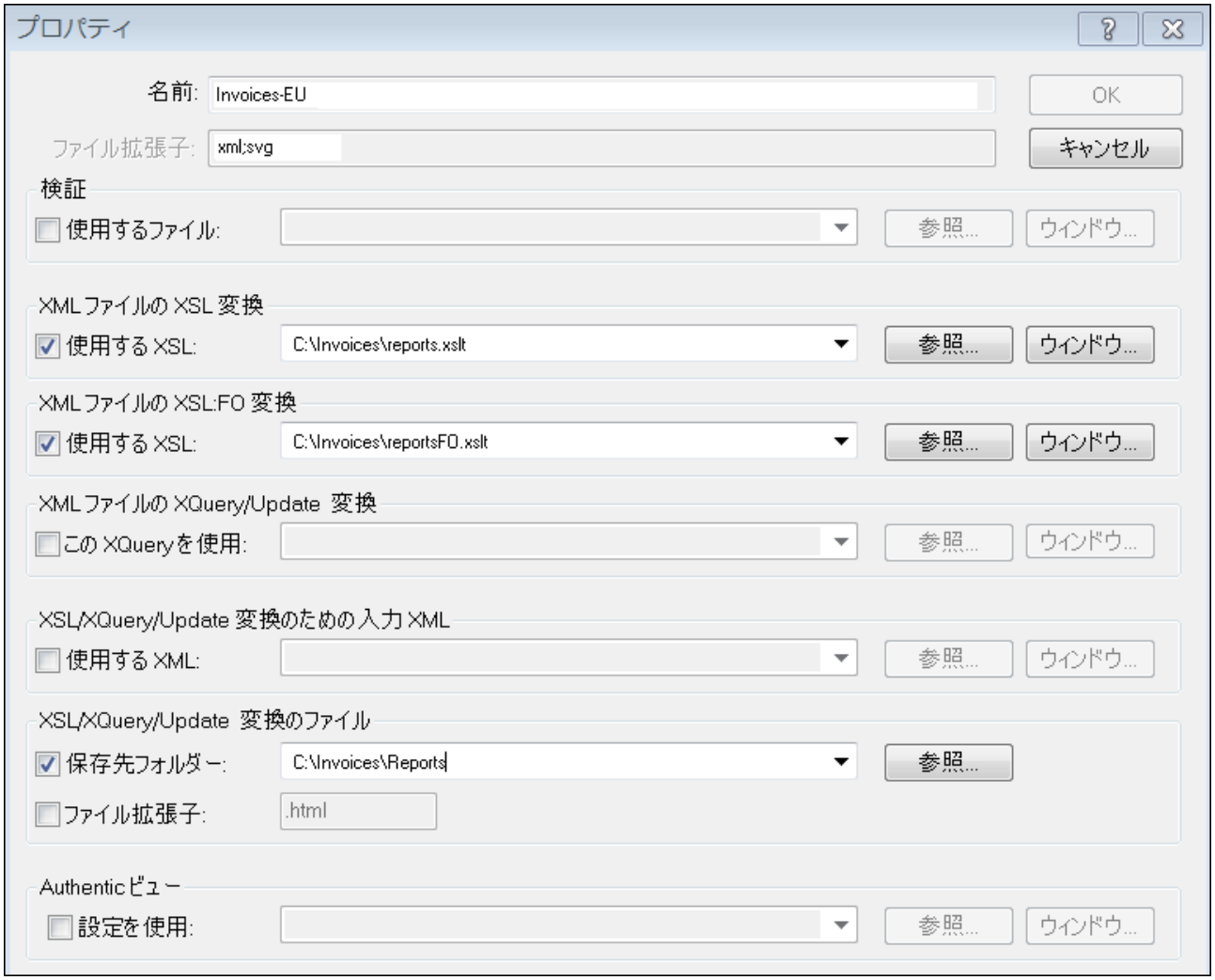
3. *プロジェクト スクリプトの有効化* チェックボックスをチェックし、必要な *スクリプトプロジェクト*（`.asprj` ファイル）を選択します。XMLSpy プロジェクトがロードされる時に、自動マクロを実行する場合、自動マクロを実行するチェックボックスをチェックします。
4. 「OK」をクリックして完了します。

メモ: XMLSpy プロジェクトのスクリプトプロジェクトを無効化（割り当てを解除）するには、*プロジェクト スクリプトを有効化する*チェックボックスのチェックを解除します。

28.3.16 プロパティ

「**プロジェクト | プロジェクトプロパティ**」コマンドは、アクティブなプロパティのプロパティダイアログを開きます（*下のスクリーンショット*）。（プロジェクトフォルダー自身をクリックすることとは異なり）プロジェクトウィンドウ内のフォルダーを右クリックし、「**プロパティ**」を選択すると、そのフォルダーのプロパティダイアログが開かれます。ダイアログ設定については下で説明されています。

メモ プロジェクトファイルがソース管理下にある場合、プロンプトが表示され、プロジェクト（`.spp`）ファイルをチェックアウトするかが問われます。設定を編集し、保存するには、「OK」をクリックします。



設定

ファイル拡張子
ファイル拡張子 設定は、個別のフォルダーのために有効化されており、プロジェクトフォルダーのためには有効化されていません。ファイルがプロジェクトに追加されると、定義されているファイル拡張子を持つフォルダーに追加されます。`MyReport.xml` という名前のファイルがプロジェクトに追加されたと仮定します。(上のスクリーンショット内で示されるとおり) `.xml` ファイル拡張子が `Invoices-EU` フォルダーで設定されている場合、`MyReport.xml` が `Invoices-EU` フォルダーに追加されます。XML ファイルに追加するフォルダーが1つ以上存在する場合、個別のXML ファイルを(プロジェクトではなく)直接フォルダーに追加します。

外部フォルダーのためのユーザーID とパスワード
(外部 Web フォルダーを含む) 外部フォルダーでは、サーバーへのアクセスに必要な可能性のあるユーザーID とパスワードを保存することができます。

検証
現在のフォルダー（または、プロパティがプロジェクトの場合はプロジェクト全体）、DTD、XML スキーマ、またはJSON スキーマ[662] 内で使用される検証[1227] ファイル。

XML ファイルのXSL 変換
XSLT スタイルシートはフォルダー内でのXML ファイルのXSLT 変換[1279] のために使用されます。

XML ファイルのXSL-FO 変換
XSLT スタイルシートは、フォルダー内のXSL-FO でのXML ファイル変換のために使用されます。

XML ファイルのXQuery/Update 変換
フォルダー内のXML ファイルのXQuery またはXQuery Update 実行のために使用されるXQuery またはXQuery Update ファイル。

XML ファイルのXSL/XQuery/Update 変換のためのXML 入力
XSLT 変換のために入力として使用されるXML ファイル、またはフォルダー内で対応するXSLT、XQuery またはXQuery Update ファイルを持つXQuery/XQuery Update 実行

XSL/XQuery/Update 変換のための出力ファイル
変換の保存先ディレクトリと任意の結果ドキュメントのファイル拡張子。

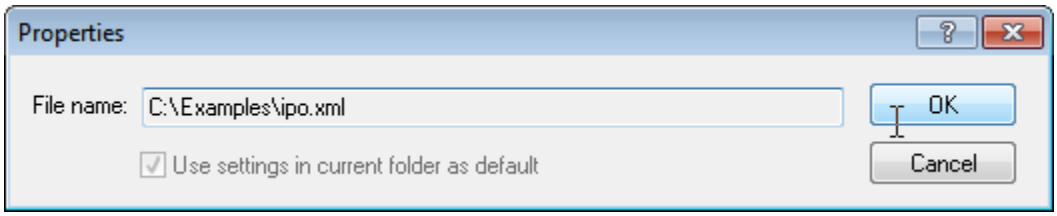
XULE 実行
XMLSpy アプリケーションウィンドウ内でアクティブなXULE ドキュメントを処理するためのXBRL インスタンスファイル

Authentic ビュー
*構成の使用*は、フォルダー内のXML ファイルのAuthentic View 表示のために使用されるStyleVision Power Stylesheet (SPS ファイル) を指定します。XML ファイルは、SPS のために使用される同じスキーマに対して有効である必要があります。

プロジェクトプロパティの注意点

以下の点と優先事項に注意してください：

- 変換、または、XSLT/XQuery 変換がプロジェクトフォルダーコンテキストメニューにより実行される場合、このダイアログで指定される検証 または変換 ファイルに XML ファイル内の割り当てより先優先順位が与えられます。更に、個別のプロジェクトフォルダーのために指定されている設定に、祖先フォルダーのために指定されている設定より先優先順位が与えられます。
- プロジェクトの複数フォルダー内で1つのファイルが存在する場合、異なるフォルダー内の異なる検証または変換 ファイルは、異なるフォルダーに割り当てられ、プロジェクトの外部でファイルが処理される場合、割り当てを設定することができます。これらの設定を指定する方法は以下のとおりです: 割り当てを希望するプロジェクトフォルダー内のファイルを検索します。プロジェクトフォルダー内のプロジェクトを右クリックして、「**プロパティ**」を選択します。（*下のスクリーンショット*）表示されるダイアログ内で「*現在のフォルダ内の設定をデフォルトとして使用*」を選択します。（現在のフォルダーはファイルが存在するプロジェクトフォルダーです。）オプションが無効化された場合、現在のフォルダーの設定は、既にデフォルトの設定として使用されることを意味します。デフォルトではないプロジェクトフォルダー内のファイルインスタンスを選択すると、オプションが有効化され、デフォルトの設定をこのフォルダーの設定に切り替えることができます。ファイルにローカルの割り当て（ファイル自身内の割り当て）が存在する場合、ローカルの割り当てが使用され、デフォルトのフォルダー設定は無視されます。



28.3.17 最近使用されたプロジェクト

このコマンドは最近使用されたプロジェクトを9つ表示し、それぞれのファイルへの素早いアクセスを提供します。

XMLSpy を起動すると、最後に使用されていたプロジェクト 1455 を自動的に開くこともできます(「ツール | オプション | ファイル」セクションにて、プログラム起動時に最後のプロジェクトを開く)。

28.4 XML メニュー

XML メニューには、XML ドキュメントの処理を行う際に使用されるコマンドが収められています。要素の挿入や追加、要素構造の変更、名前空間プレフィックスのセット、個別の XML ドキュメントにおける文脈にて XPath の評価を行うことができます。

最も頻繁に使用される XML 処理には、ドキュメントの整形式チェック[1225] や、XML ドキュメントの検証[1227] も含まれます。これらの処理に対応したコマンドはこのメニューに含まれます。

28.4.1 挿入

「XML | 挿入」コマンドは、(全てのビューで有効になってはいますが) グリッドビューでのみ使用することができます。サブメニューからは、以下にあるアイテムの挿入を行うことができます(以下のスクリーンショットを参照):

- XML 宣言ならびに XML ドキュメント内のノード型(属性、要素、テキスト、CDATA、コメント、処理命令)
- DOCTYPE 宣言ならびに XML ドキュメント内における外部 DTD 宣言

- DTD ドキュメント内の DTD 宣言（ELEMENT、ATTLIST、ENTITY、そして NOTATION）ならびに XML ドキュメントの DTD 宣言

28.4.1.1 属性の挿入

Ctrl+Shift+I

「XML | 挿入 | 属性」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たな属性を挿入します。属性は、現在選択されているアイテムの親要素の直後、そして子要素の前に表示されるため、挿入された属性は現在選択されているアイテムから数行前の場所に挿入されることもあります。

28.4.1.2 要素の挿入

Ctrl+Shift+E

「XML | 挿入 | 要素」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たな要素を挿入します。属性が選択されている場合、新たに作成される要素は、属性の親要素に含まれている最初の子要素の前に表示されます。

28.4.1.3 テキストの挿入

Ctrl+Shift+T

「XML | 挿入 | テキスト」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たなテキストを挿入します。属性が選択されている場合、新たに挿入されるテキストは、属性の親要素に含まれている最初の子要素の前に表示されます。

28.4.1.4 CDATA の挿入

Ctrl+Shift+D

「XML | 挿入 | CDATA」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たな CDATA ブロックを挿入します。属性が選択されている場合、新たに挿入される CDATA は、属性の親要素に含まれている最初の子要素の前に表示されます。

28.4.1.5 コメントの挿入

Ctrl+Shift+M

「XML | 挿入 | コメント」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たなコメントを挿入します。属性が選択されている場合、新たに挿入されるコメントは、属性の親要素に含まれている最初の子要素の前に表示されます。

28.4.1.6 XML の挿入

「XML | 挿入 | XML」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たな XML 宣言を挿入します。XML 宣言の子属性と、その値を挿入する必要があります。XML 宣言は以下のようになります:

```
<?xml version="1.0" encoding="UTF-8"?>
```

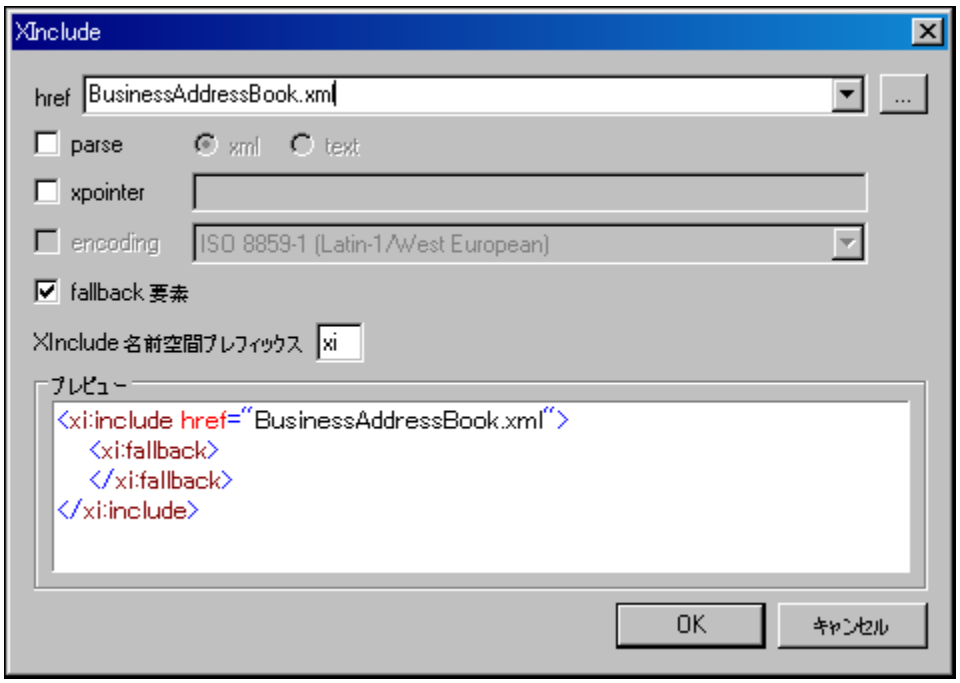
メモ: XML ドキュメントにはファイルのトップでしか XML 宣言を配置することができないため、このコマンドは XML 宣言が存在しないドキュメントの最上部でしか使用するべきではありません。

28.4.1.7 処理命令の挿入

「XML | 挿入 | 処理命令」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に処理命令 (PI) を挿入します。属性が選択されている場合、新たに挿入される処理命令 (PI) は、属性の親要素に含まれている最初の子要素の前に表示されます。

28.4.1.8 XInclude の挿入

「XML | 挿入 | XInclude」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの前に新たな XInclude 要素を挿入します。属性が選択されている場合、新たに挿入される XInclude 要素は、属性の親要素に含まれている最初の子要素の前に表示されます。このコマンドにより XInclude ダイアログが表示されます (以下のスクリーンショットを参照)。



インクルードされる XML ファイルを href テキストボックスに入力します (または、テキストボックスの右側にある ... (参照) ボタンをクリックすることで、ファイルの参照を行うこともできます)。ファイル名が href 属性の値として、XML ドキュメント内に入力されます。XInclude (xi:include) 要素の parse、xpointer や encoding 属性、そして xi:include の fallback 子要素も、このダイアログにて挿入することができます。対応するチェックボックスにチェックを入れ、必要な値を選択/入力してください。fallback 要素の場合は、チェックボックスにチェックを入れることで、空要素を挿入することができます。fallback 要素のコンテンツは、編集ビューにて追加する必要があります。

parse 属性により、インクルードされるドキュメントが XML または テキストとして解析 (パース) されるかが決定されます (デフォルトでは XML

という値がセットされており、指定を行う必要はありません)。xpointer 属性は、href 属性により指定されるドキュメント内フラグメント(つまり、インクルードされるフラグメント)を特定します。encoding 属性ではインクルードされるドキュメントのエンコーディングが指定され、XMLSpy により、インクルードするドキュメントの内容がインクルード元のドキュメントエンコーディングに合う形で変換されます。インクルードされるドキュメントが見つからない場合は、xi:include 要素がfallback 子要素により置換えられます。

XInclude により2つのXMLドキュメントをインクルードするXMLドキュメントのサンプルを以下に示します:

```
<?xml version="1.0" encoding="UTF-16"?>
<AddressBook xsi:schemaLocation="http://www.altova.com/sv/myaddresses AddressBook.xsd"
       xmlns="http://www.altova.com/stylevision/tutorials/myaddresses"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="BusinessAddressBook.xml"/>
       <xi:include href="PersonalAddressBook.xml"/>
</AddressBook>
```

XMLドキュメントがパースされると、上の例で2つあるXInclude 要素が、href 属性により指定されたファイルの内容に置換えられます。

xml:base

XMLSpy のXML バリデーターによりXMLドキュメントが読み取られ、XInclude 名前空間内のinclude 要素(これ以降はxi:include と表記)が発見されると、この要素(xi:include)を、xi:include 要素のhref 属性により指定されたXMLドキュメントに置換えます。インクルードされたXMLドキュメントのドキュメント要素(ルート要素)や、XPointer により指定された要素にもxml:base 属性は含まれ、インクルードされたベースURI を保持します。(インクルードされたXML やツリーフラグメントを含む)結果XMLドキュメントが、スキーマに従う形で妥当でなければならない場合、インクルードされたドキュメントのドキュメント要素(ルート要素)やツリーフラグメントのトップレベル要素を、xml:base 属性を許可するコンテンツモデルにより作成する必要があります。スキーマにより、xml:base 属性がこの要素に対して認められない場合、ドキュメントは妥当でないと評価されます。XMLSpy のスキーマビューにて要素コンテンツモデルのxml:base 属性を定義する方法は、ドキュメンテーションのスキーマビューセクション、xml: プレフィックス属性 228 にて記述されます。

XPointers

XMLSpy ではXInclude 内におけるXPointer がサポートされます。関連するW 3C 勧告は、XPointer Framework ならびにXPointer element() スキーマの2つです。XInclude 要素内のXPointer により、XMLドキュメント全体の代わりに、XMLドキュメント内部でインクルードする箇所を特定することができます。XPointer は、以下にあるように、XInclude 要素内部で使用されます::

```
<xi:include href="PersonalAddressBook.xml" xpointer="element(usa)"/>
<xi:include href="BusinessAddressBook.xml" xpointer="element(/1/1)"/>
<xi:include href="BobsAddressBook.xml" xpointer="element(usa/3/1)"/>
<xi:include href="PatsAddressBook.xml" xpointer="element(usa)element(/1/1)"/>
```

XPointer のelement() スキーマでは、NCName や子シーケンスを使用することができます。

- 上の例にある最初のxi:include 要素のxpointer 属性ではusa というNCName を伴った要素スキーマが使用されています。XPointer Framework に従うと、このNCName によりusa というID を持つ要素が特定されます。
- 2つ目のxi:include では、element(/1/1) という値がxpointer 属性で使用されています。最初のステップでは、ドキュメントにある最初の子要素(つまりドキュメントが整形式であれば、ドキュメント要素)が特定されます。次のステップでは、最初のステップで特定された要素から見た最初の子要素が特定され、結果としてドキュメント要素にある最初の子要素がインクルードされます。
- 3番目のxi:include にあるxpointer 属性では、NCName と子シーケンスの両方が使用されています。XPointer により、usa というID を持つ要素の3番目の子要素、その子要素から見た最初の子要素がインクルードされます。.
- 最初に使用したXPointer が期待通りに動作するか分からない場合、4番目の例として示されているxpointer="element(usa)element(/1/1)" という書式を使用することもできます。この方法では、たとえusa というID を持つ要素が存在しない場合でも、ドキュメント要素の最初の子要素が選択されます。これ以外にもバックアップとなる式を追加することができます。各XPointer は、空白スペース以外で分けることはできず、以下のような書式に従う必要があります: xpointer="element(usa)element(addresses/1)

element(/1/1)"。

メモ: element() スキーマは qualified name でサポートされておらず、名前空間バインディングコンテキストは、element() スキーマにて使用されません。

28.4.1.9 DOCTYPE の挿入

「XML | 属性 | DOCTYPE」コマンドは XML ファイルを開いているグリッドビューで、最上位のノードが選択されている時のみ使用することができます。このコマンドにより DOCTYPE 宣言が XML ドキュメントのトップに追加されます。DOCTYPE の名前を入力する必要がありますが、名前はドキュメント要素のそれと同一でなければなりません。

DOCTYPE の名前を入力すると、DTD 内部サブセットで使用する宣言を入力することができます。

メモ:

- DOCTYPE 宣言は XML 宣言と XML ドキュメント要素の間にしか記述することができません。
- このコマンドの代わりに DTD の割り当て コマンドを使用することで、外部 DTD ドキュメントを参照する DOCTYPE ステートメントを作成することもできます。

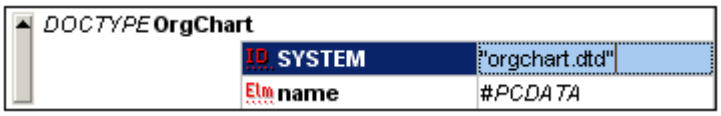
28.4.1.10 ExternalID の挿入

XML ファイル内の DOCTYPE 宣言には、DTD 宣言を含んでいる外部リソースに対する参照を含めることができます。このリソースには PUBLIC または SYSTEM 識別子により参照することができます。以下にその例を示します:

```
<!DOCTYPE doc_element_name PUBLIC "publicID" "systemID">
<!DOCTYPE doc_element_name SYSTEM "systemID">
```

SYSTEM 識別子は外部リソースを識別する URI です。PUBLIC 識別子は場所に依存するものではなく、外部リソースの場所を相互参照するために使用することができます。例えば、XMLSpy がインストールされると、広く使用されている DTD や XML スキーマが、catalog.xml という名前のファイル内に記述され、その実体が C:\Program Files\Altova\Common2021\Schemas\ フォルダー以下に収められることになります。XML ドキュメント内の PUBLIC 識別子は、これらのファイル内に記述されている DTD の相互参照に使用することができます。

グリッドビューにて XML ファイル内の DOCTYPE アイテムの「子」アイテムが選択されていると、「XML | 挿入 | ExternalID」コマンドが有効になります。このコマンドにより、外部識別子（PUBLIC または SYSTEM）のグリッドビュー行が挿入されます。識別子の種類とその値を入力してください。



上に示されるスクリーンショットのグリッドビューに対応したテキストビューは、以下のようになります:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

メモ: ExternalID の行は、選択されている DOCTYPE アイテムの子として追加したり、DOCTYPE アイテムの子アイテム（例: 上の例における name という ELEMENT 宣言）が選択されているときに挿入や追加することができます。

28.4.1.11 ELEMENT の挿入

「XML | 挿入 | ELEMENT」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントや XML ドキュメントの DOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、選択された宣言の前に ELEMENT 宣言が挿入されます。

28.4.1.12 ATTLIST の挿入

「XML | 挿入 | ATTLIST」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントや XML ドキュメントの DOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、選択された宣言の前に ATTLIST 宣言が挿入されます。

28.4.1.13 ENTITY の挿入

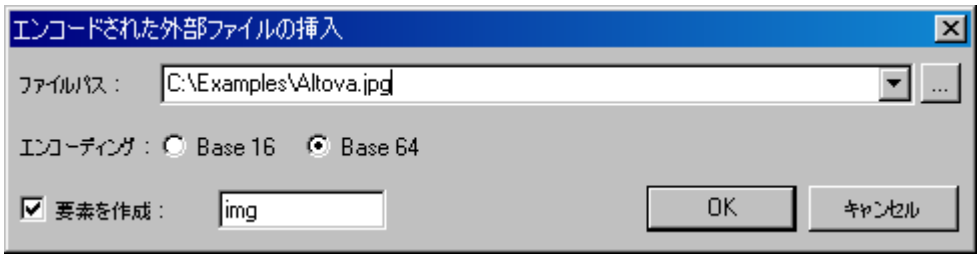
「XML | 挿入 | ENTITY」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントや XML ドキュメントの DOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、選択された宣言の前に ENTITY 宣言が挿入されます。

28.4.1.14 NOTATION の挿入

「XML | 挿入 | NOTATION」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントや XML ドキュメントの DOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、選択された宣言の前に NOTATION 宣言が挿入されます。

28.4.1.15 エンコードされた外部ファイルの挿入

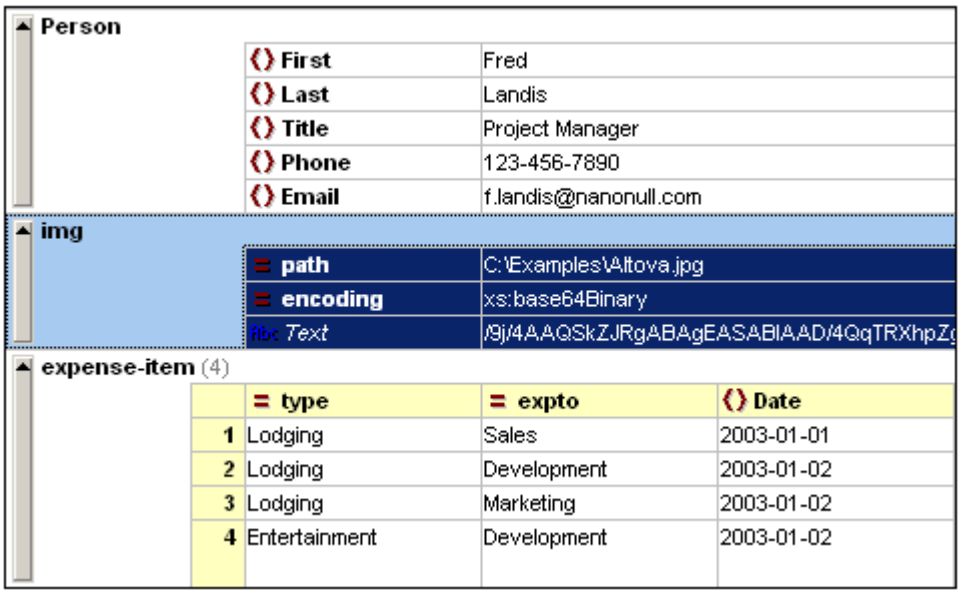
「**XML | 挿入 | エンコードされた外部ファイル**」コマンドはグリッドビューでのみ使用できます。*イメージファイルといったバイナリファイル*を、エンコードされた文字として挿入することができます。エンコードされた外部ファイルはグリッドビュー内で選択された要素の前に挿入されます。コマンドをクリックすると、エンコードされた外部ファイルの挿入ダイアログが表示されます(*以下のスクリーンショットを参照*)。ファイルへのパスを入力し、必要とするエンコードを選択し(周りの要素無しに)エンコードが現在選択されている要素の前にテキストとして挿入されるかを指定します。



エンコードならびに埋め込みが行われる外部ファイルの参照や名前の入力を行うことができます。Base-16 または Base-64 エンコーディングを指定する必要があります。エンコードされたテキストを要素内に収めるには、要素の作成チェックボックスにチェックを入れて、*要素の作成*テキストボックスに使用する要素の名前を入力してください。*要素の作成*チェックボックスがチェックされていない場合、エンコードされたテキストはカーソル位置に直接挿入されます。

「OK」をクリックすることで選択されたファイルのエンコードされたテキストが囲まれた要素と共にカーソルの現在の選択個所の前に挿入されます。

エンコードされたファイルがグリッドビューにて表示されます(*以下のスクリーンショットにおけるハイライトされた部分*)。



テキストビューでは、以下のような内容がファイルに挿入されます。

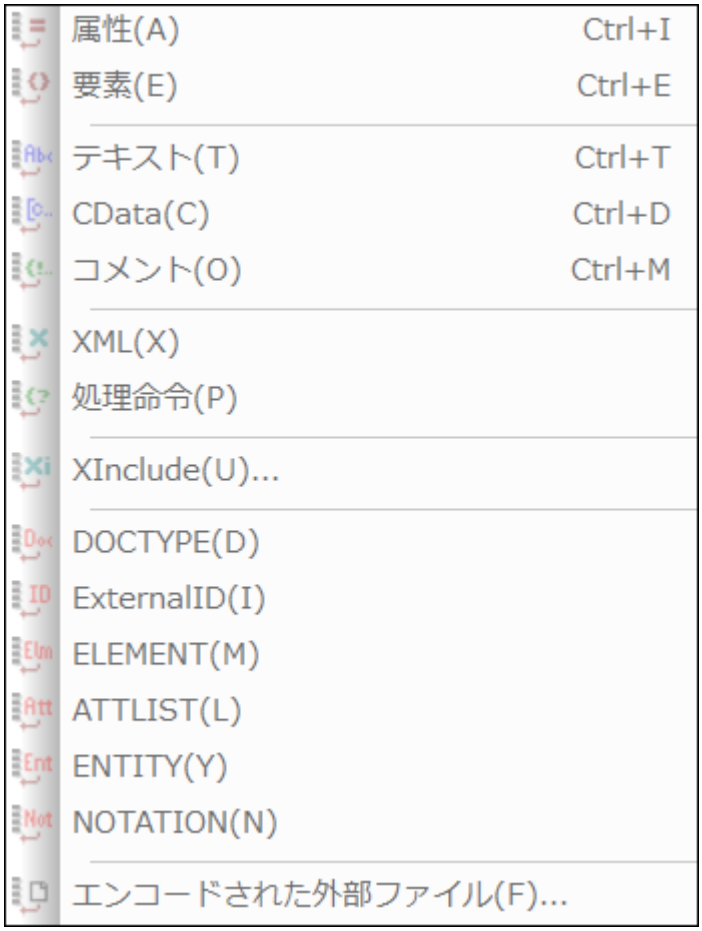
```
<img ext="jpg" encoding="xs:base64Binary">
  iVBORw0KGgoAAAANSUhEUgAAABAAAAAQAQMAAAAlPW0iAAAABlBMVEUAAAD/
  //+l2Z/dAAAAM0lEQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83NDDeNGe4U
  g9C9zwz3gVLMDA/A6P9/AFGGFyjOXZtQAAAAAElFTkSuQmCC
</img>
```

上の表示はJPG イメージファイルをテキストにエンコードしたものです。エンコードされたテキストの周りに `img` 要素が作成されます。

28.4.2 追加

「XML | 追加」コマンドは、(全てのビューで有効になっていますが) グリッドビューでのみ使用することができます。サブメニューからは、以下にあるアイテムの追加を行うことができます(以下のスクリーンショットを参照):

- XML 宣言ならびにXML ドキュメント内のノード型(属性、要素、テキスト、CDATA、コメント、処理命令)
- DOCTYPE 宣言ならびにXML ドキュメント内における外部 DTD 宣言
- DTD ドキュメント内のDTD 宣言(ELEMENT、ATTLIST、ENTITY、そしてNOTATION)ならびにXML ドキュメントの DTD 宣言



28.4.2.1 属性の追加

Ctrl+I

「XML | 追加 | 属性」コマンドはグリッドビューでのみ使用でき、新たな属性を追加します。

28.4.2.2 要素の追加

Ctrl+E

「XML | 追加 | 要素」コマンドはグリッドビューでのみ使用でき、選択された要素の兄弟要素として、最後の位置に新たな要素ノードを追加します。属性ノードが選択されている場合、属性の親要素から見た子要素の最後に新たな要素が作成されます。

28.4.2.3 テキストの追加

Ctrl+T

「XML | 追加 | テキスト」コマンドはグリッドビューでのみ使用でき、選択された要素から見て最後の兄弟要素の後にテキストブロックが挿入されます。属性ノードが選択されている場合、属性の親要素から見た子要素の最後に新たなテキストブロックが作成されます。

28.4.2.4 CDATA の追加

Ctrl+D

「XML | 追加 | CDATA」コマンドはグリッドビューでのみ使用でき、(属性ノードを除く)選択された任意のノードから見た兄弟要素の最後にCDATA ノードを追加します。属性ノードが選択されている場合、属性の親要素から見た子要素の最後にCDATA セクションが追加されます。

28.4.2.5 コメントの追加

Ctrl+M

「XML | 追加 | コメント」コマンドはグリッドビューでのみ使用でき、(属性ノードを除く)選択された任意のノードから見た兄弟要素の最後にコメントノードを追加します。属性ノードが選択されている場合、属性の親要素から見た子要素の最後にコメントノードが追加されます。

28.4.2.6 XML の追加

「XML | 追加 | XML」コマンドは、新規 XML 宣言 `<?xml version="1.0" encoding="UTF-8"?>` を、ドキュメント最初のアイテムとして挿入します。

メモ: XML ドキュメントに含めることのできる XML 宣言は1つだけで、ファイルの最初に記述されなければなりません。

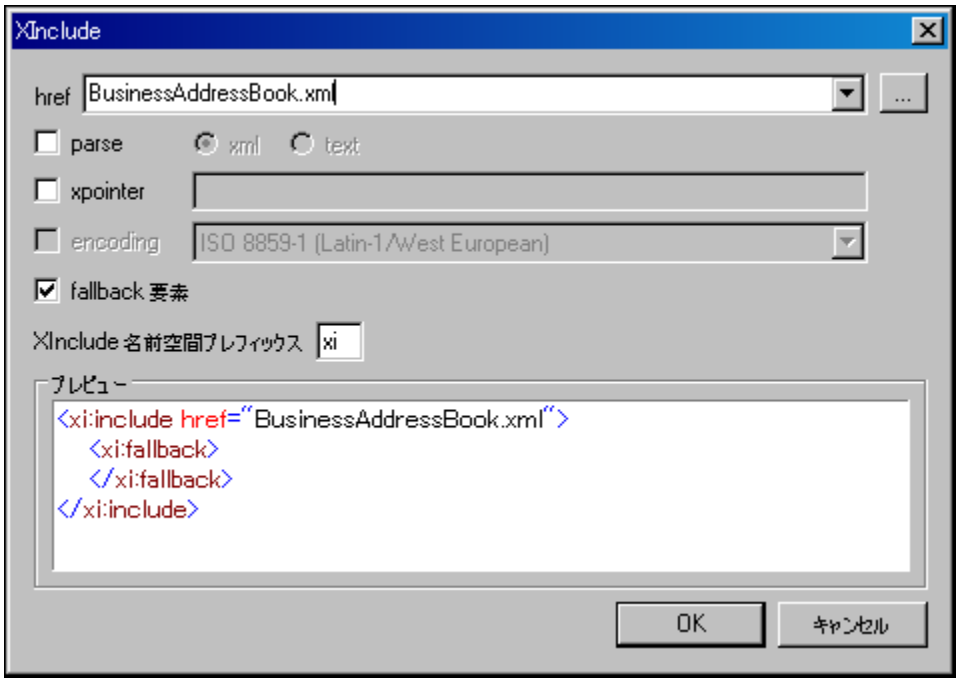
(C) 2015-2021 Altova GmbH

28.4.2.7 処理命令の追加

「XML | 追加 | 処理命令」コマンドはグリッドビューでのみ使用でき、(属性ノードを除く)選択された任意のノードから見た兄弟要素の最後に処理命令ノードを追加します。属性ノードが選択されている場合、属性の親要素から見た子要素の最後に処理命令ノードが追加されます。

28.4.2.8 XInclude の追加

「XML | 追加 | XInclude」コマンドはグリッドビューでのみ使用でき、(属性ノードを除く)選択された任意のノードから見た兄弟要素の最後にXInclude 要素を追加します。属性ノードが選択されている場合、属性の親要素から見た子要素の最後にXInclude 要素が追加されます。このコマンドによりXInclude ダイアログが表示されます(以下のスクリーンショットを参照)。



インクルードされるXML ファイルをhref テキストボックスに入力します(または、テキストボックスの右側にある... (参照)ボタンをクリックすることで、ファイルの参照を行うこともできます)。ファイル名がhref 属性の値として、XML ドキュメント内に入力されます。XInclude (xi:include) 要素のparse、xpointer やencoding 属性、そしてxi:include のfallback 子要素も、このダイアログにて挿入することができます。対応するチェックボックスにチェックを入れ、必要な値を選択/入力してください。fallback 要素の場合は、チェックボックスにチェックを入れることで、空要素を挿入することができます。fallback 要素のコンテンツは、編集ビューにて追加する必要があります。

parse 属性により、インクルードされるドキュメントがXML またはテキストとして解析（パース）されるかが決定されます（デフォルトではXML という値がセットされており、指定を行う必要はありません）。xpointer 属性は、href 属性により指定されるドキュメント内フラグメント（つまり、インクルードされるフラグメント）を特定します。encoding 属性ではインクルードされるドキュメントのエンコーディングが指定され、XMLSpy により、インクルードするドキュメントの内容がインクルード元のドキュメントエンコーディングに合う形で変換されます。インクルードされるドキュメントが見つからない場合は、xi:include 要素がfallback 子要素により置換えられます。

XInclude により2つのXML ドキュメントをインクルードするXML ドキュメントのサンプルを以下に示します:

```
<?xml version="1.0" encoding="UTF-16"?>
<AddressBook xsi:schemaLocation="http://www.altova.com/sv/myaddresses AddressBook.xsd"
       xmlns="http://www.altova.com/stylevision/tutorials/myaddresses"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="BusinessAddressBook.xml"/>
       <xi:include href="PersonalAddressBook.xml"/>
</AddressBook>
```

XML ドキュメントがパースされると、上の例で2つあるXInclude 要素が、href 属性により指定されたファイルの内容に置換えられます。

xml:base

XMLSpy のXML バリデーターによりXML ドキュメントが読み取られ、XInclude 名前空間内のinclude 要素（これ以降はxi:include と表記）が発見されると、この要素（xi:include）を、xi:include 要素のhref 属性により指定されたXML ドキュメントに置換えます。インクルードされたXML ドキュメントのドキュメント要素（ルート要素）や、XPointer により指定された要素にもxml:base 属性は含まれ、インクルードされたベースURI を保持します。（インクルードされたXML やソーフラグメントを含む）結果 XML ドキュメントが、スキーマに従う形で妥当でなければならない場合、インクルードされたドキュメントのドキュメント要素（ルート要素）やソーフラグメントのトップレベル要素を、xml:base 属性を許可するコンテンツモデルにより作成する必要があります。スキーマにより、xml:base 属性がこの要素に対して認められない場合、ドキュメントは妥当でないと評価されます。XMLSpy のスキーマビューにて要素コンテンツモデルのxml:base 属性を定義する方法は、ドキュメンテーションのスキーマビューセクション、xml: プレフィックス属性 228 にて記述されます。

XPointers

XMLSpy ではXInclude 内におけるXPointer がサポートされます。関連するW 3C 勧告は、XPointer Framework ならびにXPointer element() スキーマの2つです。XInclude 要素内のXPointer により、XML ドキュメント全体の代わりに、XML ドキュメント内部でインクルードする箇所を特定することができます。XPointer は、以下にあるように、XInclude 要素内部で使用されます::

```
<xi:include href="PersonalAddressBook.xml" xpointer="element(usa)"/>
<xi:include href="BusinessAddressBook.xml" xpointer="element(/1/1)"/>
<xi:include href="BobsAddressBook.xml" xpointer="element(usa/3/1)"/>
<xi:include href="PatsAddressBook.xml" xpointer="element(usa)element(/1/1)"/>
```

XPointer のelement() スキーマでは、NCName や子シーケンスを使用することができます。

- 上の例にある最初のxi:include 要素のxpointer 属性ではusa というNCName を伴った要素スキーマが使用されています。XPointer Framework に従うと、このNCName によりusa というID を持つ要素が特定されます。
- 2つ目のxi:include では、element(/1/1) という値がxpointer 属性で使用されています。最初のステップでは、ドキュメントにある最初の子要素（つまりドキュメントが整形式であれば、ドキュメント要素）が特定されます。次のステップでは、最初のステップで特定された要素から見た最初の子要素が特定され、結果としてドキュメント要素にある最初の子要素がインクルードされます。
- 3番目のxi:include にあるxpointer 属性では、NCName と子シーケンスの両方が使用されています。XPointer により、usa というID を持つ要素の3番目の子要素、その子要素から見た最初の子要素がインクルードされます。.
- 最初に使用したXPointer が期待通りに動作するか分からない場合、4番目の例として示されているxpointer="element(usa) element(/1/1)" という書式を使用することもできます。この方法では、たとえ usa というID を持つ要素が存在しない場合でも、ドキュメント要素の最初の子要素が選択されます。これ以外にもバックアップとなる式を追加することができます。各 XPointer は、空白スペース

以外で分けることはできず、以下のような書式に従う必要があります: xpointer="element(usa)element(addresses/1) element(/1/1)"。

メモ: element() スキーマはqualified name でサポートされておらず、名前空間バインディングコンテキストは、element() スキーマにて使用されません。

28.4.2.9 DOCTYPE の追加

「XML | 追加 | DOCTYPE」コマンドはXML ファイルを開いているグリッドビューで、最上位のノードが選択されている時のみ使用することができます。このコマンドによりDOCTYPE 宣言がXML ドキュメントの上位に追加されます。DOCTYPE の名前を入力する必要がありますが、名前はドキュメント要素のそれと同一でなければなりません。

DOCTYPE の名前を入力すると、DTD 内部サブセットで使用する宣言を入力することができます。

メモ:

- DOCTYPE 宣言は XML 宣言と XML ドキュメント要素の間にしか記述することができません。
- 代わりに DTD の割り当て[1243] コマンドを使用することで、外部 DTD ドキュメントを参照するDOCTYPE ステートメントを作成することもできます。

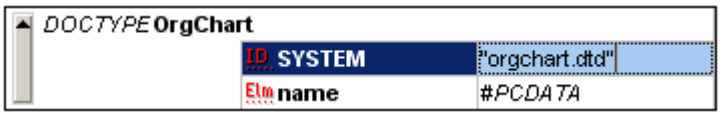
28.4.2.10 ExternalID の追加

XML ファイル内のDOCTYPE 宣言には、DTD 宣言を含んでいる外部リソースに対する参照を含めることができます。このリソースはPUBLIC またはSYSTEM 識別子により参照することができます。以下にその例を示します:

```
<!DOCTYPE doc_element_name PUBLIC "publicID" "systemID">
<!DOCTYPE doc_element_name SYSTEM "systemID">
```

SYSTEM 識別子は、外部リソースを識別するURI です。PUBLIC 識別子は場所に依存するものではなく、外部リソースの場所を相互参照するために使用することができます。例えば、XMLSpy XMLSpy がインストールされると、広く使用されているDTD やXML スキーマが、catalog.xml という名前のファイル内に記述され、その実体が`C:\Program Files\Altova\Common2021\Schemas\`フォルダー以下に収められることになります。XML ドキュメント内のPUBLIC 識別子は、これらのファイル内に記述されているDTD の相互参照に使用することができます。

グリッドビューにてXML ファイル内のDOCTYPE アイテムの「子」アイテムが選択されていると、「XML | 挿入 | ExternalID」コマンドが有効になります。このコマンドにより、外部識別子(PUBLIC またはSYSTEM)のグリッドビュー行が挿入されます。識別子の種類とその値を入力してください。



上に示されるスクリーンショットのグリッドビューに対応したテキストビューは、以下のようになります:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

メモ: ExternalID の行は、選択されているDOCTYPE アイテムの子として追加したり、DOCTYPE アイテムの子アイテム(例: 上の例におけるname というELEMENT 宣言)が選択されているときに挿入や追加することができます。

28.4.2.11 ELEMENT の追加

「XML | 追加 | ELEMENT」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、宣言のリストにELEMENT 宣言が追加されます。

28.4.2.12 ATTLIST の追加

「XML | 追加 | ATTLIST」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、宣言のリストにATTLIST 宣言が追加されます。

28.4.2.13 ENTITY の追加

「XML | 追加 | ENTITY」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、宣言のリストにENTITY 宣言が追加されます。

(C) 2015-2021 Altova GmbH

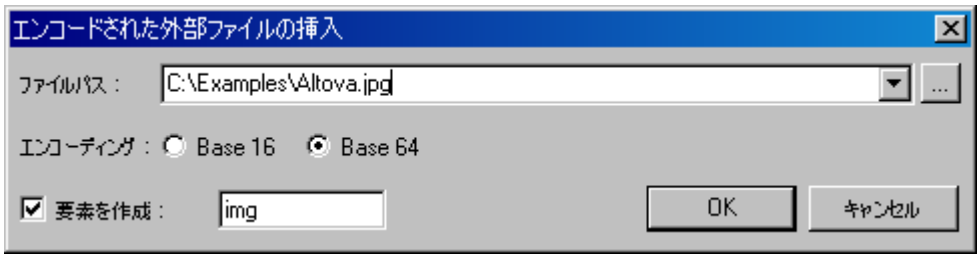
28.4.2.14 NOTATION の追加

「XML | 追加 | NOTATION」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されているときだけ選択可能になります。このコマンドにより、宣言のリストにNOTATION 宣言が追加されます。

28.4.2.15 エンコードされた外部ファイル

「XML | 追加 | エンコードされた外部ファイル」コマンドはグリッドビューでのみ使用できます。イメージファイルなどのバイナリファイルを、エンコードされた文字として追加することができます。エンコードされた外部ファイルがグリッドビューにおける選択対象の前に追加されます。

コマンドをクリックすると、エンコードされた外部ファイルの挿入ダイアログが表示されます(*下のスクリーンショットを参照*) 。ファイルへのパスを入力し、必要とするエンコードを選択し、エンコードされたファイルが要素内に挿入されるかを指定します。



エンコードならびに埋め込みが行われる外部ファイルの参照や名前の入力を行うことができます。Base-16 またはBase-64 エンコーディングを指定する必要があります。エンコードされたテキストを要素内に収めるには、要素の作成チェックボックスにチェックを入れて、要素の作成テキストボックスに使用する要素の名前を入力してください。

「OK」をクリックすることで選択されたファイルのエンコードされたテキストがカーソル位置に指定されている場合囲まれた要素と共に挿入されます。

グリッドビュー内でエンコードされたファイルが追加されます。

28.4.3 子の追加

「XML | 子の追加」コマンドは、(全てのビューで有効になってはいますが) グリッドビューでのみ使用することができます。サブメニューからは、現在選択されている要素に対して、以下にある子アイテムを追加することができます(以下のスクリーンショットを参照) :

- XML 宣言ならびにXML ドキュメント内のノード型(属性、要素、テキスト、CDATA、コメント、処理命令)
- DOCTYPE 宣言ならびにXML ドキュメント内における外部 DTD 宣言
- DTD ドキュメント内のDTD 宣言(ELEMENT、ATTLIST、ENTITY、そしてNOTATION)ならびにXML ドキュメントの DTD 宣言

28.4.3.1 属性

Ctrl+Alt+I

「XML | 子の追加 | 属性」コマンドはグリッドビューで要素ノードが選択されている時のみ使用できます。新たな属性が選択された要素ノードの子として挿入されます。

28.4.3.2 要素

Ctrl+Alt+E

「XML | 子の追加 | 要素」コマンドはグリッドビューでのみ使用できます。選択されたノードの子として新たな要素を挿入します。

(C) 2015-2021 Altova GmbH

28.4.3.3 テキスト

Ctrl+Alt+T

「XML | 子の追加 | テキスト」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの子として新たなテキストコンテンツを挿入します。

28.4.3.4 CDATA

Ctrl+Alt+D

「XML | 子の追加 | CDATA」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの子として新たな CDATA セクションを挿入します。

28.4.3.5 コメント

Ctrl+Alt+M

「XML | 子の追加 | コメント」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの子として新たなコメントノードを挿入します。

28.4.3.6 XML

「XML | 子の追加 | XML」コマンドはグリッドビューでファイルが空の時のみ使用でき、ドキュメントにおける最初のアイテムとして XML 宣言 <?xml version="1.0" encoding="UTF-8"?> を挿入します。

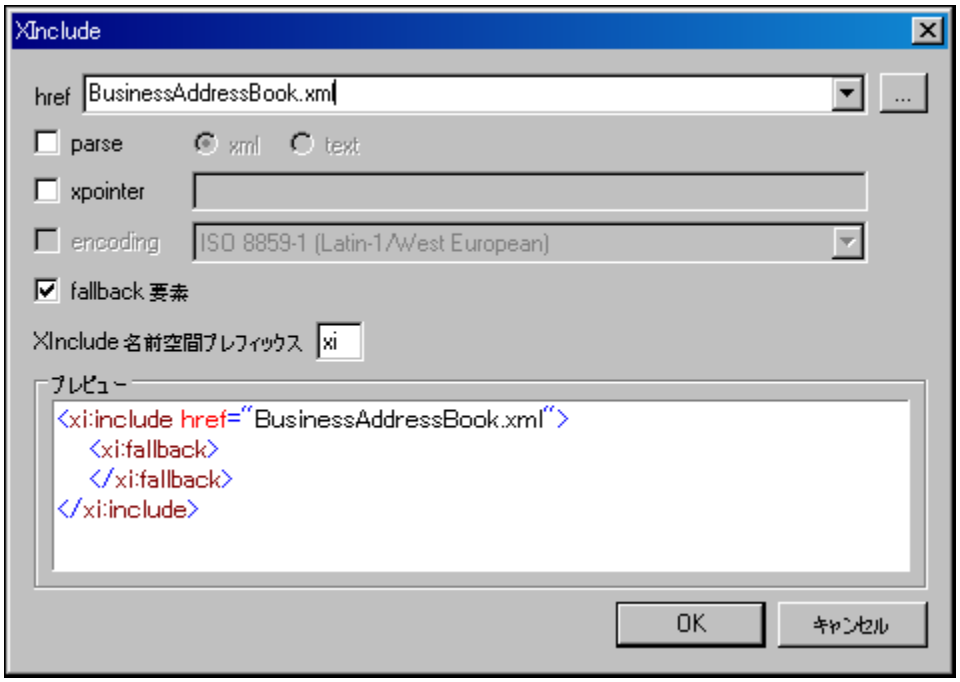
注意：XML ドキュメントに含めることのできる XML 宣言は1つだけで、ファイルの最初の方に記述されなければなりません。

28.4.3.7 処理命令

「XML | 子の追加 | 処理命令」コマンドはグリッドビューでのみ使用でき、選択されたアイテムの子として新たな処理命令（PI）を挿入します。

28.4.3.8 XInclude

「XML | 子の追加 | XInclude」コマンドはグリッドビューでのみ使用でき、選択されているアイテムの子として新たにXInclude 要素を挿入します。このコマンドによりXInclude ダイアログが表示されます（以下のスクリーンショットを参照）。



インクルードされるXML ファイルをhref テキストボックスに入力します（または、テキストボックスの右側にある... （参照）ボタンをクリックすることで、ファイルの参照を行うこともできます）。ファイル名がhref 属性の値として、XML ドキュメント内に入力されます。XInclude (xi:include) 要素のparse、xpointer やencoding 属性、そしてxi:include のfallback 子要素も、このダイアログにて挿入することができます。対応するチェックボックスにチェックを入れ、必要な値を選択/入力してください。fallback 要素の場合は、チェックボックスにチェックを入れることで、空要素を挿入することができます。fallback 要素のコンテンツは、編集ビューにて追加する必要があります。

parse 属性により、インクルードされるドキュメントがXML またはテキストとして解析（パース）されるかが決定されます（デフォルトではXML という値がセットされており、指定を行う必要はありません）。xpointer 属性は、href 属性により指定されるドキュメント内フラグメント（つま

り、インクルードされるフラグメント）を特定します。encoding 属性ではインクルードされるドキュメントのエンコーディングが指定され、XMLSpy により、インクルードするドキュメントの内容がインクルード元のドキュメントエンコーディングに合う形で変換されます。インクルードされるドキュメントが見つからない場合は、xi:include 要素がfallback 子要素により置換えられます。

XInclude により2つのXMLドキュメントをインクルードするXMLドキュメントのサンプルを以下に示します:

```
<?xml version="1.0" encoding="UTF-16"?>
<AddressBook xsi:schemaLocation="http://www.altova.com/sv/myaddresses AddressBook.xsd"
       xmlns="http://www.altova.com/stylevision/tutorials/myaddresses"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xi="http://www.w3.org/2001/XInclude">
       <xi:include href="BusinessAddressBook.xml"/>
       <xi:include href="PersonalAddressBook.xml"/>
</AddressBook>
```

XMLドキュメントがパースされると、上の例で2つあるXInclude 要素が、href 属性により指定されたファイルの内容に置換えられます。

xml:base

XMLSpy のXMLバリデーターによりXMLドキュメントが読み取られ、XInclude 名前空間内のinclude 要素（これ以降はxi:include と表記）が発見されると、この要素（xi:include）を、xi:include 要素のhref 属性により指定されたXMLドキュメントに置換えます。インクルードされたXMLドキュメントのドキュメント要素（ルート要素）や、XPointer により指定された要素にもxml:base 属性は含まれ、インクルードされたベースURI を保持します。（インクルードされたXML やツリーフラグメントを含む）結果XMLドキュメントが、スキーマに従う形で妥当でなければならない場合、インクルードされたドキュメントのドキュメント要素（ルート要素）やツリーフラグメントのトップレベル要素を、xml:base 属性を許可するコンテンツモデルにより作成する必要があります。スキーマにより、xml:base 属性がこの要素に対して認められない場合、ドキュメントは妥当でないと評価されます。XMLSpy のスキーマビューにて要素コンテンツモデルのxml:base 属性を定義する方法は、ドキュメンテーションのスキーマビューセクション、xml: プレフィックス属性 228 にて記述されます。

XPointers

XMLSpy ではXInclude 内におけるXPointer がサポートされます。関連するW 3C 勧告は、XPointer Framework ならびにXPointer element() スキーマの2つです。XInclude 要素内のXPointer により、XMLドキュメント全体の代わりに、XMLドキュメント内部でインクルードする箇所を特定することができます。XPointer は、以下にあるように、XInclude 要素内部で使用されます::

```
<xi:include href="PersonalAddressBook.xml" xpointer="element(usa)"/>
<xi:include href="BusinessAddressBook.xml" xpointer="element(/1/1)"/>
<xi:include href="BobsAddressBook.xml" xpointer="element(usa/3/1)"/>
<xi:include href="PatsAddressBook.xml" xpointer="element(usa)element(/1/1)"/>
```

XPointer のelement() スキーマでは、NCName や子シーケンスを使用することができます。

- 上の例にある最初のxi:include 要素のxpointer 属性ではusa というNCName を伴った要素スキーマが使用されています。XPointer Framework に従うと、このNCName によりusa というID を持つ要素が特定されます。
- 2つ目のxi:include では、element(/1/1) という値がxpointer 属性で使用されています。最初のステップでは、ドキュメントにある最初の子要素（つまりドキュメントが整形式であれば、ドキュメント要素）が特定されます。次のステップでは、最初のステップで特定された要素から見た最初の子要素が特定され、結果としてドキュメント要素にある最初の子要素がインクルードされます。
- 3番目のxi:include にあるxpointer 属性では、NCName と子シーケンスの両方が使用されています。XPointer により、usa というID を持つ要素の3番目の子要素、その子要素から見た最初の子要素がインクルードされます。.
- 最初に使用したXPointer が期待通りに動作するか分からない場合、4番目の例として示されているxpointer="element(usa) element(/1/1)" という書式を使用することもできます。この方法では、たとえ usa というID を持つ要素が存在しない場合でも、ドキュメント要素の最初の子要素が選択されます。これ以外にもバックアップとなる式を追加することができます。各 XPointer は、空白スペース以外で分けることはできず、以下のような書式に従う必要があります: xpointer="element(usa)element(addresses/1) element(/1/1)"。

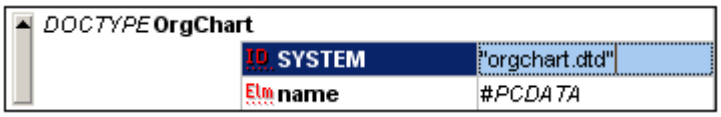
メモ: element() スキーマはqualified name でサポートされておらず、名前空間バインディングコンテキストは、element() スキーマにて使用されません。

28.4.3.9 DOCTYPE

「XML | 子の追加 | DOCTYPE」コマンドは空のドキュメントを表示しているグリッドビューでのみ使用できます。このコマンドによりXMLドキュメント内にDOCTYPE 宣言が挿入されます。DOCTYPE 宣言は内部 DTD サブセットの宣言に使用することができます。

28.4.3.10 ExternalID

「XML | 子の追加 | ExternalID」コマンドはXML ファイルのDOCTYPE 宣言がグリッドビューで選択されている時のみ使用することができます。このコマンドにより、外部識別子(PUBLIC または SYSTEM)のグリッドビュー行が挿入されます。識別子の種類とその値を入力してください。



上に示されるスクリーンショットのグリッドビューに対応したテキストビューは、以下のようになります:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

メモ: External-ID の行は、選択されているDOCTYPE アイテムの子として追加したり、DOCTYPE アイテムの子アイテム(例: 上の例における name というELEMENT 宣言)が選択されているときに挿入や追加することができます。

28.4.3.11 ELEMENT

「XML | 子の追加 | ELEMENT」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されている時だけ選択可能になります。このコマンドにより、宣言のリストにELEMENT 宣言が追加されます。

28.4.3.12 ATTLIST

「XML | 子の追加 | ATTLIST」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されている時だけ選択可能になります。このコマンドにより、宣言のリストにATTLIST 宣言が追加されます。

28.4.3.13 ENTITY

「XML | 子の追加 | ENTITY」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されている時だけ選択可能になります。このコマンドにより、宣言のリストにENTITY 宣言が追加されます。

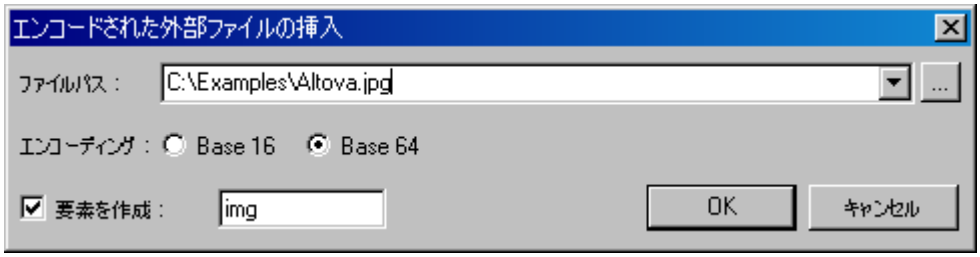
28.4.3.14 NOTATION

「XML | 子の追加 | NOTATION」コマンドはグリッドビューでのみ使用でき、DTD ドキュメントやXML ドキュメントのDOCTYPE 宣言が選択されている時だけ選択可能になります。このコマンドにより、宣言のリストにNOTATION 宣言が追加されます。

28.4.3.15 エンコードされた外部ファイル

「XML | 子の追加 | エンコードされた外部ファイル」コマンドはグリッドビューでのみ使用でき、イメージファイルといったバイナリファイルを、エンコードされた文字として追加することができます。エンコードされた外部ファイルがグリッドビューにおける選択対象の子として挿入されます。

コマンドをクリックすると、エンコードされた外部ファイルの挿入ダイアログが表示されます（以下のスクリーンショットを参照）。ファイルへのパスの入力とエンコードの選択を行い、エンコードされたファイルを要素の中に含めるか選択します。



「OK」をクリックすることで、（それが指定されている場合、要素に収められたかたちで）選択されたファイルのエンコードされたテキストがカーソル位置に挿入されます。

エンコードされたファイルが、追加された子としてグリッドビューに表示されます。

28.4.4 変換

「XML | 変換」コマンドにより、グリッドビューで選択されているアイテムを他のアイテム型に変換することができます。この処理はグリッドビューにて、子ノードを持たない単一のノードに対してのみ使用することができます。「カーソルを変換」コマンドにマウスオーバーするとサブメニューが表示され、選択されたアイテムから変換することができる種類のアイテムが表示されます（以下のスクリーンショットを参照）。

メモ: 選択された処理によりデータが失われるような場合（例えば属性をコメントに変換することで、属性名が失われます）、警告ダイアログボックスが表示されます。

28.4.4.1 属性

「XML | 変換 | 属性」コマンドにより、選択されたアイテムが属性に変換されます。

28.4.4.2 要素

「XML | 変換 | 要素」コマンドにより、選択されたアイテムが要素に変換されます。

28.4.4.3 テキスト

「XML | 変換 | テキスト」コマンドにより、選択されたアイテムがテキストコンテンツに変換されます。

28.4.4.4 CDATA

「XML | 変換 | CDATA」コマンドにより、選択されたアイテムがCDATA セグメントに変換されます。

28.4.4.5 コメント

「XML | 変換 | コメント」コマンドにより、選択されたアイテムがコメントに変換されます。

28.4.4.6 XML

「XML | 変換 | XML」コマンドにより、選択されたアイテムがXML 宣言に変換されます:

```
<?xml version="1.0" encoding="UTF-8"?>.
```

メモ: XML ドキュメントに含めることのできるXML 宣言は1つだけで、ファイルの最初に記述されなければなりません。

28.4.4.7 処理命令

「XML | 変換 | 処理命令」コマンドにより、選択されたアイテムが新たな処理命令（PI）に変換されます。

28.4.4.8 DOCTYPE

「XML | 変換 | DOCTYPE」コマンドにより、選択されたアイテムが(XML ファイル内の) DOCTYPE 宣言に変換されます。

メモ: DOCTYPE 宣言はXML 宣言とXML ドキュメント要素の間にしか記述することができません。

28.4.4.9 ExternalID

「XML | 変換 | ExternalID」コマンドにより、選択されたアイテムがDOCTYPE 宣言内のDTD 参照(PUBLIC またはSYSTEM 識別子)に変換されます。

28.4.4.10 ELEMENT

「XML | 変換 | ELEMENT」コマンドにより、選択されたアイテムがDOCTYPE 宣言または外部 DTD 内の要素型宣言に変換されます。

28.4.4.11 ATTLIST

「XML | 変換 | ATTLIST」コマンドにより、選択されたアイテムがDOCTYPE 宣言または外部 DTD 内の属性リスト宣言に変換されます。

28.4.4.12 ENTITY

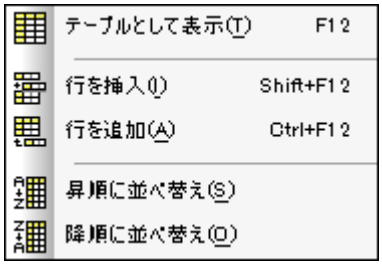
「XML | 変換 | ENTITY」コマンドにより、選択されたアイテムがDOCTYPE 宣言または外部 DTD の内エンティティ宣言に変換されます。

28.4.4.13 NOTATION

「XML | 変換 | NOTATION」コマンドにより、選択されたアイテムがDOCTYPE 宣言または外部 DTD 内の記法宣言(NOTATION) に変換されます。

28.4.5 テーブル

「**XML** | テーブル」コマンドは、(全てのビューで有効になっていますが) グリッドビューでのみ使用することができます。サブメニューから、グリッドビューの[テーブルの表示](#) 281 に関連したコマンドにアクセスすることができます。



28.4.5.1 テーブルとして表示

テーブルとして表示 (F12) メニューコマンドとツールバーアイコン

XML | **テーブルとして表示** コマンドにより、ドキュメント要素の標準的な グリッドビュー[149] と テーブルの表示[281] を切り替えることができます。テーブルビューでは、繰り返して表示される要素をテーブルとして表示することで、出現を行として、(コメントやCDATA セクション、処理命令などを含む) 子ノードをカラムとして扱うことができます。

28.4.5.2 行を挿入

Shift + F12

行またはセルが選択されると **行の挿入** コマンドは テーブルの表示[281] 内で有効化されます。このコマンドにより、選択されている行の前に新たな行が挿入されます。新たな行はテーブル要素の出現に対応します。必須となっている子要素が新たな要素に対して作成されます

28.4.5.3 行を追加

Ctrl+F12

行またはセルが選択されると **行の付加** コマンドは テーブルの表示[281] 内で有効化されます。これによりテーブルの最後に新たな行が挿入されます。新たな行は要素の出現に対応します。。必須となっている子要素が新たな要素に対して作成されます

28.4.5.4 昇順に並べ替え

行または セルが選択されると 昇順に並べ替え コマンドは テーブルの表示[281] 内で有効化されます。列のデータ型に従い 列をアルファベット順または数値順に昇順に並べ替えられます。

28.4.5.5 降順に並べ替え

行または セルが選択されると 降順に並べ替え コマンドは テーブルの表示[281] 内で有効化されます。列のデータ型に従い 列をアルファベット順または数値順に降順に並べ替えられます。

28.4.6 左へ移動

Ctrl+L

「XML | 左へ移動」コマンドはグリッドビューでのみ使用することができます。このコマンドにより、選択されたノードが 1 つ左のレベルに移動され、親要素から見た子要素が兄弟要素になります。このコマンドは、「レベル上げ」コマンドと呼ばれることもあります。

28.4.7 右へ移動

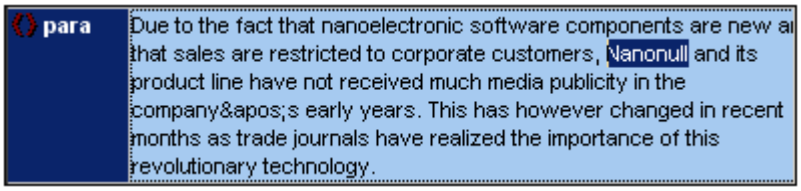
Ctrl+R

「XML | 右へ移動」コマンドはグリッドビューでのみ使用することができます。このコマンドにより、選択されたノードが 1 つ右のレベルに移動され、ある要素が、1 つ前の要素を親とする子要素になります。このコマンドは、「レベル下げ」コマンドと呼ばれることもあります。

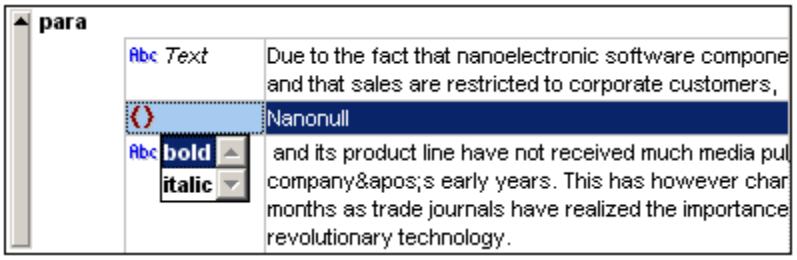
28.4.8 要素の中へ

「XML | 要素の中へ」コマンドは、グリッドビューでのみ使用することができます。このコマンドにより、選択されたテキスト範囲を新たな要素の中に収めることができます。新たな要素が選択されたテキストの周りに作成されます。スキーマや DTD をベースにしたドキュメントを編集している場合、テキストを収めることができる妥当な要素名のリストが表示されます。

例えば、以下の例では para 要素内にある Nnonull というテキストがハイライトされています。



「XML | 要素の中へ」コマンドを選択すると、テキスト Nanonull が新たに作成されたインライン要素に収められ、要素の名前に使用される選択しとして bold と italic が表示されます。これらの要素は para の子としてスキーマでは定義されています。



ここで選択した結果が、新たに作成される要素の名前になります。それ以外にも要素に他の名前を入力することもできます。

28.4.9 XPath を評価

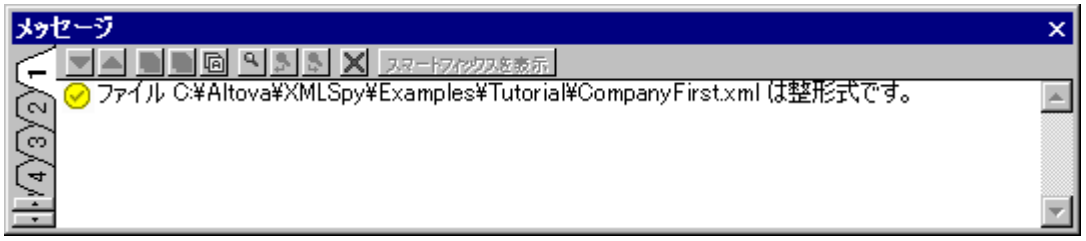
「XML | XPath を評価」コマンドにより、(出力ウィンドウが閉じられている場合は) 出力ウィンドウが開かれ、出力ウィンドウの XPath タブ[113]が有効になります。XPath タブでは、アクティブなドキュメントに対して XPath 条件式を評価することができ、その結果が出力ウィンドウに表示されます。

28.4.10 整形式のチェック

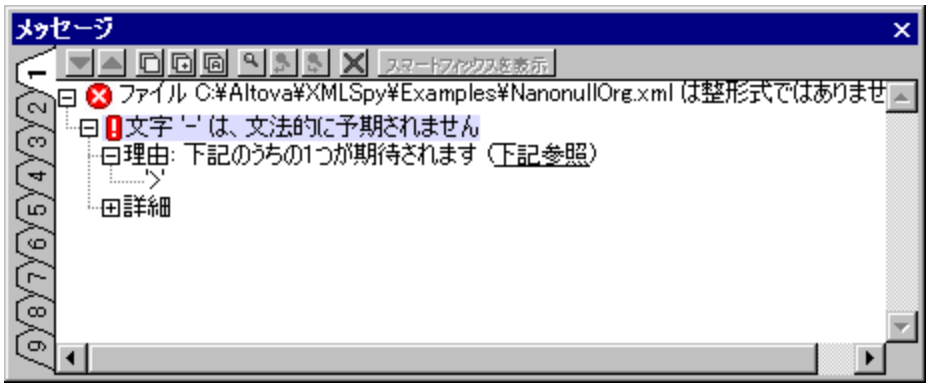
F7

「XML | 整形式のチェック(F7)」コマンドにより、XML 1.0 仕様の定義に従い、アクティブなドキュメントの整形式をチェックすることができます。全ての XML ドキュメントは整形式でなければなりません。XMLSpy では、ドキュメントが開かれたときと保存します。またはテキストビューから他のビューへの切り替えが行われた時に整形式がチェックされます。このコマンドを使用することで、編集中はいつでも整形式のチェックを行うことができます。

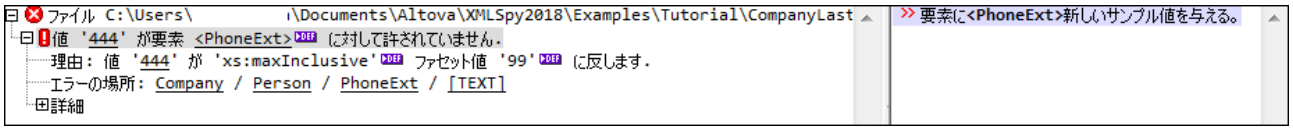
整形式のチェックに成功すると、その旨のメッセージがメッセージウィンドウに表示されます(以下のスクリーンショットを参照)。



整形式チェックの結果、エラーが見つかった場合、エラーメッセージが表示されます(以下のスクリーンショットを参照):



テキストビュー または グリッドビュー 内で整形式のチェックが行われた場合、スキーマ内の情報をベースにして、スマートフィックスのためのオプションが提案されます。スマートフィックス オプションのリストを閲覧するには、**スマートフィックスを表示** ボタンをクリックします(*上のスクリーンショット参照*)。スマートフィックスオプションを表示するペインがメッセージウィンドウ内に表示されます(*下のスクリーンショットを参照*)。メッセージウィンドウ内のエラーでは、1度に1つのエラーが表示されます。



スマートフィックス ペイン内で、「**フィックス + 検証**」ボタン または「**フィックス**」ボタンをクリックして提案されたスマートフィックスから選択します(*上のスクリーンショット参照*)。エラーは、スマートフィックスを使用して修正されます。または、スマートフィックスをダブルクリックすることもできます。このアクションは、ドロップダウンオプションリスト内で選択されたオプションに従い、エラーをフィックスするか、または、エラーをフィックスしドキュメントを検証します。「**フィックス + 検証**」コマンドは、固定されたエラー以外にも次のエラーが存在する場合はそのエラーをフィックスするため、とても役に立ちます。

スマートフィックス ペインを非表示にするには、「**スマートフィックスの非表示**」ボタンをクリックします(*上のスクリーンショット参照*)。

メモ メッセージウィンドウには9つのタブがあります。整形式チェックの結果はアクティブなタブに表示されます。そのため、タブ1にて整形式のチェックを行った後、タブ2に切り替えて、整形式チェックの結果を保持したまま、別のドキュメント検証を行うといったことができます(タブが切り替えられない場合、最初のタブに出力された情報が上書きされます)。

プロジェクト ウィンドウから検証を行う

[**検証**] コマンドをアクティブなプロジェクト内のファイル、フォルダー、ファイルのグループに適用することができます。(クリックして)プロジェクト ウィンドウ内で必要とされるファイルまたはフォルダーを選択します。[XML | Validate XMLの検証 1227]をクリックします。サーバー上のXMLの検証(ハイパフォーマンス) 1231 コマンドを使用することもできます。プロジェクト内の無効なファイルが開かれ、メインウィンドウ内でアクティブ化され、*ファイルが有効ではない ことを知らせる*エラーメッセージが表示されます。

メモ メッセージウィンドウには9つのタブがあります。整形式チェックの結果はアクティブなタブに表示されます。そのため、タブ1にて整形式のチェックを行った後、タブ2に切り替えて、整形式チェックの結果を保持したまま、別のドキュメント検証を行うといったことができます(タブが切り替えられない場合、最初のタブに出力された情報が上書きされます)。

通常問題のあるXMLドキュメントを保存することは許されていませんが、XMLSpy ではそれでも保存を行うことができます。作業を行なっているドキュメントの保存を(整形式で無い状態で)行い、後で作業を再開するような状況で、この機能を利用することができます。

メモ 整形式のチェックコマンドは、プロジェクトウィンドウ[110]でアクティブなファイルやフォルダー、ファイルのグループに対しても使用することができます。目的のアイテムをクリックし、整形式のチェックアイコンをクリックしてください。

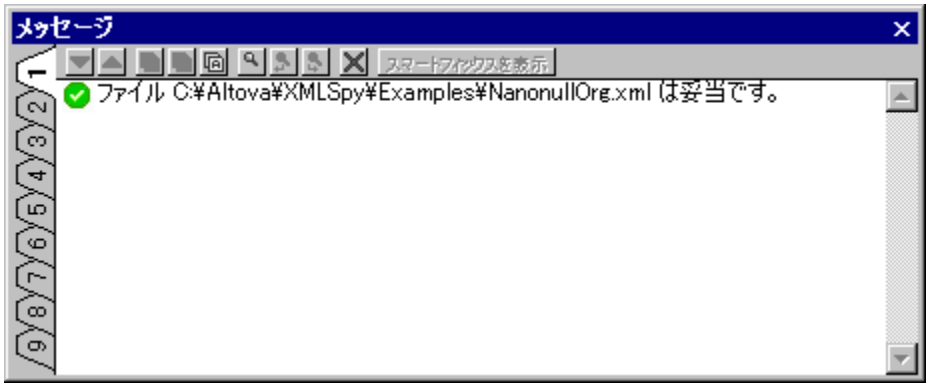
28.4.11 XML の検証

F8

「XML 検証(F8)」コマンドにより、DTD やXML スキーマ、またはその他のスキーマに対してXMLドキュメントの検証を行うことができます。テキストビューから他のビューへの切り替えが行われた際にも、検証は自動的に行われます。ファイルが開かれたり保存されたときに、自動的にドキュメントの検証が行われるよう指定することもできます(「ツール | オプション | ファイル」)。「検証」コマンドでは、検証が行われる前に整形式のチェックも行われます。そのため、「検証」コマンドを使用する前に整形式のチェック[1225]を行う必要はありません。

メモ データを編集中に検証するために編集中に検証[1233] コマンドをオンに切り替えることができます。

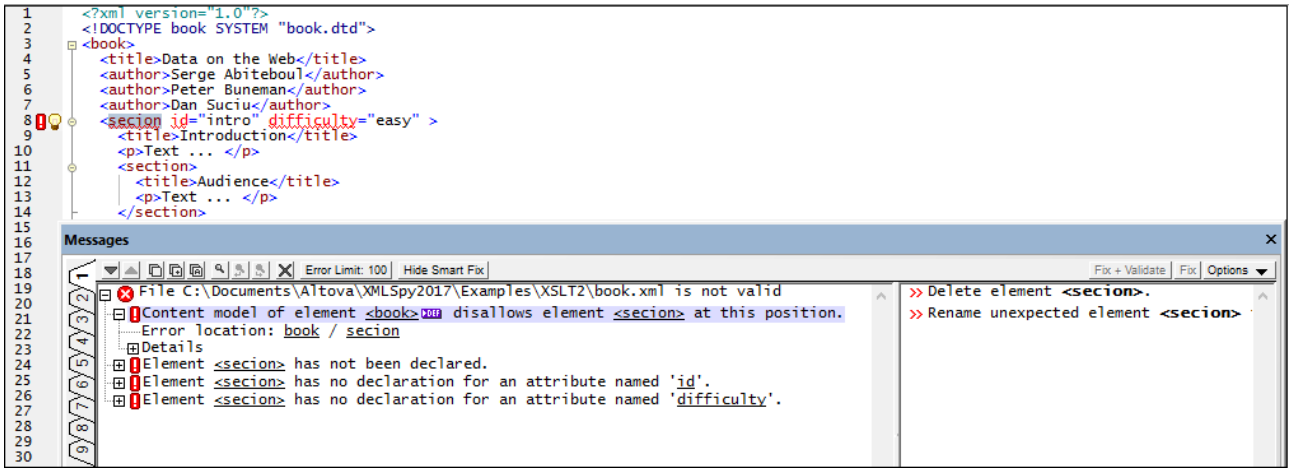
ドキュメントが有効な場合、メッセージウィンドウ内に検証の成功メッセージが表示されます。



そうでない場合、エラーの内容がメッセージウィンドウに表示されます(以下のスクリーンショットを参照)。リンクをクリックすることで、XML ファイル内のエラーが発見された箇所へジャンプすることができます。エラーメッセージの詳細と、XMLSpy のスマートフィックスを使用して検証エラーをフィックスする方法に関しては次のセクションを参照してください。

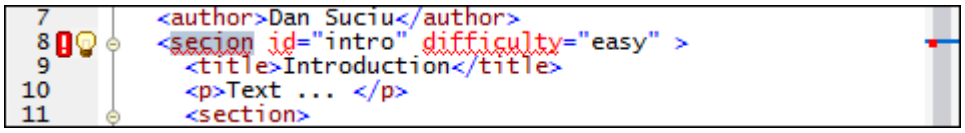
検証エラーとそのフィックス

「テキストビュー」または「グリッドビュー」にて検証が行われると、スキーマ内にある情報をベースに、修正の候補(スマートフィックス)が表示されます。スマートフィックスのオプションを参照するには、「スマートフィックスを表示」ボタンをクリックしてください(上のスクリーンショットを参照)。スマートフィックスの候補がメッセージウィンドウ内のペインに表示されます(以下のスクリーンショットを参照)。これらの提案を表示しない場合、「スマートフィックスを非表示」をクリックします。(一致しない開始と終了タグなどの)整形式のエラーが存在する場合、これらのエラーは検証エラーの前に表示されます。ですから(整形式 エラーがすべて修正された後)「表示/非表示 スマートフィックス」ボタンは検証エラーが発生すると有効化されます。

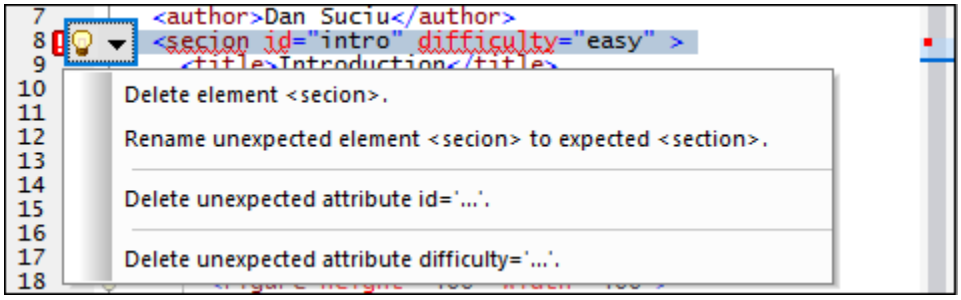


スマートフィックスを適用するには、以下を行います(i) ダブルクリックする(ii)「フィックス」または「フィックス + 検証」オプションを選択してをクリックします(*下のスクリーンショットを参照*)。「フィックス + 検証」コマンドは、固定されたエラー以外にも次のエラーが存在する場合はそのエラーをフィックスします。

テキストビュー内では、2つの検証エラーの追加インジケーターが存在します(*下のスクリーンショットを参照*): (i) 行番号マージン内の赤い感嘆符 アイコン、および、(ii) (ウィンドウの右の)スクロールバー内の赤い四角いマーカー。



感嘆符 アイコンの横の電球 アイコンは、スマートフィックスアイコンです(*上のスクリーンショット参照*)。マウスをポイントすると、エラーの原因の全てのスマートフィックスが表示されます(*下のスクリーンショットを参照*)。適用するスマートフィックスを選択します。



メモ : 出力ウィンドウには9つのタブがあります。検証の結果はアクティブなタブに表示されます。そのため、最初のタブにて整形式のチェックを行った後、2つ目のタブに切り替えて、別のドキュメント検証を行うということができます。2番目のドキュメントを検証するために、チェックを実行する前に、タブ2（またはタブ3）に切り替えます。タブを切り替えない場合、タブ1（またはアクティブなタブ）が最新の検証のフォーミュラにより上書きされます。

プロジェクトウィンドウからの検証

検証コマンドはアクティブなプロジェクト内部にあるファイル、フォルダー、またはファイルのグループに対しても適用することができます。目的のファイルやフォルダーをプロジェクトウィンドウにて（クリックにより）選択し、メニューオプションから「XML | XML 検証」を選択するか、「F8」を押下してください。プロジェクト内の妥当ではないファイルがメインウィンドウに表示され、ファイルが妥当でない旨のメッセージが表示されます。

XML ドキュメントの検証

XML ファイルを検証するには、XML ドキュメントをメインウィンドウにてアクティブな状態にした後、「XML | 検証」をクリックするか、「F8」キーを押下します。XML ファイルから参照されているスキーマに対して XML ドキュメントが検証されます。参照が無い場合、エラーメッセージがメッセージウィンドウに表示されます。XML ドキュメントが開かれている場合、スキーマもメモリ内に保管されます（DTD/スキーマメニュー以下にある フラッシュメモリキャッシュ 1255 を参照ください）。

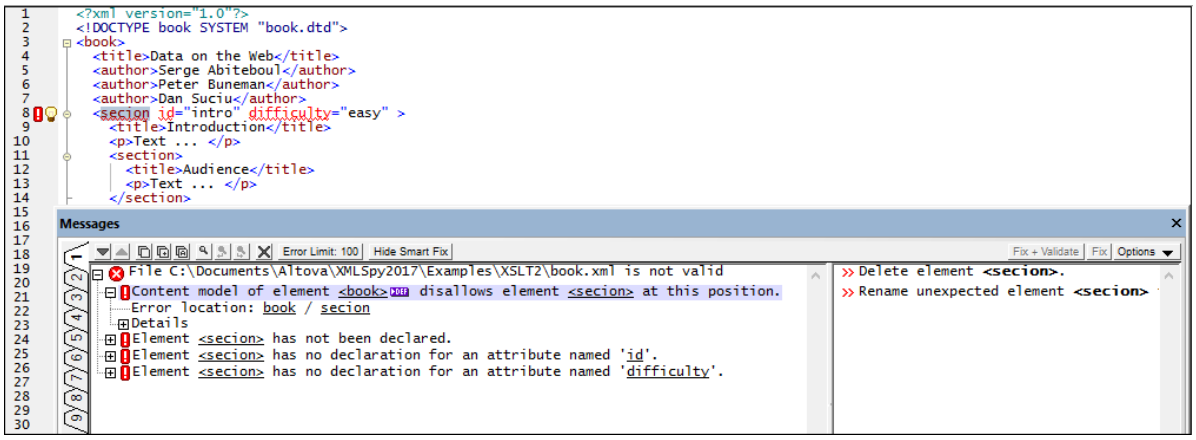
スキーマドキュメンテーション（DTD とXML スキーマ）の検証

XMLSpy では、DTD やXML スキーマといった広く使用されているスキーマがサポートされます。スキーマドキュメントの検証を行うには、ドキュメントをメインウィンドウにてアクティブにして、「XML | 検証」をクリックするか、「F8」 キーを押下します。

検証メッセージ

メッセージには以下の2種類があります:

- スキーマ（DTD または XML スキーマ）が妥当な場合、検証に成功した旨のメッセージがメッセージウィンドウに表示されます。
- スキーマが妥当で無い場合、エラーメッセージがメッセージウィンドウに表示されます（以下のスクリーンショットを参照）。



エラーメッセージは3つのパートから構成されます：例えば、上のスクリーンショットでは、検証エラーを引き起こした可能性のある4つの理由についての詳細が報告されています。最初の理由は展開されており、3つの理由は折りたたまれています。それぞれの理由は3つの部分に分割されています:

1. エラーの説明：説明には関連する宣言や定義へのリンクが含まれます。特定のスキーマ定義に移動し、ドキュメントが無効な理由を確認することができます。
2. エラーの原因と、エラーが発生したノードへのフルパス：パス内のノードをクリックすることで、ドキュメント内の該当するノードがハイライトされます。
3. エラーの詳細：更に詳細なエラーの情報と、該当する仕様の関連する段落へのリンクが含まれています。

メモ 検証がテキストビューで行われると、メッセージウィンドウのリンクをクリックすることで、関連した宣言や定義がテキストビューにてハイライトされます。検証がスキーマビューで行われると、関連する宣言や定義が、デザインウィンドウやスキーマビュー、または関連する入力ヘルパーにてハイライトされます。

カタログ

XMLSpy ではOASIS XML カタログ機能のサブセットがサポートされます。カタログ機能により、一般的に使われているスキーマ(や、スタイルシートなどのファイル)が、XMLSpy によりローカルのユーザーフォルダーから取得されます。これにより全体的な処理スピードが向上し、(ネットワークに接続していない状態でも)ユーザーはオフラインで作業することができるようになり、(URI の変更はカタログファイル内部に限定されるので)ドキュメントのポータビリティーが向上します。XMLSpy のカタログ機能は以下のように動作します:

- XMLSpy により、検索を行うカタログファイルのリストが含まれる RootCatalog.xml というファイルがロードされます。RootCatalog.xml ファイルにて nextCatalog 要素を加えることで、検索を行ないたいファイルを増やすこともできます。
- RootCatalog.xml に収められたカタログファイルが検索され、カタログファイルで指定されたマッピングにより、URI が解決されます。エラーを起こす可能性があるため、マッピングの複製を行わないよう注意する必要があります。
- XMLSpy では2つのカタログファイルが用意されています。これらの動作方法については XMLSpy のカタログ 409 セクションを参照ください。
- XML ファイルの DOCTYPE ステートメントにおける PUBLIC ならびに SYSTEM 識別子はカタログの検索にて使用されます。広く使用されているスキーマでは、PUBLIC 識別子が既に定義されており、複数のコンピューターで XML ドキュメントが使用されている場合カタログファイル内部の URI を変更するだけで十分となります。

CustomCatalog.xml ファイル(または他のカスタムカタログファイル)を編集する場合、XMLSpy によるカタログが正しく行われるように、以下のサブセットだけを使用するようにしてください。サポートされているサブセットの各要素には xml:base 属性を含めることができ、その要素のベースURI を指定するのに使用することができます。

```
<catalog...>
...
```

```
<public publicId="PublicID of Resource" uri="URL of local file"/>
<system systemId="SystemID of Resource" uri="URL of local file"/>
<rewriteURI uriStartString="StartString of URI to rewrite" rewritePrefix="String to
replace StartString"/>
<rewriteSystem systemIdStartString="StartString of SystemID"
rewritePrefix="Replacement string to locate resource locally"/>
<uri name="filename" uri="URL of file identified by filename"/>
...
</catalog>
```

メモ

- %AltovaCommonFolder%\Schemas\schema[409] フォルダー内の catalog.xml ファイルには、古いバージョンの XML 仕様を実装する DTD への参照が含まれています。これらスキーマを、XML スキーマドキュメントの検証に対して使用するべきではありません。参照された DTD ファイルは、古い勧告に従ったドキュメントの編集時に使用される XMLSpy の入力ヘルパーに対してのみ利用されます。
- 特定のスキーマに対してカスタムファイル拡張を作成すると、CustomeCatalog.xml に1行追加することで、インテリジェントな編集をその拡張子を持ったファイルに対しても使用することができるようになります（例えば、.myhtml という拡張子により、HTML DTD を使った HTML ファイルの検証を行うことができます）。この例に従うと、<spy:fileExtHelper ext="myhtml" uri="schemas/xhtml/xhtml1-transitional.dtd"/> という要素を <catalog> 要素の子要素として追加することになります。この操作により、XHTML 1.0 Transitional DTD に従う形で、自動補完や入力ヘルパーといったインテリジェントな編集機能を使いながら、.myhtml ファイルの編集を XMLSpy で行うことができるようになります。
- カタログに関するより詳細な情報については、XML カタログ仕様を参照下さい。

RaptorXML 2021 を使用して検証を自動化する

RaptorXML は、XML 検証、XSLT 変換、とXQuery 変換のためのAltova のスタンドアロンアプリケーションです。Java プログラムと.NET 内のCOM インターフェイスを使用して、アプリケーションコマンドラインから使用することができます。RaptorXML を使用すると、検証のタスクを自動化することもできます。例えば、ドキュメントのセット上で検証を実行するためにRaptorXML を呼び出すバッチファイルを作成し、テキストファイルに出力を送ることができます。詳細に関しては、RaptorXML ドキュメンテーション を参照ください。

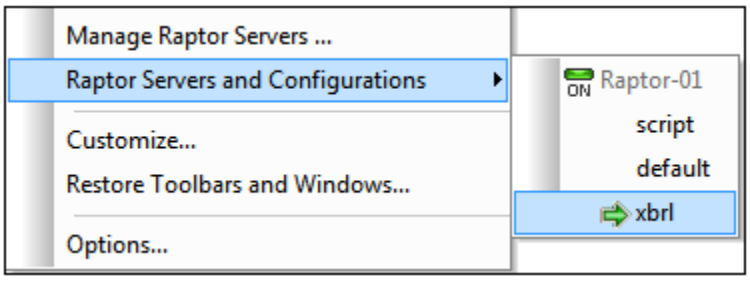
28.4.12 サーバー上の XML の検証(ハイパフォーマンス)

Ctrl+F8

XML | サーバー上の検証（ハイパフォーマンス）（Ctrl+F8）コマンドは、現在アクティブな RaptorXML Server[1438] とアクティブな 構成[1438] を使用してアクティブな XML ドキュメントを検証します。コマンドはすぐに検証を行い、メッセージウィンドウ内のフォーミュラを表示します。

メモ: 実際のパフォーマンスは、検証のためにRaptorXML Server により使用されるPC プロセッサコアにより異なります。使用されるコア数が多いほど、処理が速くなります。

複数のサーバーで複数の構成を定義した場合、サーバーとそのアクティブな構成として1つの構成を選択することができます。アクティブな構成は、以降の検証に使用されます。カーソルを「ツール | Raptor Servers および 構成」コマンドにポイントして、追加されたすべてのサーバーとその詳細が含まれるサブメニューから、を含む表示されるサブメニューが表示されます（ *下のスクリーンショット参照*）。アクティブな構成と設定するサーバー構成を選択します。したのスクリーンショットでは、`Raptor-01` という名前のサーバーの(緑色の矢印で示されている) `xbrl` 構成がアクティブな構成として選択されています。



サーバー上の XML の検証（ハイパフォーマンス）（Ctrl+F8）コマンドをプロジェクト 入力ヘルパー内で使用することができます。プロジェクト、フォルダー、または、ファイルを右クリックし、**サーバー上の XML の検証** を選択し、選択されたオブジェクト内の XML または XBRL データを検証します。

メモ: Raptor 検証はテキストビュー、グリッドビュー、および XBRL ビューで使用することができます。

28.4.13 WSDL ファイルの検証

W SDL ドキュメントは W SDL だけではなく XML でもあります。この結果 XML および W SDL として検証することができます。次のリストは XMLSpy の *Enterprise* と *Professional Editions* 内での[W SDL 検証](1227) の振る舞いに関する重要な情報が含まれています。

- *Professional Edition* は W SDL ファイルを XML ファイルとして扱い `http://schemas.xmlsoap.org/wsdl/` で定義されるスキーマに従い検証する単純なスキーマ検証を行います。
- *Enterprise Edition* では、*Professional Edition* を超えた XML 検証を行うことができます。`http://schemas.xmlsoap.org/wsdl/` に対する検証は行われません。代わりに `http://www.altova.com/specs_wsdl.html#_document-s` と独自のロジックを使用することで、W SDL の文脈に沿った更に高度な情報が提供されます。そのため、*Professional Edition* では妥当とされた W SDL ファイルが、*Enterprise Edition* では、そうでないと表示されることがあります（*以下の例を参照*）。
- `http://schemas.xmlsoap.org/wsdl/` と `http://www.altova.com/specs_wsdl.html#_document-s` には違いがあります。前者には W SDL 仕様で定義されている拡張要素（**extensibility elements**）の定義が含まれていません。この短所は公式な W 3C スキーマ内でエラーとして表示されます。この短所は（*Enterprise Edition* により使用されている）後者のスキーマで指摘されています。
- *Professional Edition* は `http://schemas.xmlsoap.org/wsdl/` を検証のために使用するため、拡張性要素は *Professional Edition* 内で無効とレポートされますが、`(http://www.altova.com/specs_wsdl.html#_document-s)` を使用する *Enterprise Edition* 内では有効とレポートされます。
- W 3C スキーマは W 3C 作業部会により提供されている公式のスキーマのため、発生するエラーに関しては、このスキーマが原因となるエラーに対しては Altova が対処することはできません。

例

以下に W SDL ファイルの一部を例として示します。このファイルでは要素 `getCityTimes` が宣言されています。この要素は、`getCityTimes` という間違った名前で参照されています。*Enterprise Edition* では、参照された要素が、ファイル内で既に宣言されているかをチェックしますが、*Professional Edition* にてこのチェックは行われません。ファイルの他の箇所が妥当であることを前提とすると、このファイルは *Professional Edition* にて妥当であると見なされ、*Enterprise Edition* では（`getCityTimes` がファイルのその他の場所にて宣言されていない場合）不正となります。

```
<s:element name="getCityTime">
```

```
   <s:complexType>
      <s:sequence>
         <s:element minOccurs="0" maxOccurs="1" name="city" type="s:string"/>
      </s:sequence>
   </s:complexType>
</s:element>
<s:element name="abc">
   <s:complexType>
      <s:sequence>
         <s:element ref="getCityTimes"/>
      </s:sequence>
   </s:complexType>
</s:element>
```

28.4.14 編集中に検証

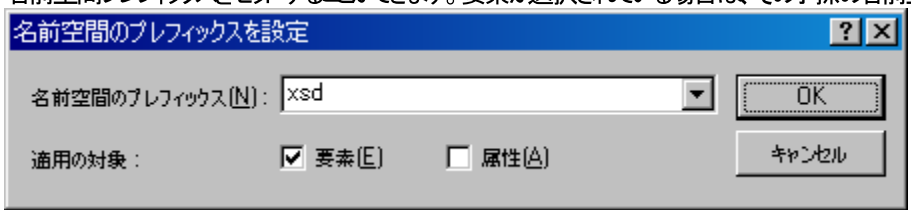
編集中に検証 コマンドは テキストビュー[130]、JSON グリッドビュー[610]、またはAuthentic ビューで入力中に検証を行う *編集中に検証* モードをオンとオフに切り替えます。コマンドのツールバーボタン または、オプションダイアログのファイル[1455]の *検証 > 編集中* オプションによりモードを切り替えることができます。

28.4.15 入力ヘルパーの更新

「XML | **入力ヘルパーを更新**」コマンドにより、使用されているDTD やスキーマを再ロードして、入力ヘルパーを更新します。開かれているXMLドキュメントがベースにしているXML スキーマやDTD を修正した場合、入力ヘルパーの更新を行って、スキーマにおける編集をインテリジェントな編集機能にも反映させるべきでしょう。

28.4.16 名前空間プレフィックス

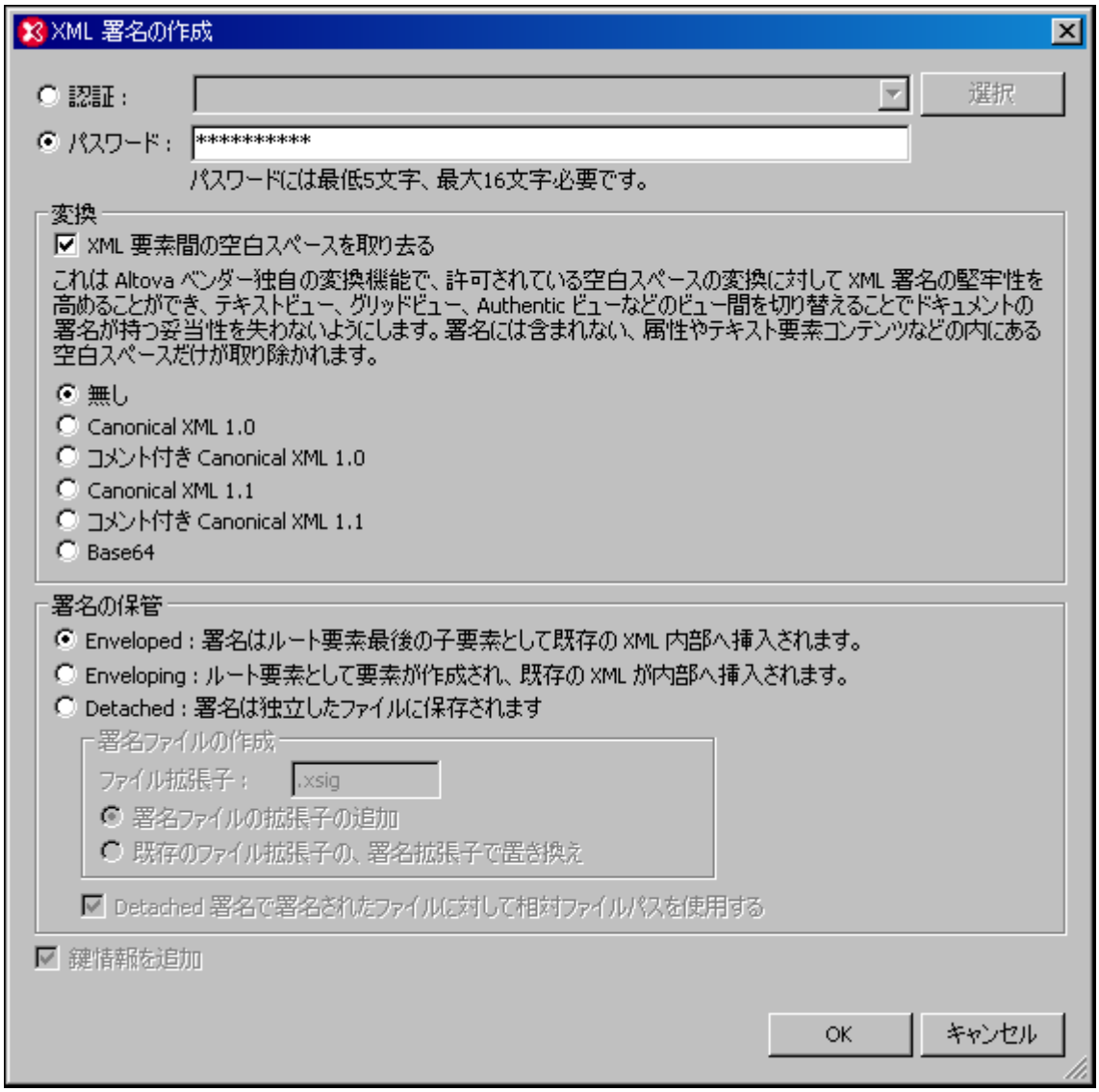
「XML | **名前空間プレフィックス**..」コマンドはグリッドビューでのみ使用することができ、ダイアログボックスから選択されている要素や属性の名前空間プレフィックスをセットすることができます。要素が選択されている場合は、その子孫の名前空間プレフィックスも変更されます。



名前空間プレフィックスを要素や属性、または両方に対してセットすることができます。名前空間プレフィックスは選択された要素や属性に、そして属性が選択されている場合はその子孫ノードに対しても、適用されます。

28.4.17 XML 署名の作成

XML 署名の作成コマンドはテキストボックス、グリッドビュー、スキーマビュー、W SDL ビュー、そして XBRL ビューで有効になり、アクティブな XML ドキュメントに対して XML 署名を作成することができます。このコマンドをクリックすることで、XML 署名の作成ダイアログが表示されます(以下のスクリーンショットを参照)。



確認方法: 証明書またはパスワード

署名は証明書またはパスワードにより行うことができます。使用するラジオボタンを選択してください。

- *認証* : 証明書を使用する場合、アクセス可能な 証明書ストア[378] 内に対応する秘密鍵が収められていなければなりません。署名はこの証明書の秘密鍵により生成されます。署名の検証を行うには、その証明書(または、その公開鍵)へアクセスする必要があります。証明書の公開鍵が署名の検証に使用されます。証明書を選択するには、「選択」ボタンをクリックし、証明書を指定します。証明書に関する詳細は、 証明書を使用する[377] セクションを参照ください。

- *パスワード*：5文字から16文字の長さのパスワードを入力してください。このパスワードは、署名の検証を行う際に必要になります。

変換

XML データ変換ならびに変換の結果が署名の作成に使用されます。署名の計算を行う前に、正規化アルゴリズムを XML データ（SignedInfo コンテンツ）のファイルへ適用することができます。以下にアルゴリズムの重要な特徴を記します:

- *（コメント付き）Canonical XML*：署名の計算にコメントが含まれる場合、XML 内にあるコメントに対して変更が加えられることで検証が失敗するようになります。署名の計算にコメントを含まない場合、XML ドキュメント内にコメントを加えたり、既にあるコメントを修正しても、署名の検証結果に影響は与えません。
- *Base64*：XML ドキュメントのルート要素（またはドキュメント要素）が Base64 にエンコードされているものとして、バイナリからの読み込みを試みます。ルート要素が Base64 でエンコードされていない場合、要素の種類によりエラーが返されるか、要素は空として認識されます。
- *無し*：変換を行うことはなく、ディスクに保存されているバイナリファイルから得られた XML データが署名の作成に直接使用されます。署名作成後にデータの編集が行われると、署名の検証が失敗するようになります。*空白スペースを取り去る* チェックボックスが有効になっている場合、全ての空白スペースが取り去られるため、空白スペースに対する変更は無視されます。*無し* オプションと *Canonical* オプションの違いは、正規化により XML データストリームが生成され、属性の順序などが正規化されるというものです。結果として正規化による変換を行った際には属性の順序といった変更が正規化され（つまり検証には成功し）、変換を伴わない場合、それらの変更が反映されることになります（つまり検証は失敗します）。

署名の保管

署名はXML ファイル内部に配置することができるほか、独立したファイルとしても作成することができます。以下のオプションが利用可能です:

- *Enveloped*：署名要素がルート要素（ドキュメント要素）最後の子要素として作成されます。
- *Enveloping*：署名要素がルート要素（ドキュメント要素）として作成され、XML ドキュメントが子要素として挿入されます。
- *Detached*：XML 署名が独立したファイルとして作成されます。この場合、署名ファイルのファイル拡張子を指定することができるほか、(i) XML ファイルに拡張子を追加するかたちで（例： test.xml.xsig）、または (ii) XML ファイルの XML 拡張子を置換ることで（例： test.xsig）ファイル拡張子の形式を指定することができます。署名ファイルにおける XML ファイルへのリファレンスを相対または絶対パスで行うか指定することもできます。

メモ: XML スキーマ（.xsd）や XBRL ファイルに対しては、外部署名ファイルとしてしか XML 署名を作成することはできません。WSDL ファイルの場合、署名ファイルを外部ファイルとして作成し、WSDL ファイル内に含めることができます（"enveloped"）。

メモ: XML 署名が独立したファイルとして作成された場合、署名ファイルにあるリファレンスにより XML ファイルと署名ファイルの関連付けが行われます。従って署名の検証を行う際には、署名ファイルがアクティブになっている状態で行う必要があります（XML ファイルがアクティブになっている状態で行うことはできません）。

鍵情報を追加

鍵情報を追加 オプションは、証明書をベースに署名を作成する際に有効になるオプションで、パスワードによる署名では無効になっています。

このオプションが有効になっている場合、公開鍵情報が署名内部に配置されます。オプションが選択されていない場合、鍵情報が署名に加えられることはありません。鍵情報を追加することで、検証を行う際に、証明書そのもの（証明書内部にある公開鍵情報）が必要なくなります（公開鍵情報が既に署名に含まれているため）。

28.4.18 XML 署名の検証

XML 署名の作成後にXML ファイルが変更されなかった場合に限りXML 署名の検証は成功し、そうでない場合、検証は失敗します。「XML 署名の検証」コマンドにより検証処理が実行され、検証の結果がメッセージウィンドウに表示されます。XMLSpy では以下に記される条件が満たされる場合、XML 署名の検証を行うことができます::

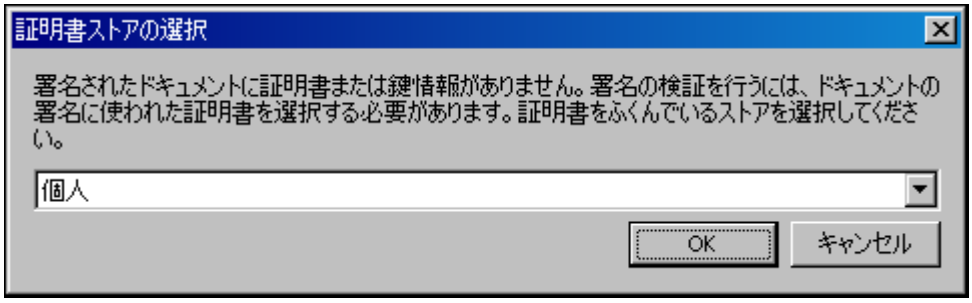
- XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれている[1236]
- XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれていない[1236]
- 証明書をベースにした署名が外部ファイルとして保存されており、鍵情報が署名に含まれている[1237]
- 証明書をベースにした署名が外部ファイルとして保存されており、鍵情報が署名に含まれていない[1237]
- パスワードをベースにした署名がXML ファイルに含まれている[1237]
- パスワードをベースにした署名が独立したファイルに存在する[1238]

XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれている

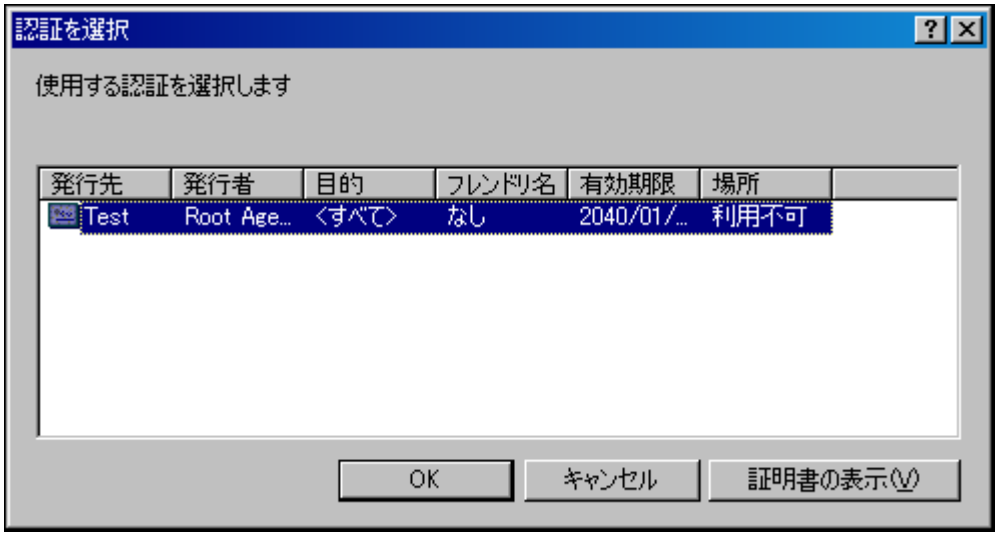
XML 署名の検証を行うには、XMLSpy にてXML ファイルをアクティブにします。「XML | XML 署名の検証」コマンドをクリックすると検証処理が実行され、(検証が成功または失敗したという)結果がメッセージウィンドウに表示されます。

XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれていない

証明書をベースにした署名に鍵情報が含まれていない場合、XMLSpy により検証に使用する公開鍵が収められている証明書を指定するよう促されます。XMLSpy にてXML ファイルがアクティブになっている状態で検証が実行されます。「XML | XML 署名の検証」コマンドをクリックすると、証明書が収められている証明書ストア[377] を選択するよう促されます(以下のスクリーンショットを参照)。



証明書ストア[377] を選択し、「OK」ボタンをクリックすると、そのストア内にある証明書を表示するダイアログが表示されます(以下のスクリーンショットを参照)。検証に必要な証明書を選択し、「OK」をクリックします。



検証処理が実行され、結果がメッセージウィンドウに表示されます。

証明書をベースにした署名が外部ファイルとして保存されており、鍵情報が署名に含まれている

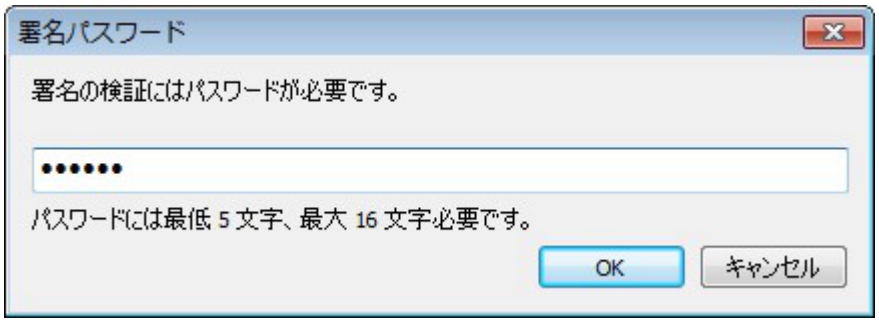
証明書をベースにしたXML 署名が独立したファイルとして存在する場合、署名ファイルがXMLSpy にてアクティブになっている状態で署名の検証が行われます。「XML | XML 署名の検証」コマンドをクリックすると検証処理が実行され、その結果(検証が成功したか失敗したか)がメッセージウィンドウにて表示されます。.

証明書をベースにした署名が外部ファイルとして保存されており、鍵情報が署名に含まれていない

証明書をベースにしたXML 署名が独立したファイルとして存在する場合、署名ファイルがXMLSpy にてアクティブになっている状態で署名の検証が行われます。「XML | XML 署名の検証」コマンドをクリックすると、検証に必要な公開鍵情報が得られる証明書を指定するようXMLSpy により促されます。: *XML ファイルに証明書をベースにした署名が含まれており、鍵情報が署名に含まれていない* 1236 の記述に従い、証明書を選択してください。

パスワードをベースにした署名がXML ファイルに含まれている場合

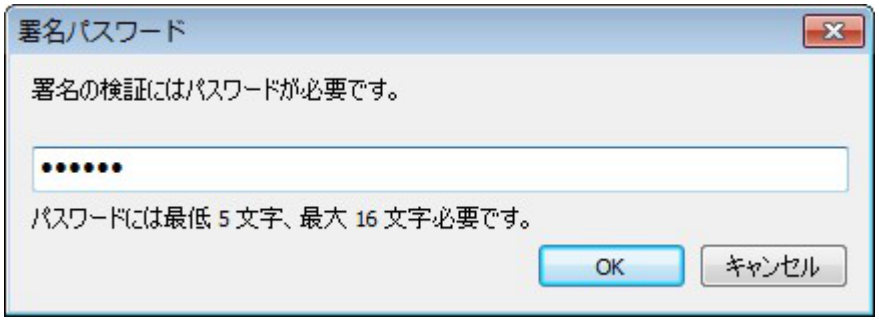
XML ファイル内にパスワードをベースにしたXML 署名が含まれている場合、XMLSpy ではそのXML ファイルがアクティブになっている状態で検証を行うことができます。「XML | XML 署名の検証」コマンドをクリックすると、パスワードを入力するためのダイアログが表示されます(以下のスクリーンショットを参照)。



5文字から16文字のパスワードを入力し、「OK」をクリックします。検証処理が実行され、その結果（検証が成功したか失敗したか）がメッセージウィンドウにて表示されます。

パスワードをベースにした証明が独立したファイルに存在する

独立したファイルにパスワードをベースにした署名が収められている場合、XMLSpy では署名ファイルがアクティブになっている状態で検証を行うことができます。「XML | XML 署名の検証」コマンドをクリックすると、パスワードを入力するためのダイアログが表示されます（以下のスクリーンショットを参照）。

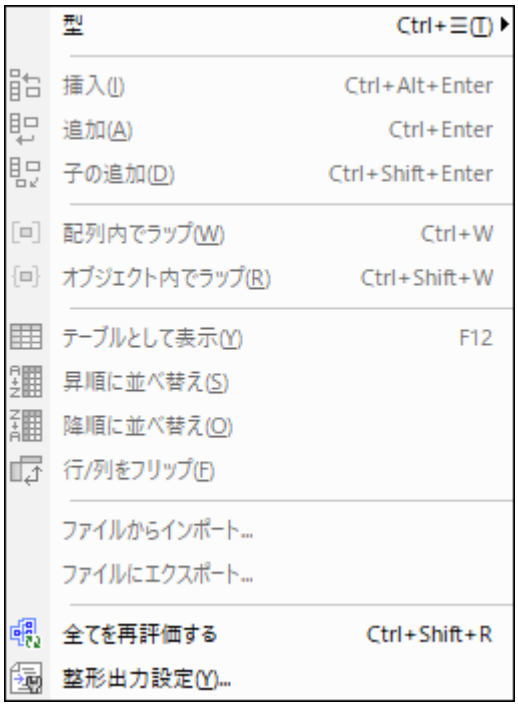


5文字から16文字のパスワードを入力し、「OK」をクリックします。検証処理が実行され、その結果（検証が成功したか失敗したか）がメッセージウィンドウにて表示されます。

(C) 2015-2021 Altova GmbH

28.5 JSON メニュー

JSON メニューには、JSON ドキュメントの処理を行う際に使用されるコマンドが収められています。コマンドの多くは JSON グリッドビュー[600] と作業するために使用されます。コマンドが現在の箇所に適用できない場合、これらのコマンドは無効化されます。

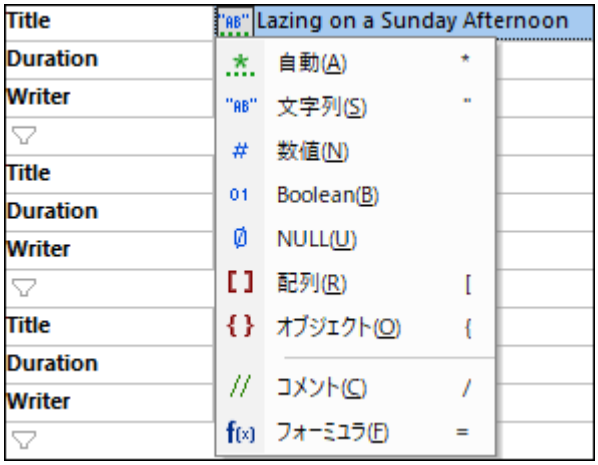


このメニューのコマンドはこのセクションのサブセクションで説明されています:

- 型[1239]
- 子の挿入、付加、追加[1240]
- 配列/オブジェクト内でラップ[1240]
- 右から左への読み取り[1240]
- 移動[1241]
- コピー[1241]
- 展開、折りたたみ[1241]
- 昇順に並べ替え、降順に並べ替え[1242]

28.5.1 型

型 (Ctrl + メニューキー) コマンドは JSON データ型を含むメニューを表示しています (*下のスクリーンショット*)。グリッドビュー内で現在選択されているセルに割り当てるメニューから 1 つの型を選択します (メニュー キーは通常キーボードの右下で **Ctrl** キーの横に配置されています。アイコンは以下のように表示されます:)。



編集の型に関する詳細は、JSON ドキュメントコンテンツの編集[607]を参照してください。

28.5.2 子を前/後に挿入、子の付加、子の追加

The **後に挿入**、**前に挿入**、**付加**、および**この追加** コマンドはグリッドビュー内の現在の選択がコンポーネント を挿入、付加、および子として挿入されることを許可している場合有効化されます。

- **後に挿入** は同じ型のコンポーネントを選択されたコンポーネントの上のグリッドビュー内で選択されたコンポーネントとして挿入します。
- **前に挿入** は同じ型のコンポーネントをグリッド行内の選択されたコンポーネントとして選択されたコンポーネントの上に挿入します。
- **付加** 同じ型のコンポーネントをグリッド行内の選択されたコンポーネントとして選択されたコンポーネントのすべての兄弟の後に付加します。
- **この追加** は新規の子コンポーネントを最後の子として追加します。型は最後に追加された子と同じです。

JSON ドキュメント構造の編集に関する詳細はJSON ドキュメント構造の編集[606]を参照してください。

28.5.3 配列/オブジェクト内のラップ

「**配列内でラップ**」と「**オブジェクト内でラップ**」コマンドはそれぞれ選択されたコンポーネントを配列またはオブジェクト内でそれぞれラップします。JSON ドキュメント構造の編集に関する詳細はJSON ドキュメント構造の編集[606]を参照してください。

28.5.4 右から左への読み取り

このコマンドは文字列型を含むセルのために使用することができ、読み取りの向きをセルの右から開始するように切り替えます。この機能はアラブ語とヘブライ語などの言語との作業に役に立ちます。

28.5.5 移動

コンポーネントを現在の位置からグリッド内で上下左右に移動しすることが可能になり、対応するコマンドが有効化されます。移動を実行するために対応するコマンド選択してください。

28.5.6 コピー

コピー コマンドのサブメニュー内のコマンドを使用して現在の選択範囲をコピーすることができます。

コマンド	説明
JSON テキストとしてコピーする	現在の選択項目がJSON またはJSON ラインテキストとしてシリアル化されます
構成されたテキストとしてコピーする	現在の選択項目がTSV（タブ区切り値）としてシリアル化されます
イメージとしてコピーする	現在のイメージセルがイメージとしてコピーされます
XPath のコピー	XPath 3.1 構文例: `?Artists?1?Albums?1?Tracks`
JSON ポインターをコピーする	JSON ポインター構文。例: `/Artists/1/Albums/1/Tracks`

以下の点に注意してください：

- *イメージとしてコピー*はBase64 エンコード文字列をイメージファイルにエクスポートします。Base64 エンコード内で指定されているイメージ型に一致しているファイル拡張子がイメージファイルに与えられていることを確認してください。エクスポートされる文字列がBase64 エンコードではない場合、指定したファイル名を持つファイルが作成されますが、コンテンツは存在しません。
- テキストをセルに挿入するには、テキストをコピーしてセルに張り付けます。
- 例えばイメージなどをファイルからインポートするには、[編集 | 挿入 | エンコードされた外部ファイル]1154コマンドを使用します。Base64 エンコード文字列としてイメージファイルを挿入し、この文字列をJSON グリッドビューで表示します。インポートのためにサポートされているイメージファイルフォーマットは以下の通りです: PNG、JPEG、BMP、GIF、TIFF。

28.5.7 展開、折りたたみ

フルに展開する
このコマンドは選択されているコンポーネントまたはその子孫が折りたたまれている場合に有効化されます。コンポーネントと子孫コンポーネントが展開されます。

選択されていない範囲を折りたたむ
このコマンドは選択されているコンポーネントとその祖先コンポーネント以外のすべてのコンポーネントを折りたたみます。

昇順に並べ替え、降順に並べ替え
これらのコマンドは選択されたテーブル列に従いテーブルのコンポーネントを並べ替えるために使用されます。行が隣接する場合、並べ替えはテーブル行の選択範囲にも適用することができます。

28.5.8 昇順に並べ替え/降順に並べ替え

コンポーネントのテーブル表示内の列が選択されている場合、これらの2つのコマンドは有効化されます。列を選択するにはヘッダーを選択します。並べ替えは列内の値をベースにしています。

28.6 DTD/スキーマメニュー

「DTD/スキーマ」メニューにはDTD やXML スキーマを使った作業を効率的に行うためのコマンドが収められています。

28.6.1 DTD の割り当て

「DTD/スキーマ | DTD の割り当て...」コマンドはXML ファイルがアクティブな時に有効になります。このコマンドによりDTD がXML ドキュメントに割り当てられ、ドキュメントの検証や、インテリジェントな編集を行うことができるようになります。コマンドを実行すると、ファイルの割り

当てダイアログが表示され、割り当てを行うDTD ファイルを指定することができます。ファイルは(参照 1132 ボタンをクリックすることで) グローバルリソースやURL から、または(ウィンドウボタンをクリックすることで) XMLSpy にて既に開かれているウィンドウから選択することもできます。*…に対する相対パスにする* チェックボックスにチェックを入れることで、割り当てられたDTD ファイルに対するパスを相対パスにすることができます。操作が完了すると、割り当てられたDTD に対する参照とともに、DOCTYPE 宣言がXML ドキュメントに追加されます。DOCTYPE 宣言は以下のようになります:

```
<!DOCTYPE main SYSTEM "http://link.xmlspy.com/spyweb.dtd">
```

メモ: 新たにXML ファイルが作成されたときも、DTD を割り当てることができます。

28.6.2 スキーマの割り当て

「DTD/スキーマ | スキーマの割り当て...」コマンドはXML ファイルがアクティブな時に有効になります。このコマンドによりXML スキーマがXML ドキュメントに割り当てられ、ドキュメントの検証や、インテリジェントな編集を行うことができるようになります。コマンドを実行すると、ファイルの割り当てダイアログが表示され、割り当てを行うXML スキーマファイルを指定することができます。ファイルは(参照 1132 ボタンをクリックすることで) グローバルリソースやURL から、または(ウィンドウボタンをクリックすることで)XMLSpy XMLSpy にて既に開かれているウィンドウから選択することもできます。*…に対する相対パスにする* チェックボックスにチェックを入れることで、割り当てられたXML スキーマファイルに対するパスを相対パスにすることができます。操作が完了すると、適切な名前空間とともにXML スキーマの割り当てがXML ドキュメントに追加されます。割り当ては以下のようになります:

```
xmlns="http://www.xmlspy.com/schemas/icon/orgchart"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xsi:schemaLocation="http://www.xmlspy.com/schemas/icon/orgchart
    http://schema.xmlspy.com/schemas/icon/orgchart.xsd"
```

28.6.3 他の DTD を読み込み

「DTD/スキーマ | 他のDTD を取り込み...」コマンドにより、他のDocument Type Definition (DTD) や外部解析対象エンティティを、文章型定義の内部サブセットやDTD ドキュメントへインクルードします。この処理により対応する外部解析対象宣言が定義され、そのエンティティを以下のように使用することができるようになります:

```
<!ENTITY % navigation.dtd SYSTEM "S:\xml\navigation.dtd">
%navigation.dtd;
```

コマンドによりファイルの割り当てダイアログが表示され、使用しているDTD にインクルードするDTD を指定することができます。

メモ: このコマンドはグリッドビューでのみ使用することができます。

28.6.4 DTD へ移動

「DTD/スキーマ | DTD へ移動」コマンドにより、アクティブな XML ドキュメントがベースにしている DTD を開くことができます。DTD が割り当てられていない場合、エラーメッセージが表示されます。

28.6.5 スキーマへ移動

「DTD/スキーマ | スキーマへ移動」コマンドにより、アクティブな XML ドキュメントがベースにしている XML スキーマを開くことができます。XML スキーマが割り当てられていない場合、エラーメッセージが表示されます。

28.6.6 定義へ移動

「DTD/スキーマ | 定義へ移動」コマンドにより、選択されているノードに対応した、文章型定義（DTD）やスキーマ内にある要素や属性の定義が表示されます。

アイテムの定義をグリッドビューで確認するには

1. アイテムをクリックします。
2. メニューオプションの「DTD/スキーマ | 定義へ移動」を選択するか、アイコンをクリックします。

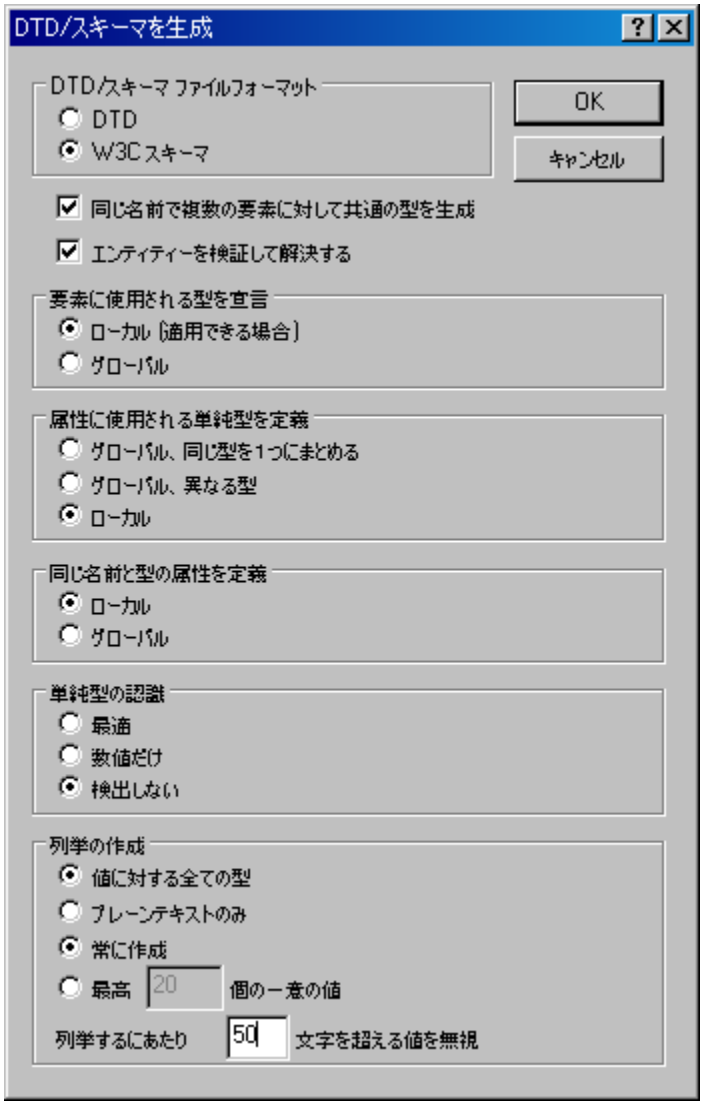
アイテムの定義をスキーマビューで確認するには

- 定義を確認したいアイテムを Ctrl キーを押下しながらダブルクリックするか、
- アイテムをクリックした後、メニューオプションから「DTD/スキーマ | 定義へ移動」を選択、またはアイコンをクリックします

どちらの場合でも、対応する DTD または スキーマファイルが開かれ、目的の定義がハイライトされます。

28.6.7 DTD/スキーマの生成

「DTD/スキーマ | DTD/スキーマの生成」コマンドにより、XML ドキュメント（又はプロジェクトウィンドウのフォルダーに収められた複数の XML ファイル）から新たな DTD または W3C XML スキーマを生成することができます。このコマンドは XML ドキュメントから DTD や XML スキーマを生成する際に便利な機能となります。



XML スキーマを生成する際には、以下のオプションが利用可能です:

- ***要素*** : 要素の型を、ローカルかグローバルか選択することができます(要素に使用される型の定義)。同じ名前の要素が複数ある場合、これら要素に対して共通の型を宣言することができます(共通の型を生成)。
- ***属性*** : 属性の単純型(属性に使用される単純型の定義)は、(i) 共通のグローバル型、(ii) 一意のグローバル型、または (iii) ローカル型、として定義することができます。同じ名前の属性をローカルまたはグローバルで定義することができます。
- ***単純型の認識*** : 型の認識(単純型の認識)を、(i) できるだけ最適な認識、(ii) 数値のデータ型だけ検出、または (iii) データ型の検出無しからセットすることができます((iii) の場合は、全てのデータ型に xs:string がセットされます)。
- ***エントリーの解決*** : XML ドキュメントにおいて、エンティティは要素と属性の値両方にて出現しえます。そのため、それらエンティティが解決されるかどうかが、列挙の値に対して重要な意味を持ちます(エンティティを検証して解決)。更に、エンティティが解決されているかによって(特にパースされたエンティティにマークアップが含まれる場合)、コンテンツモデルが影響を受けることもあります。スキーマが生成される前に、XML ドキュメントが検証されることに注意してください。ドキュメントが妥当でない場合、スキーマの生成は行われません。
- ***列挙*** : 値に対する全ての型、または文字列の値のみから列挙を行うことができます。

(C) 2015-2021 Altova GmbH

DTD を生成すると、エンティティの解決や列挙オプションが有効になります。

DTD/スキーマの生成コマンドは、通常メインウィンドウで処理されますが、アクティブなプロジェクトウィンドウ上のファイルやフォルダー、またはファイルのグループに対しても「DTD/スキーマの生成」コマンドを使用することができます。

1つ以上の名前空間ないで要素または属性が存在する場合、XMLSpy は個別のXML スキーマを各固有の名前空間のために生成します。このため、複数のファイルがディスク上に作成される可能性があります。

28.6.8 DTD のフラット化

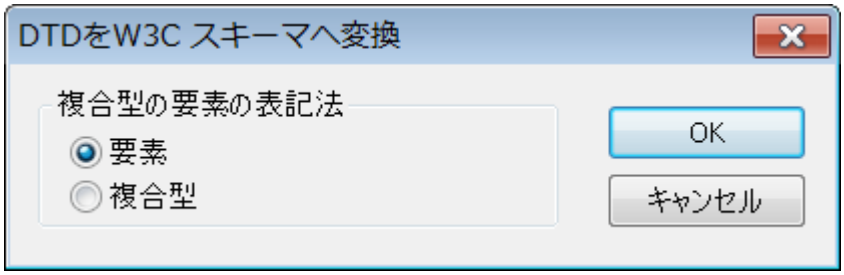
DTD のフラット化 コマンドは、DTD がアクティブなドキュメントな場合有効化されます。新規のフラットな DTD を作成し、パラメーターエンティティを削除し、モジュールのコレクションからを単一のDTD 作成します。IGNORE とマークされた箇所を抑制し、使用されていないパラメーターエンティティを削除します。

コマンドは生成された DTD ファイルでの場所を選択する保存ダイアログをポップアップします。「**保存**」をクリックして、変換を実行します。フラット化された DTD ファイルは生成され、XMLSpy 内で開かれます。

28.6.9 DTD をスキーマに変換

DTD がアクティブなドキュメントの場合、**DTD をスキーマに変換** コマンドが有効化されます。DTD を XML スキーマドキュメント（XSD）に変換します。

コマンドにより、複合型要素の要素 または 複合型への変換を選択することのできる、DTD を W 3C スキーマに変換するダイアログがポップアップします（*下のスクリーンショット*）。「OK」をクリックすると、生成された XSD ファイルを保存する場所を選択するようにプロンプトされます。変換を実行するために「保存」をクリックします。XMLSpy XSD ファイルが生成され、内で開かれます。



DTD を XML スキーマに変換する場合、XMLSpy により、情報が限られているため、いくつかの想定がなされます。最も顕著な例は、特定のDTD コンポーネントの値が、構文的な解析よりも、文字通り扱われる点です。これは、プログラムが複数の使用の可能性について特定できないためです。この場合、生成された変換を変更する必要があります。

拡張するために、生成された変換を注意深く確認する必要がある場合、修正の必要がある可能性のある箇所は以下にリストされています。

属性 Datatyping
DTD は10 属性 データ型のみを許可しますが、XML スキーマは、例えば、40 以上のデータ型 と派生する データ型を許可します。生成された XML スキーマを拡張する場合、例えば、更に制約された データ型を使用する場合などがあげられます。 XML スキーマがDTD に変換される 1249 場合、データ型 情報は、失われます。

名前空間
DTD は、名前空間を考慮していません。この結果、名前空間はDTD 内で指定され、要素と属性名内にハードコードされている必要があります。これは、1つのスキーマから他のスキーマに変換する際に問題を引き起こす可能性があります。

エンティティ
XML スキーマにはDTD のための一般的なエンティティ宣言に相当する宣言は存在しません。XMLSpy がDTD をXML スキーマに変換する場合、エンティティ宣言は無視されます。

解析されていないデータ宣言
DTD とXML スキーマは、解析されていないデータを扱うために異なるメカニズムを使用します。この点に関しては下で説明されています。

DTD は次のしくみを使用します:

- 表記は宣言されており、名前と識別子により構成されています。例えば:
  ```
  <!NOTATION gif SYSTEM "image/gif">
  ```
- エンティティを宣言します。例えば:
  ```
  <!ENTITY cover_img SYSTEM "graphics/cover_img.gif" NDATA gif>
  ```
- 通常、対応する属性上で ENTITY の属性の型を指定します。例えば:<!ELEMENT img EMPTY>
  ```
  <!ATTLIST img format ENTITY #REQUIRED>
  ```

XML スキーマ内での、対応するメカニズムは、以下のとおりです:

- 表記法を宣言します。この関数は、DTD と同じ方法です。
  ```
  <xs:notation name="gif" public="image/gif"/>
  ```

  public 属性 は、識別子を持つためには必須であることご注意してください。任意のsystem 属性にはシステム識別子が存在し、通常、表記型のソースと作業できる実行可能です。
- NOTATION データ型を使用して、notation 宣言を与えられた属性値と関連付けます。しかしながら、NOTATION データ型を直接使用することはできず、NOTATION データ型から他のデータ型を派生する必要があります。
  ```
  <xs:simpleType name="formatType">
     <xs:restriction base="xs:NOTATION">
        <xs:enumeration value="gif"/>
        <xs:enumeration value="jpeg"/>
     </xs:restriction>
  </xs:simpleType>
  ```
- 属性をNOTATION データ型から派生したデータ型に関連付けます。例:
  ```
  <xs:complexType name="imgType">
     <xs:attribute name="height"/>
     <xs:attribute name="width"/>
     <xs:attribute name="location"/>
     <xs:attribute name="format" type="formatType" use="required"/>
  </xs:complexType>
  <xs:element name="img" type="imgType"/>
  ```

DTD からXML スキーマへ変換する場合、XMLSpy は以下をおこないます:

- 以下は
  ```
  <!ATTLIST image format ENTITY #REQUIRED
  ...>
  ```

  以下に変換されます
  ```
  <xs:attribute name="format" type="xs:ENTITY" use="required"/>
  ```

- 以下は
  ```
  <!NOTATION gif SYSTEM "image/gif">
  ```
  以下に変換されます
  ```
  <xs:notation name="gif" system="image/gif"/>
  ```

以下の変更を加えます:

1. `<xs:notation name="gif" system="image/gif"/>` のような表記は、`system` を `public` と置換え、任意のシステム識別子を必要な場合追加します。
2. `NOTATION` データ型から `formatType` のために説明されているとおりから派生します。
3. 派生したデータ型対応した属性と関連付けします。

メモ: XML スキーマ仕様に従い、必要に応じて外部エンティティを宣言します。

28.6.10 フラット化されたスキーマ

フラット化されたスキーマ コマンドは、XML スキーマがアクティブなドキュメントの場合有効化されます。新規のフラットな XSD を以下の方法で生成します (i) アクティブな スキーマのグローバルコンポーネントとして含まれる全てのスキーマのコンテンツを追加する、(ii) 含まれているスキーマを削除する。

コマンドは、スキーマデザインメニューの[フラット化されたスキーマ](#)[1275]をリダイレクトします。[フラット化されたスキーマ](#)[1275] コマンドは、スキーマビューのみで使用することができるため、スキーマビューへの切り替えが問われます。 詳細に関しては、[フラット化されたスキーマ](#)[1275] コマンドを参照してください。

28.6.11 スキーマを DTD に変換する

スキーマを DTD に変換する コマンドは、XML スキーマがアクティブなドキュメントな場合に有効化されています。 XML スキーマドキュメント (XSD) を DTD に変換します。

コマンドは、生成された DTD ファイルを保存する保存先を選択する保存ダイアログをポップアップします。「**保存**」をクリックして、変換を行います。DTD ファイルは生成され、XMLSpy 内で開かれます。

以下の点に注意してください:

1. XML スキーマを DTD に変換する場合、名前空間 URI または名前空間宣言内ではなく、XML スキーマ内で使用される名前空間プレフィックスは、DTD 内の対応する要素と属性の名前全体で使用されます。
2. DTD に対して XML ドキュメントの検証を行う際、名前空間は無視されます。名前空間の宣言そのものは変換されません。
3. xs:schema 要素の elementFormDefault ならびに attributeDefault 属性により、どの要素や属性が変換処理に含まれるか指定することができます。unqualified がセットされていると、変換の際にグローバルに宣言された要素や属性に対してだけプレフィックスが含まれます。qualified がセットされていると、変換の際に全ての属性や要素に対してプレフィックスが含まれます。
4. プレフィックスは、それぞれ対応する文字列とコロンに変換されます。デフォルトの名前空間にある要素や属性は、default_NS_X で始まる名前に変換されます (X には 1 から始まる数値が入り、XML スキーマで使用されているデフォルトの名前空間の数だけ数値は用意されます)。
5. DTD において、要素名はパラメーターエンティティにより構成されます。これにより、XML ドキュメント内にあるプレフィックスが変更されたときでも、DTD 内のプレフィックスを簡単に変更することができるようになります。パラメーターエンティティ定義の変更は、DTD ドキュメント内で直接行うか、XML ドキュメントの内部 DTD サブセットにてパラメーターエンティティをオーバーライドすることで行うことができます。

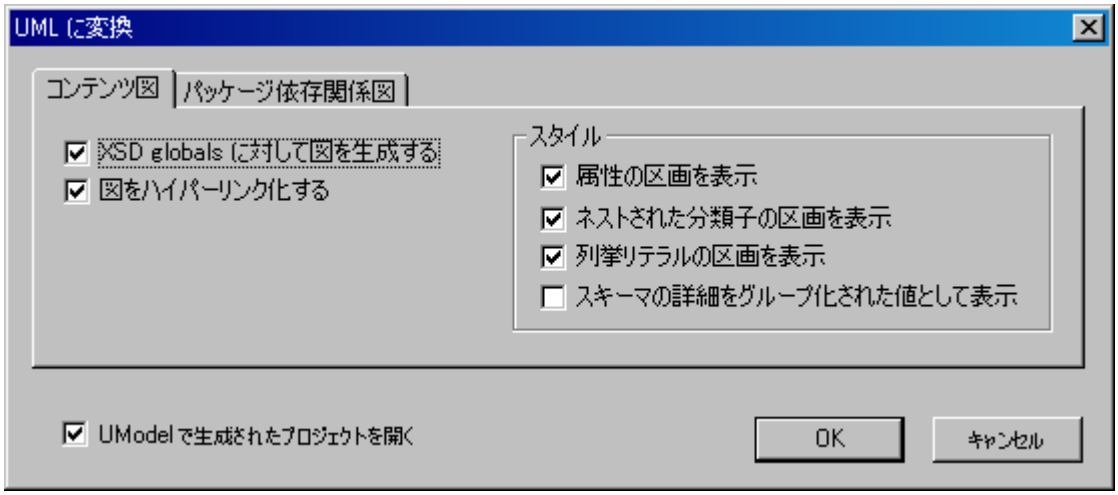
メモ: DTD における名前空間はセマンティックな意味を持っておらず、XML スキーマから変換された名前空間プレフィックスは、DTD で定義された要素名や属性名における文字列の一部でしかありません。

28.6.12 UML へ変換

「DTD/スキーマ | UML へ変換」コマンドにより、W3C XML スキーマを Altova UModel プロジェクト (.ump) ドキュメント (UModel プロジェクト) へ変換することができます。UMP は、Altova の UML モデリングアプリケーションである Altova UModel のネイティブフォーマットです。UMP ファイルは Altova UModel にて閲覧、編集することができます。

スキーマを UML へ変換するには、以下の操作を行います:

1. スキーマが開かれた状態で、**UML へ変換**コマンドをクリックします。UML に変換ダイアログが表示されます (以下のスクリーンショットを参照)。



2. コンテンツ図タブにて、*XSD globals に対して図を生成する* オプションを選択します。このオプションにより、各グローバルコンポーネントに対してコンテンツモデル図が UModel プロジェクトにて生成されます。
3. 必要なオプションをダイアログから選択します。各オプションの説明については、以下を参照ください。
4. 作成されたプロジェクトを UModel にて直ちに確認したい場合、*UModel で生成されたプロジェクトを開く* オプションを選択してください。
5. 「OK」をクリックします。
6. 名前を付けて保存ダイアログが表示されるので、出力先フォルダーを参照し、UMP ファイルの名前を入力した後、**保存**をクリックします。

UML への変換オプション

UML に変換ダイアログでは、以下のオプションが利用できます。

コンテンツ図タブにて:

- *図をハイパーリンク化する* オプションにより、各ダイアグラムにてグローバルコンポーネントのエントリーへのリンクがモデルツリービューに作成され、スキーマ階層からコンポーネントの確認を素早く行うことができるようになります。
- *スタイル* ペインでは、区画を表示オプションにより、表示する区画を指定することができます。

パッケージ依存関係タブにて:

- *図を生成* オプションにて、パッケージ依存関係図を生成するか指定することができます。パッケージ依存関係図ではパッケージ全体の概要が表示され、パッケージコンポーネント間の関係を確認することができます。このタブにある他のオプションは、図を生成オプションが選択されていないと有効になりません。
- *ダイアグラムのパッケージをハイパーリンク化* オプションにより、パッケージ依存関係図からモデルツリービューへのリンクが作成されます。
- パッケージ依存関係図のレイアウトに対して4つのオプションが指定できます：(i) 自動レイアウトを行わない、(ii) 階層レイアウト（自動レイアウトと階層オプションを選択）、(iii) ブロック（自動レイアウトとブロックオプションを選択）、(iv) 均等に配置（自動レイアウトと有向化を強制オプションを選択）。UModel にてダイアグラムを編集することでレイアウトを修正することもできます。

メモ: UML への変換機能は、W3C XML スキーマだけをサポートしています。

28.6.13 MapForce で DB/Excel/EDI から XML を生成

「DTD/スキーマ | MapForce で DB/Excel/EDI から XML を生成...」コマンドにより、（アプリケーションがインストールされていれば）Altova の MapForce が起動します。MapForce ではスキーマを他の DTD や XML スキーマ、データベースへマッピングして、XML を生成することができます。

28.6.14 StyleVision で HTML/PDF/Word 出力をデザイン...

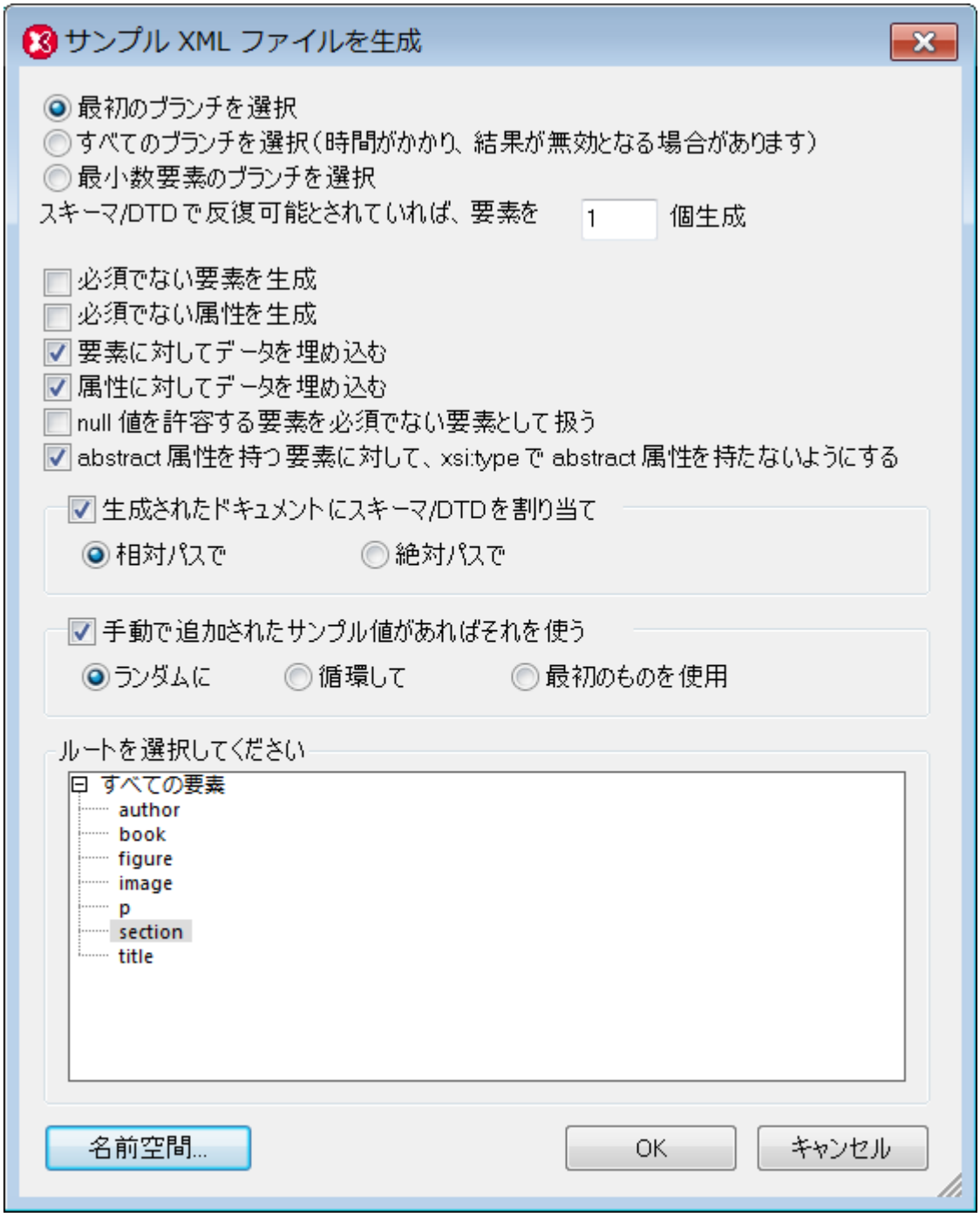
「DTD/スキーマ | StyleVision で HTML/PDF/Word 出力をデザイン...」コマンドにより（アプリケーションがインストールされていれば）Altova の StyleVision が起動します。StyleVision では HTML や PDF、RTF 出力に対してスタイルシートをデザインすることができます。

28.6.15 サンプル XML/JSON ファイルの生成

「サンプル XML/JSON ファイルの生成」コマンドがテキストビュー、グリッドビュー、とスキーマビューで有効化されており、現在アクティブなスキーマファイルをベースにして、XML または JSON インスタンスが生成されます。現在アクティブなファイルが DTD または XML スキーマの場合、XML インスタンスファイルが生成されます。現在アクティブなファイルが JSON スキーマの場合、JSON インスタンスファイルが生成されます。

サンプル XML ファイルの生成

DTD または XML スキーマがアクティブな場合、スキーマをベースにしてサンプル XML インスタンスを生成することができます。「サンプル XML/JSON ファイルの生成」コマンドをクリックすると、サンプルの生成のためのオプションを指定することのできるサンプル XML ファイルの生成 ダイアログが表示されます（下のスクリーンショット）。



選択グループの要素

選択グループは、使用される可能性のある要素のグループです。例えば、アイテムという名前の要素が以下の3 つの要素から構成される `choice` グループを持つと定義されている場合、(要素の`maxOccurs` 属性内で指定された発生に最大数である)アイテムはこれら3 つの要素を子要素として持つことができます: `cd`, `dvd`, `book`。

サンプルXML ファイルの生成 ダイアログ内では、以下を選択することができます (i) choice グループの最初のブランチ(要素)、(ii) 全てのブランチ、または、(iii) 子孫要素の最小数を持つブランチ。choice グループから1つのブランチが許可されているため、全てのブランチの選択は、無効なドキュメントを生成することができます。

choice グループのブランチは、(maxOccurs 値 が1より大きいため) 繰り返すことができ、ダイアログの最初のテキスト・ボックス内で、生成する繰り返すことのできる要素の数を、最大99まで、指定することができます。choice グループの maxOccurs 属性は、unbounded として、または、大きな番号として定義されます。サンプルXML ファイルの生成 ダイアログ内で全てのブランチが選択されている場合、choice グループの maxOccurs は、ダイアログの最初のテキスト・ボックス内で指定される繰り返すことのできる要素の数に制限されます。

必須ではない要素の生成
このオプションの有効化は、スキーマ内で定義された必須と必須ではない両方の要素を生成します。

必須ではな属性の生成
このオプションの有効化は、スキーマ内で定義された必須と必須ではない両方の要素を生成します。

繰り返しとマークされている場合 DTD 内でのX 要素の生成
このオプションの有効化により、テキスト・ボックス内に入力する繰り返すことのできる要素の数量を生成します。 choice グループ内に含まれるものも含む全ての要素に適用されます。

要素と属性にデータをフィルする
このオプションの有効化により対応する要素と属性のデータの値が挿入されます。例えば、要素がpf データ型 string として定義されている場合、要素には、string のダミーの値が与えられます。

Nillable 要素 と抽象的な型
nillable 要素のコンテンツは、必須ではないものとして扱うことができ、抽象型を持つ要素は自身の xsi:type 属性ために非抽象的な型を使用することができます。

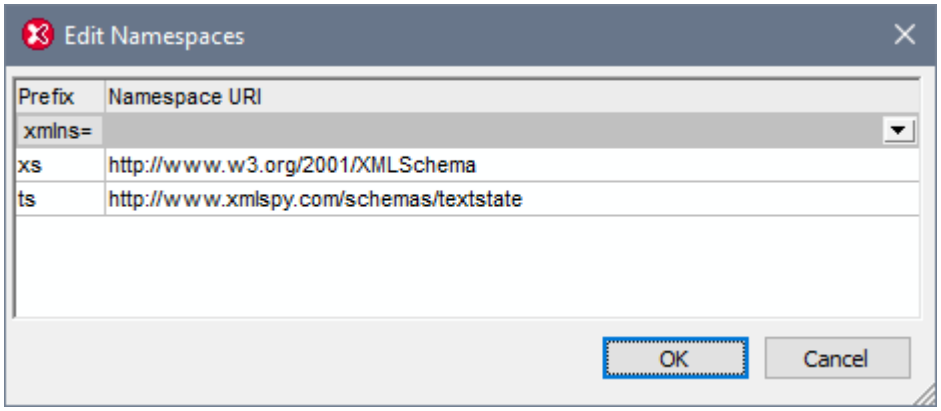
生成された XML ファイルのためのスキーマの割り当て
XML ファイルを生成するために使用されたスキーマを相対、または、絶対パスを持つ生成されたXML ファイルに割り当てることができます。

使用できる場合、手動で追加されたサンプル 値を使用する
スキーマコンポーネントに割り当てられたサンプル値が存在する場合、そのコンポーネントのコンテンツまたは値として使用されます。個別のコンポーネントに関しては、サンプル値が、サンプルタブ内のファセット 入力ヘルパー[215] 内で割り当てられています。使用することのできるサンプル値からの値が単一ファイルの生成のために指定することができます:

- ランダムな選択
- コンポーネントの各インスタンスのための各サンプル値。
- 最初の値

ルート要素
スキーマが1つ以上のグローバル要素を含む場合、サンプルXML ファイルのために必要なルート要素はリストから選択することができます。

名前空間
名前空間 ボタンをクリックして、名前空間の編集ダイアログを開きます (*下のスクリーンショット*)。名前空間は、スキーマ内で定義されます。また、サンプルXML ファイル内で必要とされる、標準 XML スキーマ 名前空間は、このダイアログに表示されます。

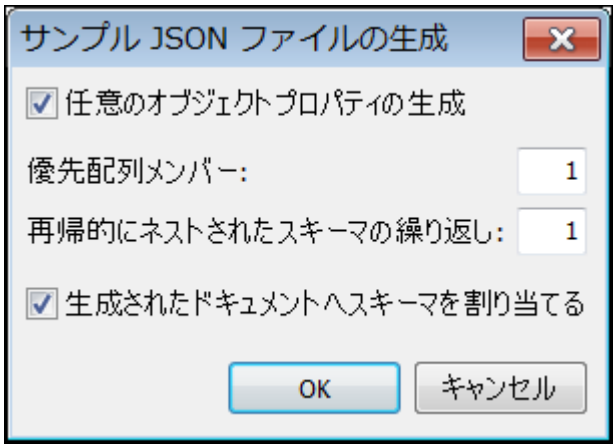


以下を編集することができます:

- ドキュメントの名前空間にバインドされている名前空間プレフィックス。このダイアログ内で設定されている名前空間プレフィックスが（生成されたXML ファイル内で使用されます。このダイアログ内で設定される名前空間プレフィックスは、対応する名前空間内のノードにプレフィックスを与えるために使用されます。例えば、スクリーンショットは、`http://www.xmlspy.com/schemas/textstate` 名前空間内のノードがサンプルファイル内で `ts:` のプレフィックスを与えられることを表示しています。
- ドキュメントの名前空間の1つを、`xmlns=` コンボボックス内でデフォルトの名前空間（`xmlns=`）を選択することにより希望する名前空間になるように設定することができます。名前空間内のノードは、デフォルトの名前空間として選択され、名前空間プレフィックスなしで生成されます。

サンプル JSON ファイルの生成

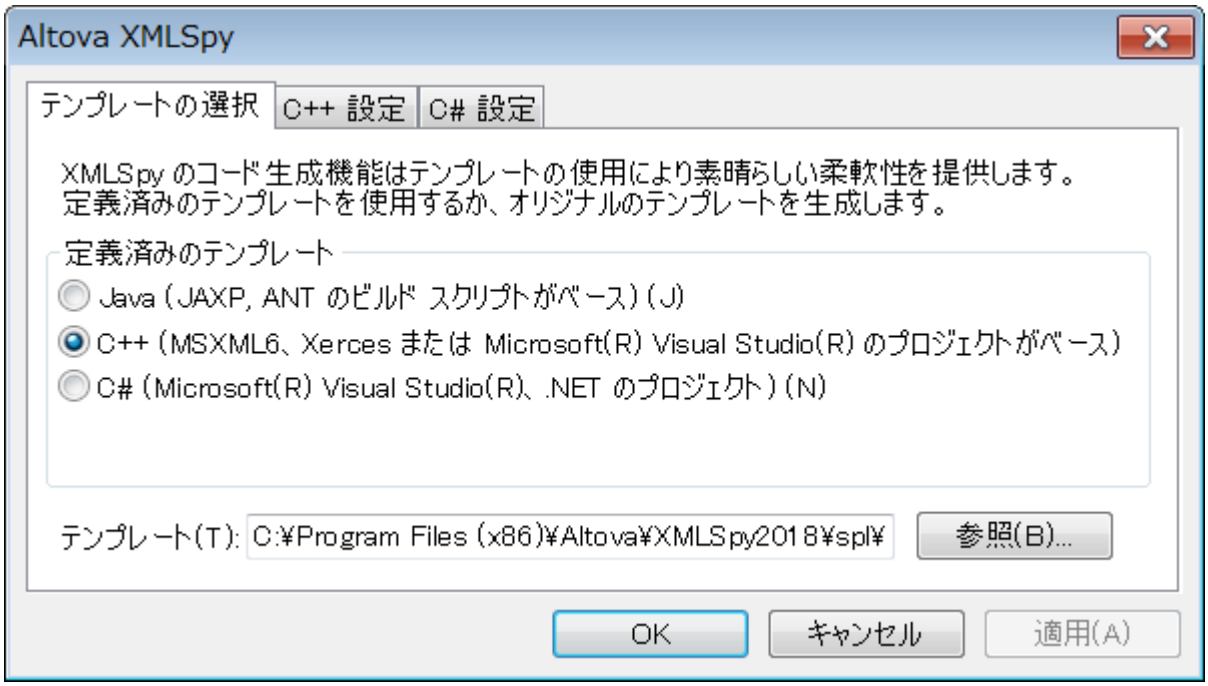
JSON スキーマがアクティブな場合、JSON スキーマをベースにした、サンプル JSON インスタンスを生成することができます。コマンドをクリックすると、サンプルの生成のオプションを指定することのできる、サンプル JSON ファイルの生成ダイアログが表示されます（下のスクリーンショット）



必須ではないオブジェクトプロパティ[640]、必須ではないパターンプロパティ[640]、配列の長さ[649]、および再帰的な定義の繰り返しの生成を選択することができます。アクティブな JSON スキーマが自動的に生成された JSON サンプルファイルに割り当てられる[662] かを指定する事もできます。

28.6.16 プログラムコードの生成

「DTD/スキーマ | プログラムコードの生成...」コマンドを使うことで、アクティブなスキーマ(DTD または XML スキーマ)ドキュメントから、Java、C++、または C# のクラスファイル定義を生成することができます。



生成に関する詳細については、このドキュメンテーションのコードジェネレーター[1017] セクションを参照ください。

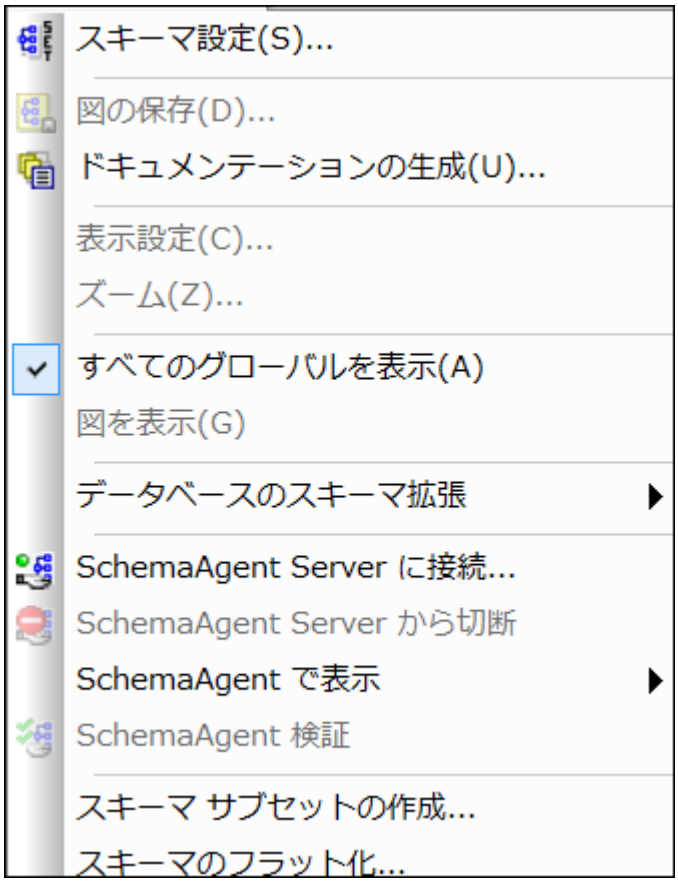
28.6.17 メモリキャッシュの破棄

「DTD/スキーマ | メモリキャッシュの破棄」コマンドにより、メモリ上にキャッシュされているスキーマ(DTD ならびに XML スキーマ)ドキュメントを破棄します。XMLSpy では検証や編集のスピードを向上させるため、最近使用されたスキーマドキュメントや外部の解析されたエンティティをメモリに蓄えます。キャッシュされているドキュメントから得られた情報は、 定義へ移動[1245] コマンドが呼び出された時などにも表示されます。

メモリにあるキャッシュを破棄することで、システム上のメモリ空き容量を増やしたり、ベースとしているスキーマの情報を更新することができます。

28.7 スキーマ設計メニュー

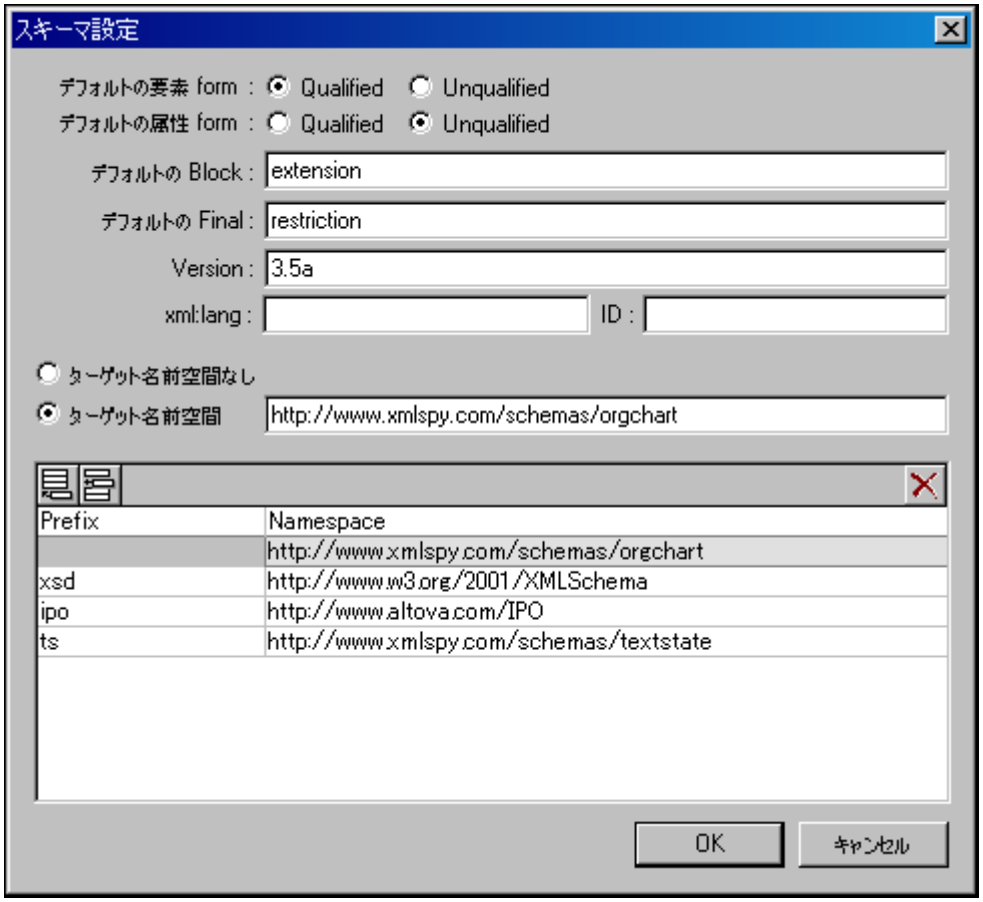
「スキーマ設計」メニューではXMLSpy のスキーマビュー設定を行ったり、XML スキーマのデザインを GUI から行うことができます。XML スキーマがスキーマビューでアクティブな状態であれば、これらメニューコマンドを使用することができます。



このセクションでは、このメニューから使用することのできるコマンドを説明します。

28.7.1 スキーマ設定

「スキーマ設計 | スキーマ設定...」コマンドを使用することで、アクティブなスキーマに対してグローバルな設定を行うことができます。これらの設定はXML スキーマドキュメント要素 xs:schema の属性やその値に対して適用されます。



スキーマ設定ダイアログ内で定義される設定（XSD モードが1.1 に設定されている場合）は次の`xs:schema` 要素を作成します。

```
<xs:schema xmlns="http://www.altova.com/schemas/org"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       targetnamespace="http://www.altova.com/schemas/org"
       elementFormDefault="qualified"
       attributeFormDefault="unqualified"
       xpathDefaultNamespace="##targetNamespace"
       version="1.1"
       defaultAttributes="Contact">
```

メモ: スキーマ設定 ダイアログの内容は、アクティブな XSD モードにより異なります。XSD 1.0 がアクティブな モードな場合、XSD 1.1 属性は、ダイアログ内に存在しません。

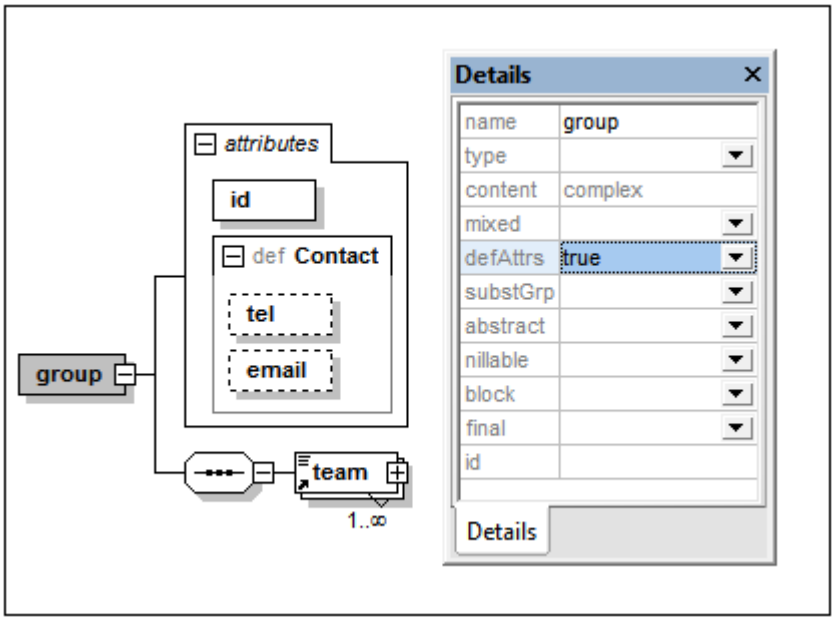
`defaultAttributes` と`xpathDefaultNamespace` 属性は、XML スキーマ1.1 の機能で、XSD 1.1 モード[161] のみで使用することができます。他の属性は、XSD 1.0 とXSD 1.1 で使用することができます。

version 属性

バージョン 属性はドキュメント バージョンです。ドキュメントのXSD バージョン[161] ではありません。

defaultAttributes 属性

defaultAttributes 属性により、隙間内の全ての複合型のデフォルトの属性グループとして属性グループを選択することができます。 デフォルトの属性 グループは、この複合型のコンテンツモデル内で表示されています。 下のスクリーンショット内では、 `Contact` 属性 グループがデフォルトの属性 グループで、 グループ 要素上で自動的に使用することができます(*上のスクリーンショット参照も参照してください*)。属性 グループを無効化するには、複合型 `defaultAttributesApply` を `false` に設定します。複合型の詳細入力ヘルパー内の *defAttrs* プロパティを使用してもこの設定を行うことができます (*下のスクリーンショットを参照*).



xpathDefaultNamespace 属性

`xpathDefaultNamespace` 属性は、スキーマ内で使用される XPath 式内の要素のためのデフォルトの名前空間を設定します。スキーマ設定 ダイアログ内で設定されている場合、トップレベル `xs:schema` 要素に属性が適用されます。宣言のスコープは、ドキュメント全体です。属性が許可されている箇所では、要素上の宣言が `xs:schema` 上の宣言を上書きすることが可能になります:

- `xs:assert` と `xs:assertion`
- `xs:alternative`
- `xs:selector` と `xs:` フィールド (識別制約)

上記の要素の入力ヘルパー内の XPath デフォルト名前空間を変更することができます。

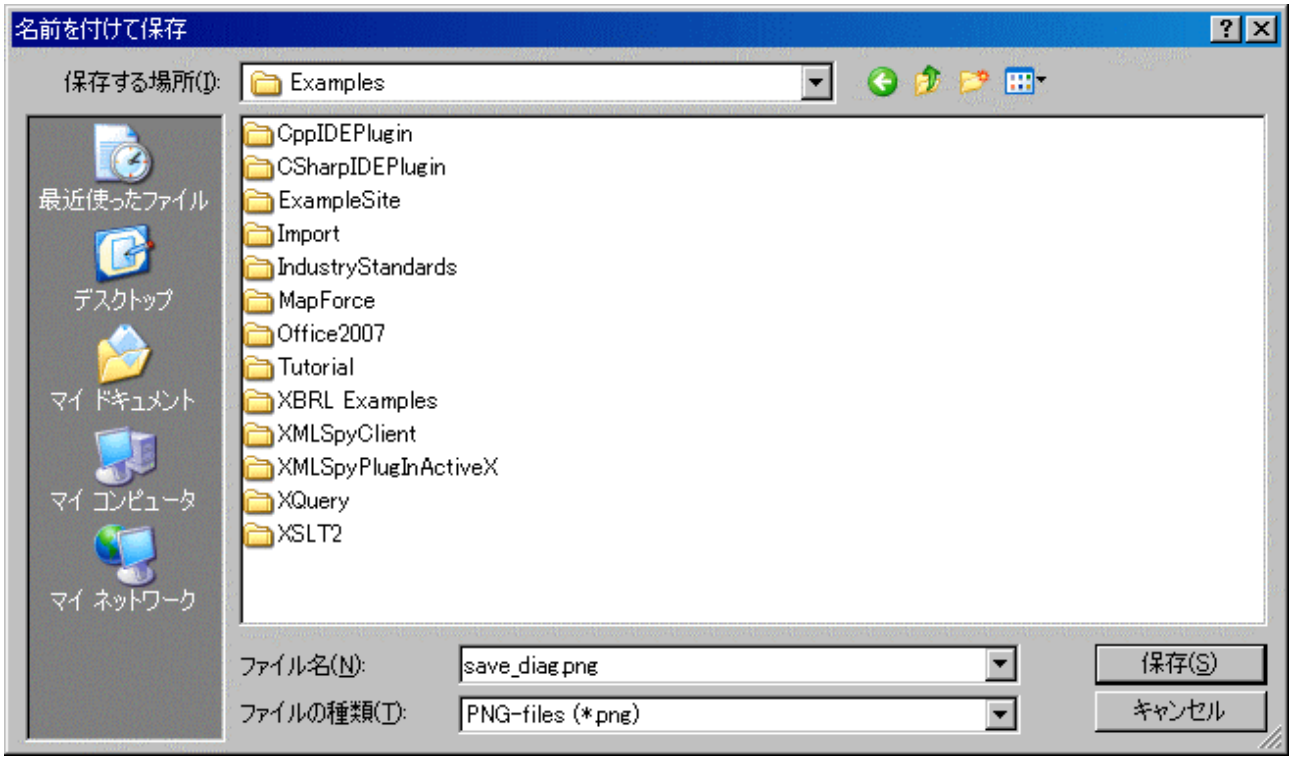
`xpathDefaultNamespace` 属性には3つの許可された値が存在します:

- `##targetNamespace`: XPath デフォルト名前空間 は、スキーマのターゲット名前空間と同じになります。
- `##defaultNamespace`: XPath デフォルト名前空間 は、スキーマのデフォルトの名前空間と同じになります。
- `##local`: XPath デフォルトの名前空間は存在しません。

XPath デフォルト名前空間がドキュメント内で宣言されていない場合、 XPath 式内のプレフィックスの無い要素には、名前空間は存在しません。 XPath デフォルト名前空間 宣言は、属性には適用されません。

28.7.2 図の保存

「**スキーマ設計 | 図の保存...**」コマンドにより、現在メインウィンドウに表示されているコンテンツモデルのダイアグラムを PNG フォーマットで、任意の場所に保存することができます。



28.7.3 ドキュメンテーションの生成

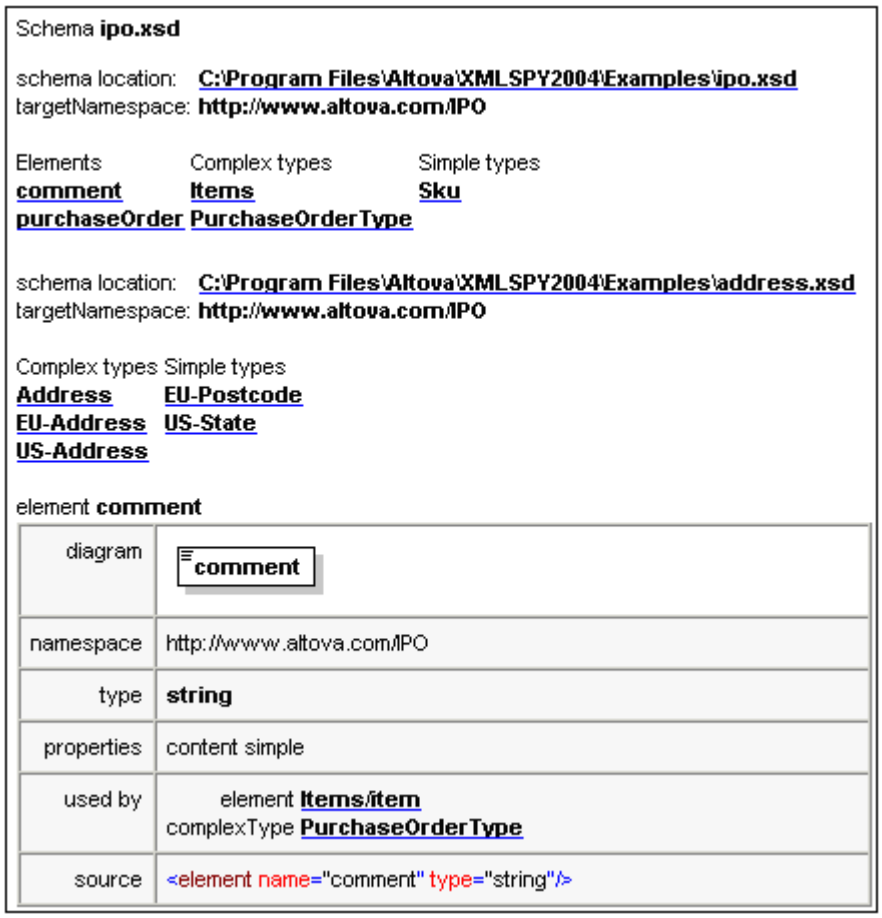
「**スキーマ設計 | ドキュメンテーションの生成...**」コマンドにより、スキーマに関する詳細なドキュメンテーションを HTML、MS W ord、RTF、または PDF 形式で生成することができます（以下のスクリーンショットを参照）。このコマンドにより生成されたドキュメンテーションは、自由に使用、変更することができ、Altova から許可を得る必要はありません。（ドキュメンテーションの生成コマンドを選択したときに表示される）スキーマドキュメンテーションダイアログにて選択したコンポーネントに関するドキュメンテーションが生成されます。子要素や複合型といった関連する要素は、通常出力されたメディアのファイルにてハイパーリンク化されており、コンポーネント間のナビゲーションを行うことができるようになります。コンテンツモデルを持つコンポーネントにもコンテンツモデルの定義に対するリンクが作成されます。スキーマのドキュメンテーションは**インクルード**ならびに**インポートされたスキーマコンポーネント**に対しても生成されることに注意してください。ドキュメンテーションの生成に関する様々なオプションについては、ドキュメンテーション オプション[1261]にて記述されます。JSON スキーマドキュメンテーションは、JSON スキーマドキュメンテーションの生成[659]のセクションで説明されています。

ドキュメンテーションのオプション[1261] は、スキーマの表示構成ダイアログ[1264] 内で指定されている設定に加え適用されます

メモ: MS Word フォーマットにてドキュメンテーションの生成を行うには、お使いのコンピューターに MS Word（バージョン 2000 以降）をインストールする必要があります。

ドキュメンテーションの生成には XMLSpy の標準デザインを使用することのできるほか、StyleVision SPS によるデザインを行うこともできます。StyleVision SPS を使用することで、生成されるドキュメンテーションのデザインや、PDF による出力を行うこともできるようになります。SPS の使用方法に関しては、ユーザー定義デザイン[1263] のセクションを参照ください。

メモ: SPS を使ったスキーマドキュメンテーションの生成を行うには、お使いのコンピューターに StyleVision をインストールする必要があります。


Schema **ipo.xsd**

schema location: **C:\Program Files\Altova\XMLSPY2004\Examples\ipo.xsd**
targetNamespace: **http://www.altova.com/IPO**

Elements	Complex types	Simple types
comment	**Items**	**Sku**
purchaseOrder	**PurchaseOrderType**	

schema location: **C:\Program Files\Altova\XMLSPY2004\Examples\address.xsd**
targetNamespace: **http://www.altova.com/IPO**

Complex types	Simple types
Address	**EU-Postcode**
EU-Address	**US-State**
US-Address	

element **comment**

diagram	comment
namespace	http://www.altova.com/IPO
type	**string**
properties	content simple
used by	element **Items/item** complexType **PurchaseOrderType**
source	<element name="comment" type="string"/>

上のスクリーンショットは生成されたスキーマドキュメンテーションの例で、インデックス（関連するスキーマとグローバルコンポーネント）がドキュメンテーションの上部に表示されています。

メモ: :W3C スキーマドキュメントに対するドキュメンテーションの生成には、XMLSpy アプリケーション内部に保存されているバージョンが使用されます。その結果、外部で新たに編集されたスキーマに関する情報は、ドキュメンテーションに反映されません。

(C) 2015-2021 Altova GmbH

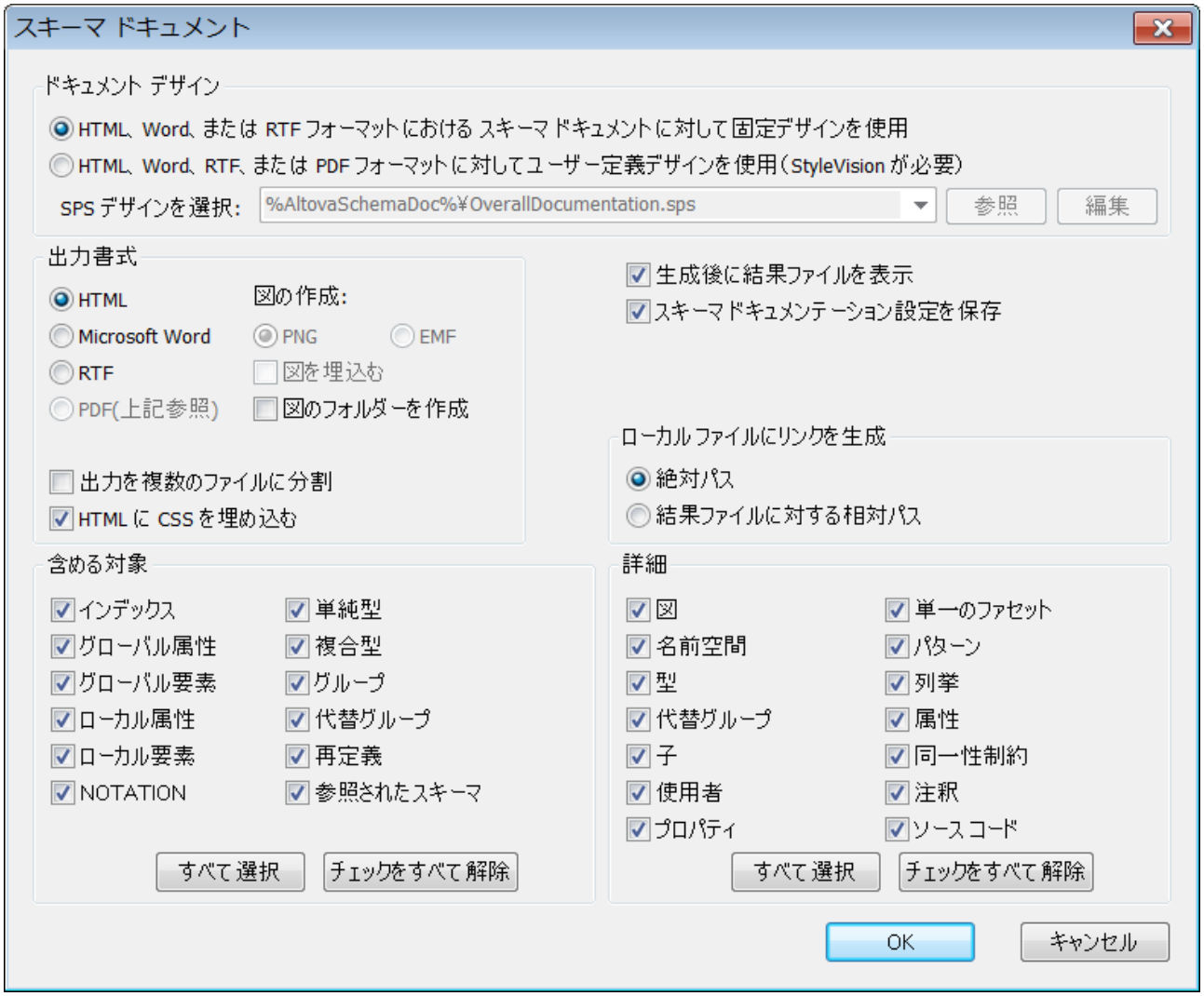
28.7.3.1 ドキュメンテーションオプション

スキーマデザイン | ドキュメンテーションの生成 コマンドは、アクティブなスキーマの詳細なドキュメントを生成します: XML スキーマまたは JSON スキーマ。このセクションでは、XML スキーマドキュメンテーションの生成について説明されています。 スキーマドキュメンテーションの生成の過程に類似しています。JSON スキーマドキュメンテーションの生成と、ドキュメント生成設定の詳細に関しては、以下のセクションを参照してください: JSON スキーマドキュメンテーションの生成 659.

スキーマドキュメンテーションの生成

XML スキーマドキュメントがアクティブで、「**ドキュメンテーションの生成**」コマンドを選択すると、スキーマドキュメンテーションダイアログが表示され、ドキュメンテーションに関するオプションを選択することができます(以下のスクリーンショットを参照)。

ダイアログのドキュメンテーションデザインペインでは、ドキュメンテーションの生成に XMLSpy の固定デザインを使用するか、StyleVision SPS により作成されたカスタムデザインを使用するか選択することができます。目的のオプションを選択してください。PDF の出力は、StyleVision SPS を使ってドキュメンテーションの生成でしか有効にならないことに注意してください。ユーザー定義されたデザインの使用方法については、ユーザーにより定義されたデザイン 1263 のセクションを参照ください。



以下に*スキーマドキュメンテーションダイアログ*にて設定することにできるオプションについて記します:

- 出力書式ペインにて、HTML、Microsoft Word、RTF、または PDF から目的のフォーマットを指定します(PDF 出力書式は StyleVision SPS を使ったドキュメンテーションの生成でしか使用できません)。OK をクリックすると、出力ファイルの名前と場所を入力するよう求められます。
- 固定デザインを使用した場合、.doc ファイル拡張子の Microsoft Word ドキュメントが生成され、StyleVision SPS を使った生成を行うと、.docx ファイル拡張子のドキュメントが生成されます。
- ドキュメンテーションは単一のファイルとして生成することも、複数のファイルに分割して生成することもできます。複数のファイルが生成される場合、各ファイルが個々のコンポーネントに対応します。出力ファイルに含められるコンポーネントは、*含める対象* ペインにて指定することができます。固定デザインを使用する場合、ドキュメント間には自動的にリンクが作成されます。
- HTML 出力では、CSS のスタイル定義を、外部 CSS ファイル内に保存するか、HTML ファイル内(<head> 要素の内部)に保存するか選択することができます。外部 CSS ファイルを作成する場合、.css という拡張子を持った HTML ファイルと同名のファイルが作成されます。*HTML に CSS を埋めこむ* オプションにより、オプションの指定を行うことができます。
- *図を埋めこむ*オプションはMS W ord、RTF、PDF 出力に対して有効になります。このオプションが有効になっていると、ダイアグラムが PNG またはEMP 形式でファイルの中に埋めこまれます。このオプションが指定されていない場合、PNG ならびにEMP ファイルが作成され、オブジェクトリンクを使ってファイルの中に表示されます。

- HTML 形式の出力が指定された場合、全てのダイアグラムは外部の PNG ファイルとして作成されます。*図のフォルダーを作成* チェックボックスにチェックが入っている場合、HTML ファイルと同じ場所にフォルダーが作成され、PNG ファイルがそのフォルダー内部に保存されます。このフォルダーの名前は、*HTMLFilename*_diagrams になります。*図のフォルダーを作成* チェックボックスにチェックが入っていない場合、PNG ファイルは HTML ファイルと同じ場所（フォルダー）に保存されます。
- *ダイアグラムのイメージや外部 CSS ファイルといったローカルファイルへのリンクは*、相対パスか絶対パスか指定することができます。*ローカルファイルにリンクを作成* ペインにて、適切なラジオボタンを選択してください。
- *含める対象* ペインでは、ドキュメンテーションに含めるアイテムを選択します。*インデックスオプション*により、関係する全てのスキーマが、グローバルコンポーネントとともにファイル上部に表示されます。**全てチェック**と**全てクリア**ボタンにより、ペインにある全てのオプションを選択または選択解除することができます。含むオプションは、視覚的な定義内のアイテムの表示に影響を与えません。 スキーマ表示の構成 1264 ダイアログにより視覚的な定義内のアイテムは管理されています。 スキーマアイテムのグラフィカルな表示内の属性の表示を無効化する場合、スキーマ表示の構成 1264 ダイアログ内の属性オプションのチェックを解除してください。
- *詳細* ペインでは、各コンポーネントに含める詳細情報を指定します。ドキュメンテーションに含める項目を選択してください。**全てチェック**と**全てクリア**ボタンにより、ペインにある全てのオプションを選択または選択解除することがで
- *生成後にファイルを表示* オプションは全ての出力書式に対して使用することができます。このオプションにチェックが入っていると、生成されたファイルがブラウザビュー（HTML 出力）、MS W ord（MS W ord 出力）、.rtf ファイルや .pdf ファイルの（それぞれ RTF 出力と PDF 出力）デフォルトアプリケーションにて表示されます。ドキュメンテーション内のリンクを相対パスにするか、絶対パスにするか選択することもできます。

パラメーター値

StyleVision SPS にて以上のパラメーター定義が含まれる場合、**OK** をクリックした際に SPS にて定義された全てのパラメーターがダイアログボックスにて表示されます。このダイアログにてパラメーター値を入力すると、SPS に割り当てられたデフォルトのパラメーター値をオーバーライドすることができます。

28.7.3.2 ユーザーにより定義されたデザイン

標準で用意されている XMLSpy の固定デザインの代わりに、カスタマイズしたスキーマドキュメンテーションのデザインを作成することができます。カスタムデザインは、出力ドキュメントのためのデザインテンプレートである StyleVision SPS 内で作成することができます。

SPS の作成

StyleVision Power Stylesheet（SPS）は、Altova の StyleVision により作成することができます。スキーマドキュメンテーションを生成するための SPS は、スキーマドキュメンテーションの構造を指定した XML スキーマをベースにする必要があります。このスキーマは SchemaDocumentation.xsd という名前で、XMLSpy のパッケージに同封されています。ファイルは以下のフォルダー以下に収められています：`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2021\Documentation\Schema`.

StyleVision にて SPS を作成する場合、`SchemaDocumentation.xsd` スキーマから得られたノードがデザインテンプレートに表示され、スタイルとプロパティが割り当てられます。リンクやテーブル、イメージといった追加コンポーネントも SPS デザインに追加することができます。このようにして、出力ドキュメント全体を SPS にてデザインすることができます。StyleVision における SPS デザインの作成方法に関する詳細は、StyleVision のユーザーマニュアルを参照ください。

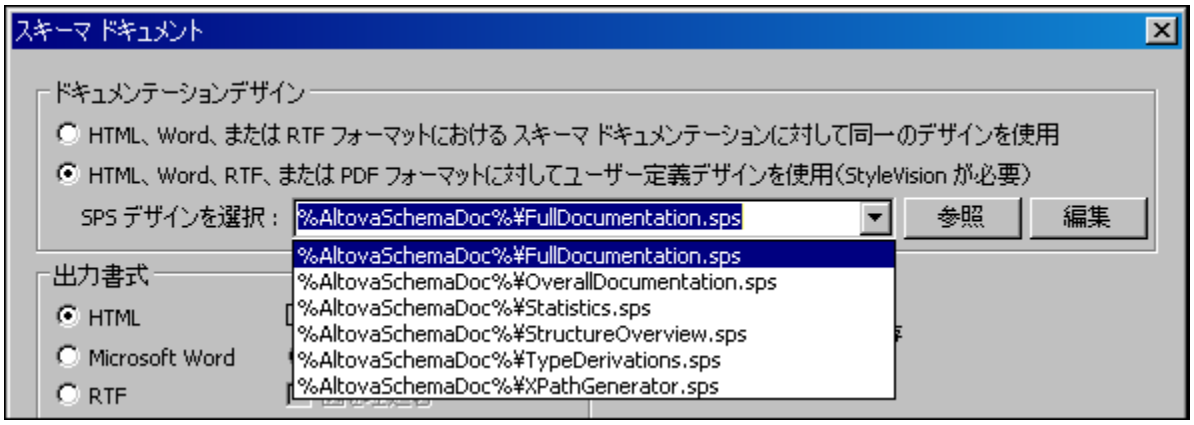
SPS によるスキーマドキュメンテーションの生成により、ドキュメンテーションのデザインを完全にコントロールすることができます。ユーザー定義の SPS を使用することで、PDF によるスキーマドキュメンテーションの生成を行うこともできます。XMLSpy の固定デザインを使用する場合、PDF 出力オプションを選択することはできません。

スキーマドキュメンテーションに使用する SPS の指定

SPS が作成されると、スキーマドキュメンテーションの生成に使用することができます。スキーマドキュメンテーションの生成に使用する SPS は、（「**スキーマ設計 | ドキュメンテーションの生成**」コマンドからアクセスすることができる）スキーマドキュメンテーションダイアログにて選択することができます。ダイアログのドキュメンテーションデザインペインにて、*ユーザー定義デザインを使用* ラジオボタンを選択します（*以下のスクリー*

ンショットを参照)。参照ボタンをクリックすることで、目的のSPS ファイルを選択することができます。ダイアログの「OK」ボタンをクリックして、表示される保存ダイアログにて出力ファイルの保存先フォルダーを選択、ファイル名を入力します。

メモ SPS ファイルでは、ベースとなっている次のスキーマを正しく指定する必要があります: `SchemaDocumentation.xsd` (*上記参照*)。



XMLSpyを使用して次の編集可能な SPS デザインの生成を行うことができます `Altova\XMLSpy2021\Documentation\Schema\` subfolder of the マイドキュメントフォルダー[33]内で見つけることができます:

- `OverallDocumentation.sps`:スキーマに関する完全なドキュメンテーションを生成します。
- `Statistics.sps` :グローバルやローカル要素、属性、属性グループ、単純型、複合型を、メインスキーマと関連する各スキーマごとに表示します。
- `StructureOverview.sps` は構成できる深さまでグローバル要素の構造と複合型の構造ををを生成します。
- `TypeDerivations.sps` は単純型および複合型をツリーの書式で直接的にまたは間接的に生成しリストします。
- `XPathGenerator.sps`, は構成可能な深さまで可能なすべてのXPath ステートメントを生成します。

これらのファイルは他のSPS ファイルと共に *ユーザー定義を使用* オプションのコンボボックス内で使用できるようになります(*上のスクリーンショットを参照*)。

ドキュメントデザインパネル内の「編集」ボタンのクリックするとStyleVision は選択されたSPS をStyleVision ウィンドウ内で開きます。StyleVision 内で結果ドキュメントをプレビューするには、作業 XML ファイルが必要になります。上にリストされるSPS デザインは既に `Sample.xml` という名前のサンプルXML ファイルに割り当てられています。このファイルはマイドキュメントフォルダー[33], 内の以下のサブフォルダー内で見つけることができます:

```
Altova\XMLSpy2021\Documentation\Schema\SampleData
```

メモ SPS によるスキーマドキュメンテーションの生成には、お使いのコンピューターにStyleVision をインストールする必要があります。

28.7.4 表示設定

「スキーマデザイン | ビューの構成」コマンドは、コンテンツモデルビューで有効化されており、コンテンツモデルビューの構成を行うことができます。コマンドをクリックすると、XMLSpy ウィンドウの右下のスキーマ表示の構成 ダイアログが開かれ、ダイアログ内に設定が入力されると、その効果を確認することができます。ダイアログの「OK」ボタンをクリックする、ダイアログ設定が有効になり、XML スキーマファイルのコンテンツモデルビューに対して適用されます。これらの設定は、スキーマドキュメンテーション出力[1259]と印刷出力にも適用されます。例えば、スキーマドキュメンテーション出力内のスキーマアイテムのグラフィカルな表示内の属性の表示を無効化する場合、スキーマ表示の構成[1264] ダイアログ内の属性オプションのチェックを解除してください(*下のスクリーンショット*)。

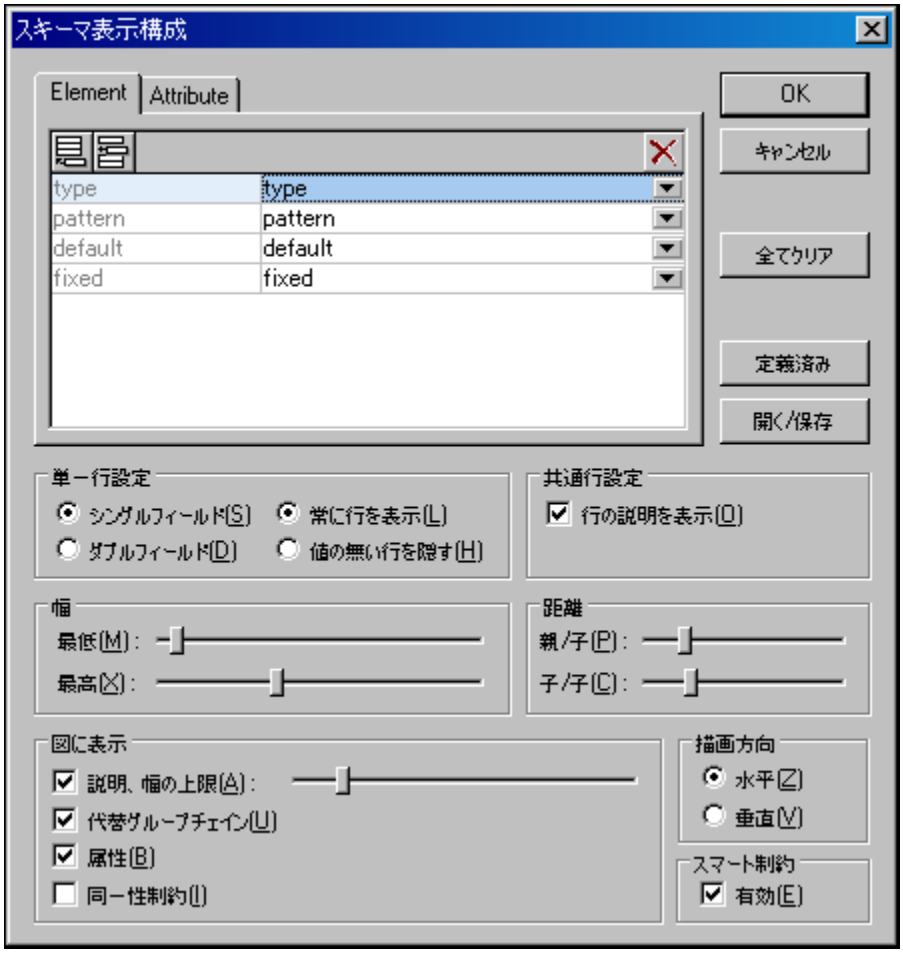
メモ: JSON の構成の方法に関しては、以下のセクションを参照してください: JSON スキーマデザインビューの構成[657]。

コンテンツモデルのためにプロパティ記述子 ラインを定義する

コンテンツモデルビューに表示される要素ならびに属性のプロパティを定義することができます。これらプロパティは、コンポーネントボックスにてグリッド線とともに表示されます。

プロパティ記述子 ラインを定義するには、以下を行います:

1. 「スキーマデザイン | ビューの構成」を選択します。スキーマ表示の構成 ダイアログが表示されます。
2. 「要素」または「属性」タブ内で、追加 または 挿入 アイコンをクリックし、プロパティ記述子ラインを追加します。ラインがダイアログ、およびコンテンツモデルビュー内の要素ボックスに追加されます。
3. コンボボックスから、表示するプロパティを選択します。*スクリーンショットを参照してください*。
4. ステップ 2 と3 を必要に応じて繰り返してください。



コンテンツモデルビューが更新され、定義されたプロパティ記述行が、要素に対して表示されます。

メモ: 属性のプロパティは、ダイアグラムに属性が表示されている時だけ表示されます(コンテンツモデルビュー以下にあるペインに属性が表示されている場合、意味を持ちません) ビューの構成が定義された後、開かれている全てのコンテンツモデルビューに対して設定が適用されます。

コンテンツモデルビューからプロパティ記述子ラインを削除する

個々のプロパティ記述行を削除するには、スキーマ表示構成ダイアログにて削除するプロパティ記述行を選択し、削除アイコン をクリックします。

コンテンツモデルビューの構成のための設定

コンテンツモデルビューの設定は、スキーマ表示構成ダイアログにて行うことができます。プロパティ記述行の表示については上記を参照ください。以下はその他の設定に関する記述です。

(C) 2015-2021 Altova GmbH

単一行の設定
プロパティの記述に単一もしくは複数のコンテンツを表示するのか、個々の行が全ての要素に表示されるのか（それともプロパティが含まれる場合だけ表示されるのか）指定することができます。適切なラジオボタンを使用することで、設定の定義を行うことができます。これら2つの設定は、各行に対して個別に指定することができます（目的の行を選択して、設定を行ってください）。

共有行の設定
このオプションでは、行の説明を表示するか指定することができます。

幅
スライダーにより、コンテンツモデルビューに表示される要素長方形の最小ならびに最大サイズを指定することができます。記述が全て表示されないときや、表示を標準化するのにこの値を変更してください。

距離
スライダーにより、画面上に表示されている各要素間の距離を指定することができます。

図に表示
説明チェックボックスでは、説明テキストの表示を指定したり、その幅をスライダーにより指定することができます。代替グループチェインの表示を指定することもできます。属性ならびに同一性制約のチェックボックスにチェックが入っていれば、それらのボックスがコンテンツモデルビューにそれぞれ表示され、チェックが入っていない場合は、コンテンツモデルビューの下部にあるペインのタブとして表示されます。

描写方向'
画面上に表示される属性ツリーの描画方向を、水平または垂直方向に指定することができます。

ダイアグラムにてコンテンツモデルの編集E
要素のプロパティは、コンテンツモデルダイアグラムにて直接変更することができます。編集したいプロパティをダブルクリックして、データを入力してください。選択が有効になっている場合、ドロップダウンリストが表示され、オプションの選択を行うことができます。それ以外の場合は、値を入力して「Enter」にて確定します。

スキーマ表示の構成 ダイアログ内のボタン
ダイアログには以下のボタンが用意されています:

- 開く/保存ボタンにて、ここでなされた設定のロードや保存を行うことができます。
- 定義済みボタンにて、表示構成をデフォルト値に戻します。
- 全てクリアボタンにより、全エントリーのリストを空にします。

スマート制約の有効化
スマート制約[223]を有効にするには、スマート制約の有効チェックボックスにチェックを入れてください。

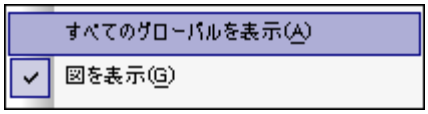
28.7.5 ズーム

「スキーマ設計 | ズーム」コマンドにより、コンテンツモデルビューの表示倍率を変更することができます。巨大なコンテンツモデルの全体をズームアウトして、メインウィンドウに表示するような場合に、この機能を使用することができます。ズームは、実際の大きさの10% から200% の間で行うことができます。

ズームインやズームアウトを行うには、スライダーをドラッグするか、入力ボックスをクリックして、パーセンテージの値を入力してください。

28.7.6 全てのグローバルを表示

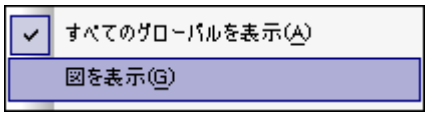
「**スキーマ設計 | 全てのグローバルを表示**」コマンドにより、コンテンツモデルビュー[176]からスキーマ概要[164]へ表示を切り替え、スキーマ内にある全てのグローバルコンポーネントを表示することができます。このコマンドは、図を表示コマンドと対になっており、現在選択されている表示方法が、チェックマークにより示されます(スクリーンショットを参照)。



その他にも、スキーマ概要の上部にある**全てのグローバルを表示**アイコン を使用して、スキーマ概要への切り替えを行うことができます。

28.7.7 図を表示

「**スキーマ設計 | 図を表示**」コマンドにより、(選択されたコンポーネントにコンテンツモデルがある場合)選択されたグローバルコンポーネントのコンテンツモデルビュー[176]への切り替えを行います。コンテンツモデルが含まれるグローバルコンポーネント(複合型、要素、要素グループ)は、 アイコンが左側に表示されます。このコマンドは、全てのグローバルを表示コマンドと対になっており、現在選択されている表示方法が、チェックマークにより示されます(スクリーンショットを参照)。



その他にも、以下の方法を使用することでコンテンツモデルビューへの切り替えを行うことができます:

- コンテンツモデルを表示したいコンポーネントの隣にある アイコンをクリック
- (画面右上にある)コンポーネントナビゲーション入力ヘルパーに表示されているコンポーネント名をダブルクリック

28.7.8 データベースに対するスキーマ拡張

このメニューコマンドでは、Oracle ならびにMS SQL Server スキーマ拡張に対するコマンドが収められています。

- Oracle スキーマ拡張を有効にする[1269]

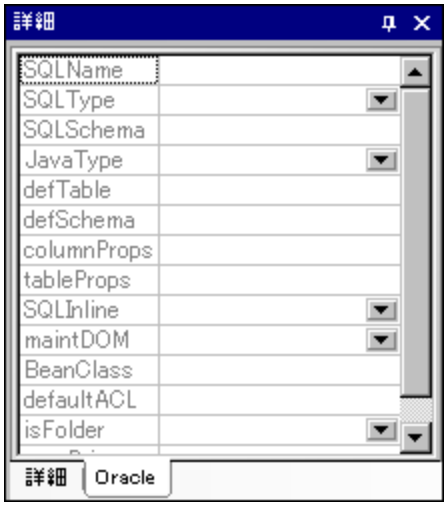
- Oracle スキーマの設定 1270
- Microsoft SQL Server スキーマ拡張を有効にする 1270
- 名前付きスキーマリレーションシップ 1271
- 名前なし要素のリレーションシップ 1271

28.7.8.1 Oracle スキーマ拡張を有効にする

XMLSpy では、Oracle 9i プロジェクト XDB にて使用することのできる Oracle スキーマの拡張がサポートされます。これらスキーマの拡張を使用することで、Oracle 9i プロジェクト XDB にて XML ドキュメントが保管される方法を設定することができます。これら XML ドキュメントへは SQL の問い合わせやそれまで使用されてきたツールによりアクセスすることができます。詳細については Oracle ウェブサイト を参照ください。

「**スキーマ設計 | データベースのスキーマ拡張 | Oracle スキーマ拡張を有効にする**」コマンドを選択すると、以下のことが起こります:

- XDB 名前空間が schema 要素に対して宣言されます: xmlns:xdb="http://xmlns.oracle.com/xdb"
- Oracle タブが詳細入力ヘルパーに表示され、属性(XDB 固有の属性も含む)を xsd:complexType や xsd:element といったスキーマ要素に加えることができるようになります。

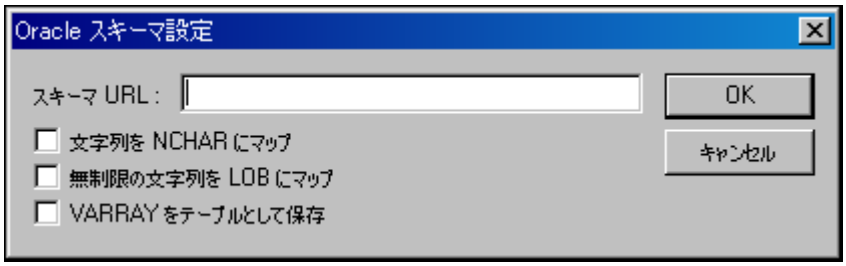


Oracle の拡張は、複合型、要素、そして属性に対して定義することができます。入力ヘルパーは、XMLSpy にあるその他の入力ヘルパーと同じように使用することができます。

メモ: このメニューコマンドはオンやオフに切り替えることが可能で、拡張を有効または無効にすることができます。Oracle 拡張が有効になっている場合、コマンドの左にチェックマークが表示されます。(有効になっているコマンドをクリックすることで) Oracle 拡張を無効にすると、XDB 名前空間ならびにファイル内にある全ての XDB 拡張が削除されます。この操作はやり直すことができないため、警告メッセージが表示されます。

28.7.8.2 Oracle スキーマの設定

「**スキーマ設計 | Oracle スキーマの設定**」コマンドにより、Oracle スキーマ拡張に対してグローバルな設定を定義することができます。



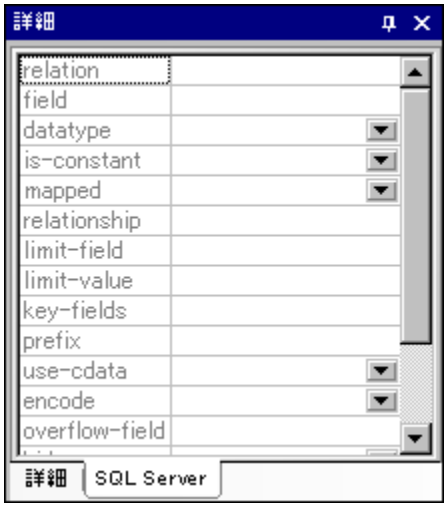
このダイアログへアクセスするには、（ Oracle スキーマ拡張を有効にする[1269] コマンドにより）Oracle スキーマ拡張が有効になっている必要があります。

28.7.8.3 Microsoft SQL Server スキーマ拡張を有効にする

XMLSpy では Microsoft SQL Server にて使用することができる Microsoft SQL Server 2000 スキーマ拡張がサポートされます。これらスキーマ拡張を使用することで、Microsoft SQL Server にて XML ドキュメントが保管される方法を設定、カスタマイズすることができます。これら XML ドキュメントへは SQL の問い合わせや、それまで使用されてきたツールによりアクセスすることができます。詳細については Microsoft ウェブサイト を参照ください。

「**スキーマ設計 | Microsoft SQL Server スキーマ拡張を有効にする**」コマンドを選択すると、以下のことが起こります:

- SQL Server 名前空間が schema 要素に対して宣言されます : xmlns:sql="urn:schemas-microsoft-com:mapping-schema"
- SQL Server タブが詳細入力ヘルパーに表示され、属性や xsd:element といったスキーマ要素を追加することができるようになります。



SQL Server 拡張は、スキーマコンポーネントに対して定義することができ、コンポーネントが選択された時、詳細入力ヘルパーに表示されます。入力ヘルパーは、XMLSpy にあるその他の入力ヘルパーと同じように使用することができます。

メモ: このメニューコマンドはオンまたはオフに切り替えることが可能で、拡張を有効または無効にすることができます。SQL Server 拡張が有効になっている場合、コマンドの右にチェックマークが表示されます。(有効になっているコマンドをクリックすることで)SQL Server 拡張を無効にすると、SQL Server 名前空間宣言ならびにファイル内にある全てのSQL 拡張が削除されます。この操作はやり直すことができないため、警告メッセージが表示されます。

28.7.8.4 名前付きスキーマリレーションシップ

「**スキーマ設計 | 名前付きスキーマリレーションシップ**」コマンドにより、名前付きリレーションシップの定義から、ドキュメント階層の作成に必要な情報を得ることができるようになります。このメニューオプションへアクセスするには、「**SQL Server スキーマ拡張を有効にする**」メニューオプションによりSQL Server スキーマ拡張を前もって有効にしておかなければなりません。

名前なしスキーマリレーションシップを作成するには:

1. 挿入 または追加アイコン をクリックして、新たな行をダイアログボックスへ追加します。
2. フィールドをクリックして、対応するリレーションシップ名を入力します。
3. 「OK」をクリックして確認します。

この操作により、SQL リレーションシップ要素が作成され、名前空間宣言のすぐ後に配置されます。

メモ: 削除アイコン をクリックすることで、ダイアログボックスから行を削除することができます。

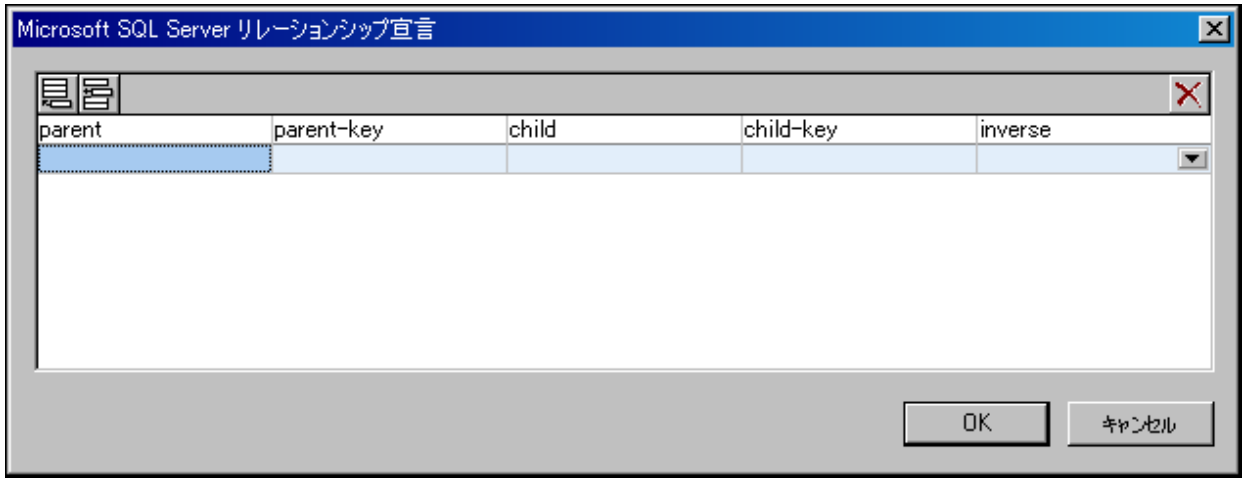
28.7.8.5 名前なし要素のリレーションシップ

名前なし要素リレーションシップコマンドにより、名前無しリレーションシップの定義から、ドキュメント階層の作成に必要な情報を得ることができるようになります。このメニューオプションへアクセスするには、「**SQL Server スキーマ拡張を有効にする**」メニューオプションによりSQL Server スキーマ拡張を前もって有効にしておかなければなりません。

名前なしスキーマリレーションシップを作成するには:

1. 挿入 または追加アイコン をクリックして、新たな行をダイアログボックスへ追加します。
2. フィールドをクリックして対応する名前を入力します。

3. 「OK」をクリックして確認します。

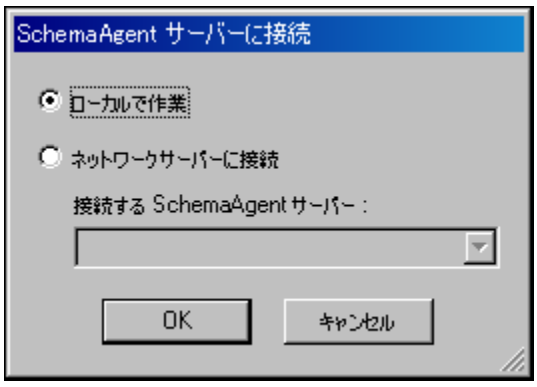


この操作により、SQL リレーションシップ要素が作成され、名前空間宣言のすぐ後に配置されます。

メモ: 削除アイコン をクリックすることで、ダイアログボックスから行を削除することができます。

28.7.9 SchemaAgent Server に接続

「スキーマ設計 | **SchemaAgent Server に接続**」コマンドは、XML スキーマドキュメントがアクティブなときに使用することができ、このコマンドにより SchemaAgent Server へ接続することができます。SchemaAgent Server への接続を行うには、お使いのコンピューターに SchemaAgent がインストールされていなければなりません。このコマンドをクリックすると、SchemaAgent サーバーへ接続ダイアログが表示されます(以下のスクリーンショットを参照):



ローカルのサーバー(Altova SchemaAgentに同封されている SchemaAgent サーバー)またはネットワークサーバー(無料で使用することができる Altova SchemaAgent Server 製品)を使用することができます。「**ローカルで作業**」を選択して「OK」をクリックすると、ローカルサーバーの SchemaAgent が起動され、接続が確立されます。「**ネットワークサーバーに接続**」を選択した場合、接続を確立するための SchemaAgent Server を前もって起動しておく必要があります。

SchemaAgent Server へ接続した場合、XMLSpy はSchemaAgentt クライアントとして動作して、パワフルなスキーマ編集ならびに管理機能が提供されます。SchemaAgent やSchemaAgent Server のインストール、そしてSchemaAgent Server への接続方法に関する詳細については、このユーザーマニュアルのDTD/XML スキーマセクション以下にある SchemaAgent[397] を参照ください。これら2つの製品のインストールや動作に関する更に詳細な情報については、製品に付属するSchemaAgent ユーザーマニュアルを参照ください。

SchemaAgent Server へ接続すると、メインウィンドウ上部のバーに、接続に関するメッセージが表示されます。これで、XMLSpy が接続しているSchemaAgent サーバーにて定義された検索パスやフォルダー内にある全てのスキーマやスキーマコンポーネントへのアクセスが行えるようになります。

注意：接続を行うには、XMLSpy がインストールされているコンピューターにて、有効なライセンスとともに、Altova のSchemaAgent Client 製品をインストールする必要があります。

28.7.10 SchemaAgent Server から切断

「**SchemaAgent Server から切断**」コマンドは、SchemaAgent Server への接続が正常に行われている際に有効になります。このコマンドを選択することで、XMLSpy がSchemaAgent Server から切断されます。

28.7.11 SchemaAgent で表示

「**SchemaAgent で表示**」メニューアイテムにより、アクティブなスキーマ（そしてオプションとして、リンクされているスキーマ）をAltova のSchemaAgent にて表示することができます。SchemaAgent の機能を使用するには、XMLSpy が動作しているコンピューターにこの製品をインストールする必要があります。スキーマがSchemaAgent のSchemaAgent Design にて新たに開かれます。

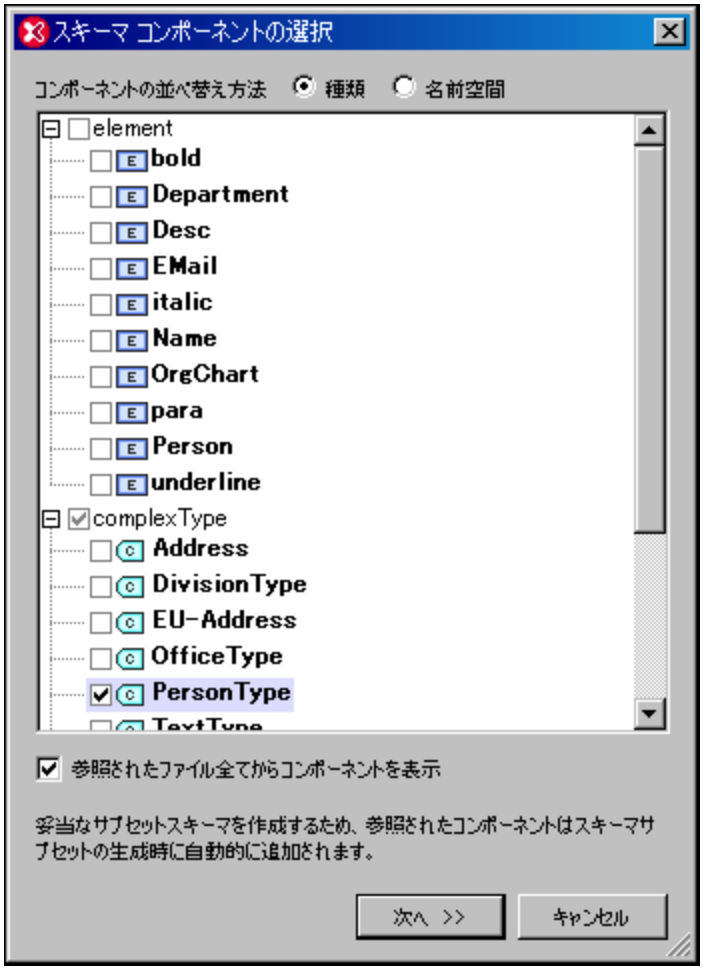
SchemaAgent で表示メニューアイテムにマウスオーバーすると、どのスキーマをSchemaAgent で表示するか選択するためのオプションがサブメニューにて表示されます。これらのオプションについては、このユーザーマニュアルのDTD/XML スキーマセクション以下にある SchemaAgent[406] を参照ください。

28.7.12 SchemaAgent 検証

「SchemaAgent 検証」コマンドにより、現在アクティブなスキーマと、現在アクティブなスキーマに関連したスキーマを検証することができます。この機能に関する詳細は、このユーザーマニュアルのスキーマビューセクション以下にあるSchemaAgent 検証[406] のセクションを参照ください。

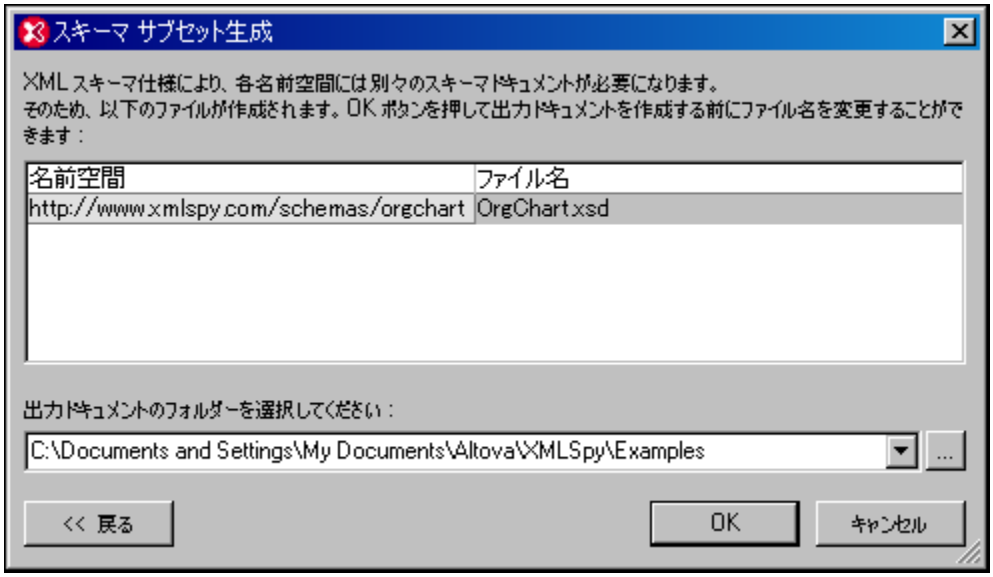
28.7.13　スキーマサブセットの作成

「**スキーマサブセットの作成**」コマンドにより、スキーマコンポーネントの選択ダイアログが表示されます（以下のスクリーンショットを参照）。このダイアログでは、単一のスキーマサブセットとして作成したいコンポーネントにチェックを入れて、**次へ**をクリックします。ペイン以下にあるチェックボックスにより、参照されている全てのファイルを表示、選択することができます。



表示されるスキーマサブセット生成ダイアログにて、スキーマサブセットパッケージのファイル名を入力します（以下のスクリーンショットを参照）。新たに作成されるスキーマサブセットファイルが保存される場所も指定する必要があります。作成されるコンポーネントが、オリジナルのスキーマにおいてインポートされたコンポーネントである場合は、スキーマサブセットパッケージを複数のファイルにより構成することもできます。異なるスキーマファイルが、スキーマサブセット内の各名前空間に対して作成されます。ダイアログペインに表示されるファイル名は、デフォルトでオリジナルのファイル名となっていますが、オリジナルのファイルを上書きすることはできないので、オリジナルのファイルと同じ場所に保存する場合は、新しいファイル名を入力してください。

(C) 2015-2021 Altova GmbH

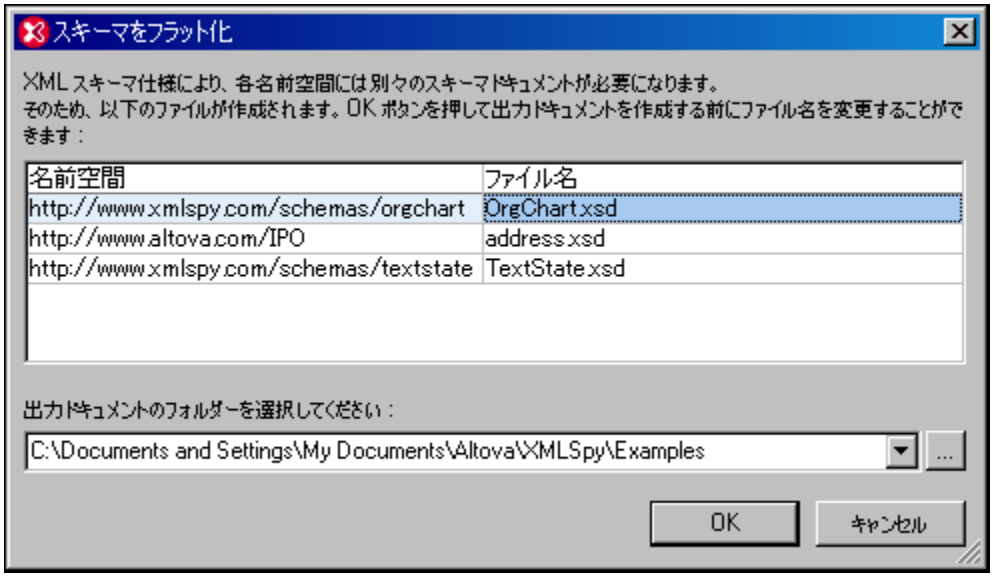


「OK」をクリックすると、アクティブなファイルに対応した名前空間を持つスキーマサブセットファイルが、スキーマビューで開かれます。パッケージ内にあるその他のファイルは、作成されますが、スキーマビューで開かれません。

28.7.14 スキーマのフラット化

スキーマビューに表示されているアクティブなスキーマのフラット化は、(i) インクルードされている全てのスキーマをグローバルコンポーネントとしてアクティブなスキーマに追加した後、(ii) インクルードされたスキーマを削除する、という処理により構成されます。

アクティブなスキーマをフラット化するには、「**スキーマ設計 | スキーマのフラット化**」コマンドを選択します。この操作によりスキーマをフラット化ダイアログが表示され、フラット化されたスキーマにおける各名前空間に対応したファイルの名前が表示されます。これらのデフォルト名は、オリジナルのファイル名と同じものですが、オリジナルのファイルを上書きすることはできないので、オリジナルのファイルと同じ場所に保存する場合は、新しいファイル名を入力してください。参照により、フラット化されたスキーマならびに関連するファイルが保存されるフォルダーを指定することができます。



「OK」をクリックすると、フラット化されたスキーマがスキーマビューにて開かれます。

(C) 2015-2021 Altova GmbH

28.8 XSL/XQuery メニュー

XSL 変換言語を使えば、どのようにXML ドキュメントが他のXML ドキュメントやテキストファイルへ変換されるかを指定することができます。XSLT ドキュメントにより生成されるXML ドキュメントの1つに、PDF の生成処理を行うために使用されるFO ドキュメントがあります。XMLSpy には(XSLT 1.0, XSLT 2.0, and XSLT 3.0 のための) XSLT プロセッサーが内蔵されており、使用中のシステムにあるFO プロセッサーと連携することで、XML ファイルから様々な形式の出力を生成することができます。XMLSpy インターフェースから、直接FO プロセッサーを使用するには、FO プロセッサーの場所をオプションダイアログ(「ツール| オプション」 1479)のXSL セクションにて指定する必要があります。

XMLSpy にはXQuery エンジンも内蔵されており、(XML ドキュメントへの参照無しでも)XQuery ドキュメントを実行することができます。

上記の変換を行うためのコマンドは、全て **XSL/XQuery** メニューに収められています。更に、このメニューからはAltova XSLT/XQuery デバッガーを使用するためのコマンドへもアクセスすることができます。

(C) 2015-2021 Altova GmbH

28.8.1 XSL 変換

F10

「XSL/XQuery | XSL 変換」コマンドにより、割り当てられたXSLT スタイルシートを使ってXMLドキュメントの変換を行うことができます。変換は、内蔵されている適切な Altova XSLT エンジン(XSLT 1.0 スタイルシートに対してはAltova XSLT 1.0 エンジンを、XSLT 2.0 スタイルシートに対してはAltova XSLT 2.0 エンジンを使用)、Microsoft から提供されているMSXML モジュール、または外部のXSLT プロセッサーを使用することで行うことができます。このコマンドにより使用されるプロセッサーは、オプションダイアログ(「ツール | オプション」)の XSL 1479 セクションにて指定することができます。

XMLドキュメントにXSLT スタイルシートへの参照が含まれている場合、このスタイルシートが変換に使用されます(XSL の割り当て 1287 コマンドを使うことで、XSLT スタイルシートをXMLドキュメントへ割り当てることができます。XMLドキュメントがプロジェクトに含まれている場合 プロジェクト プロパティ 1194 ダイアログにて、フォルダーごとにXSLT スタイルシートを指定することができます。変換を行うプロジェクトフォルダーやファイルを右クリックして「XSL 変換」を選択してください)。XSLT スタイルシートがXML ファイルに割り当てられていない場合、使用するXSLT スタイルシートを指定するよう促されます。ファイルの選択は(参照 1132 ボタンをクリックして)グローバルリソースやURL、(ウィンドウボタンをクリックして) XMLSpy にて既に開かれているファイルからも行うことができます。

RaptorXML 2021 を使用した自動検証

RaptorXML は、XML 検証、XSLT 変換、とXQuery 変換のためのAltova のスタンドアロンアプリケーションです。Java プログラムと.NET 内のCOM インターフェイスを使用して、アプリケーションコマンドラインから使用することができます。このため、RaptorXML を使用してXSLT 変換タスクを実行することができます。例えば、ドキュメントのセット上でXSLT 変換を実行するためにRaptorXML を呼び出すバッチファイルを作成し、テキストファイルに出力を送ることができます。詳細に関しては、RaptorXML ドキュメンテーション を参照ください。

ZIP ファイルへの変換

変換の結果を.docx 拡張子を持つOpen Office XML(OOXML) やZIP ファイルに収める場合は、出力ファイルのファイルパスに以下のようなZIP プロトコルを指定する必要があります:

```
filename.zip|zip/filename.xxx

filename.docx|zip/filename.xxx
```

メモ: 変換を行う前に、ディレクトリ構造の作成を行わなければならない場合もあります。Open Office XML アーカイブ形式で変換を行う場合、(例えば .docx のような)トップレベルの OOXML ファイルを作成するために、アーカイブファイルの ZIP 化を行う必要があります。

28.8.2 XSL スピードオプティマイザー

XSL スピードオプティマイザー コマンドは、XSLT またはXMLドキュメントがアクティブな場合、有効化されています。分析されたXSLT スタイルシートを使用して変換のスピードを向上することのできる可能性を分析するXSL スピードオプティマイザーを開始します。XML データセット(1つまたは複数のXMLドキュメント)とスタイルシートのパフォーマンスを分析することにより、XSLT スタイルシートが最適化されるよう

にして、オプティマイザーは作動します。最適化の戦略は、この分析より割り出され、(スタイルシートの終わりの処理命令として) XSLT スタイルシートに保存することができます。最適化されたスタイルシートは、後に、更に速い変換を生成するために使用することができます。

コマンドクリックすると、XSLT またはXML ドキュメントがアクティブであるかにより、それぞれXML ドキュメント またはXSLT スタイルシートを選択するようにプロンプトされます。「OK」をクリックすると、分析が開始されます。XSLT またはXML ドキュメントがそれぞれドキュメント内でXML 割り当て[1288] またはXSLT 割り当て[1287] を持つ場合、このステップはスキップされます。,オプティマイザーの使用については、以下のセクションを参照してください:XSL スピードオプティマイザー[440]。オプティマイザーの設定は、オプションダイアログ(「ツール | オプション」)内のXSL スピードオプティマイザータブ[1479] で行うことができます。

28.8.3 XSL-FO 変換

Ctrl+F10

FO は、印刷することを前提にしたドキュメントを記述するためのXML フォーマットです。Apache XML プロジェクトのFOP のようなFO プロセッサーでは、FO ファイルを入力することでPDF を出力として生成することができます。そのため、XML ドキュメントからPDF を生成するには2つのステップを踏むことになります。

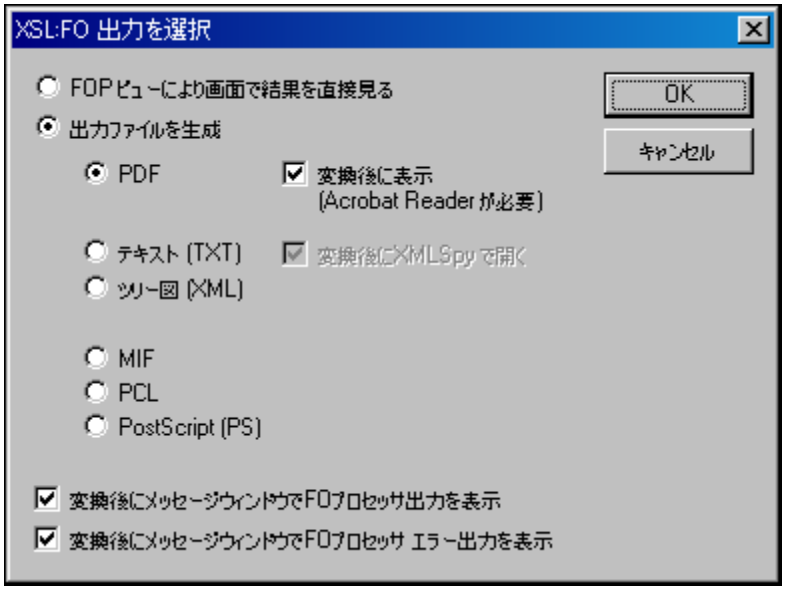
1. XSLT (または XSL-FO) スタイルシートを使うことで XML ドキュメントを FO ドキュメントへ変換します。
2. FO プロセッサーにより FO ドキュメントを処理することで、PDF(またはその他の出力)を生成します。

「XSLT/XQuery | XSL:FO 変換」コマンドによりXML ドキュメントやFO ドキュメントをPDF に変換します。

- XSL:FO 変換コマンドをソース XML ドキュメントに対して実行すると、上にある両方のステップが順に処理されます。FO への変換に必要な XSLT(または XSL-FO)スタイルシートが XML ドキュメントから参照されていない場合、変換に使用するスタイルシートを指定する必要があります(以下のスクリーンショットを参照)。ファイルの選択は(参照[1132] ボタンをクリックして) グローバルリソースやURL、(ウィンドウボタンをクリックして) XMLSpy にて既に開かれているファイルからも行うことができます。XML からXSL-FO への変換が、オプションダイアログ(「ツール | オプション」)のXSL タブ[1479] にて指定されたXSLT プロセッサーにより行われます。デフォルトでは、XMLSpy XSLT プロセッサーが選択されます。結果として生成されるFO ドキュメントが、オプションダイアログ(「ツール | オプション」)のXSL タブ[1479] にて指定されたFO プロセッサーにより変換されます。
- XSL:FO 変換コマンドがFO ドキュメントに対して実行されると、ドキュメントがオプションダイアログ(「ツール | オプション」)のXSL タブ[1479]にて指定されたFO プロセッサーにて処理されます。

XSL:FO 変換出力

XSL:FO 変換コマンドを選択すると、XSL:FO 出力を選択ダイアログが表示されます(以下のスクリーンショットを参照)。アクティブなドキュメントがXSLT 割り当てを持たないXML ドキュメントの場合、XSLT ファイルの選択を促されます。



FO プロセッサーの出力はFOP ビューアーを使用することで、画面上にて直接確認する他にも、以下のフォーマットにより出力ファイルを生成することができます:PDF、テキスト、XML エリアツリー、MIF PCL、またはPostScript。FO プロセッサーから得られたメッセージ出力するオプションにより、(i) プロセッサーの標準出力メッセージや(ii) プロセッサーのエラーメッセージをメッセージウィンドウにて表示することができます。これら2つのオプションを有効にするには、ダイアログ下部にある対応するチェックボックスにチェックを入れてください。

メモ:

- Apache XML プロジェクト のFOP プロセッサーをインストールするオプションを解除していない限り、次のフォルダーにインストールされます:`C:\ProgramData\Altova\SharedBetweenVersions`. インストールされると、パスは自動的にオプションダイアログ(**ツール | オプション**)内のXSL タブ[1479] に使用するFO プロセッサーとして入力されます。使用するFO プロセッサーにパスを設定することができます。
- XSL:変換コマンドは、メインウィンドウにてアクティブとなっているファイルだけではなく、アクティブなプロジェクトにて選択することのできるファイルやフォルダーに対しても使用することができます。右クリックで表示されるコンテキストメニューからXSL:FO 変換を選択してください。選択されたプロジェクトフォルダーに割り当てられたXSLT スタイルシートが使用されます。

28.8.4 XSL パラメーター / XQuery 実行

「**XSL/XQuery | XSL パラメーター/XQuery 変数**」コマンドにより、XSLT 入力パラメーター/XQuery 外部変数ダイアログが表示されます(*スクリーンショットを参照*)。XSLT スタイルシートにパスするパラメーターの名前や、XQuery ドキュメントへパスするXQuery 変数の名前と、対応する値を入力します。XMLSpy においてこれらのパラメーターは以下のように使用されます:

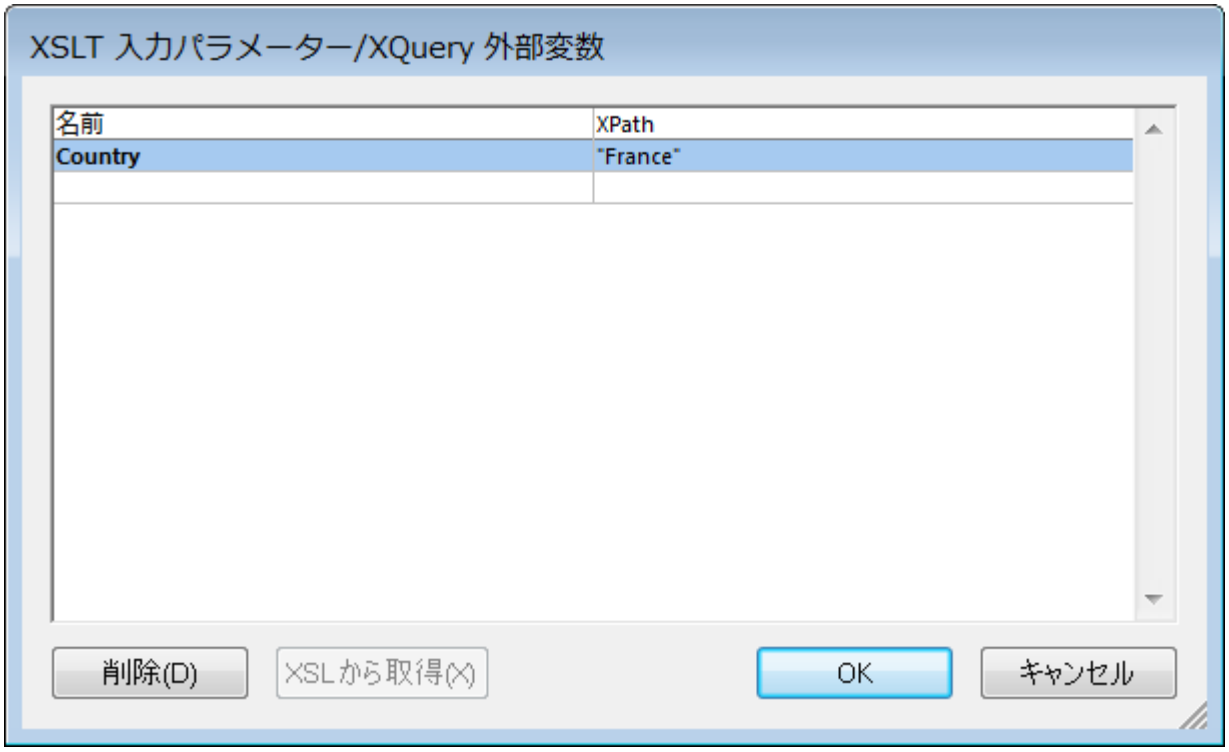
- XSL/XQuery メニューのXSL 変換コマンドがXML ドキュメントの変換に使用される際、ダイアログ内にて保存されたパラメーター値が選択されたXSLT ドキュメントへ渡され、変換に使用されます。
- XSL/XQuery メニューのXQuery 実行コマンドがXQuery ドキュメントの処理に使用される際、ダイアログにて現在保存されているXQuery 外部変数の値がXQuery ドキュメントに渡され、実行に使用されます。

メモ XSLT 入力パラメーター/XQuery 外部変数ダイアログにて入力されたパラメーターや変数は、内蔵のAltova XSLT エンジンに対してのみ使用されます。このため、MSXML 、またはは、構成済みの外部エンジンを使用する場合、これらの値はエンジンにパスされません。

メモ XSLT 入力パラメーター/XQuery 外部変数 ダイアログ内で XSLT パラメーター、または、外部 XQuery 変数 を定義することはエラーではありませんが、XSLT/XQuery ドキュメント、または、変換内では使用されません。

XSLT パラメーターの使用

パラメーターに対して入力した値は引用句無しの XPath 条件式や、引用句無しのテキスト文字列となります。アクティブなドキュメントが XSLT ドキュメントの場合、XSL から取得 ボタンが有効になります。このボタンをクリックすることで、XSLT にて宣言されている パラメーターが、デフォルト値とともにダイアログに入力されます。この操作により、宣言された パラメーターを素早く入力し、デフォルト値を必要に応じて変更することができます。



メモ パラメーター値がダイアログに入力されると、明示的に削除、または、アプリケーションが再起動されるまですべての変換で使用されます。ダイアログ内に入力される パラメーターはそのセッションのアプリケーションレベルで指定され、この時点から IDE を使用して実行される各変換のためのそれぞれの XSLT ドキュメントために パスされます。これは以下を意味します:

- パラメーターは特定のドキュメントに割り当てられているのではありません。
- ダイアログにて入力された パラメーターは、XMLSpy の終了時に削除されます。

XSLT パラメーターの使用例

国名とそれぞれの首都を含む XML ドキュメントは以下のとおりです:

```
<document>
   <countries>
      <country name="USA" capital="Washington DC"/>
      <country name="UK" capital="London"/>
      <country name="France" capital="Paris"/>
      <country name="Russia" capital="Moscow"/>
      <country name="China" capital="Beijing"/>
```

```
   </countries>
</document>
```

次のXSLTドキュメントは、XMLファイルから国名と首都を表示するXMLドキュメントを生成します。country という名前を持つパラメーターの値として名前を入力することにより国名は選択されます(*以下では黄色のハイライトにより表示されています*)。

```
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:param name="country" select="'USA'"/>
   <xsl:template match="countries">
      <xsl:for-each select="country[@name=$country]">
         <country>
            <name><xsl:value-of select="$country"/></name>
            <capital><xsl:value-of select="@capital"/></capital>
         </country>
      </xsl:for-each>
   </xsl:template>
</xsl:stylesheet>
```

このXSLTドキュメントが上記のXMLドキュメントで実行されると、結果は以下のようになります:

```
<country><name>USA</name><capital>Washington DC</capital></country>
```

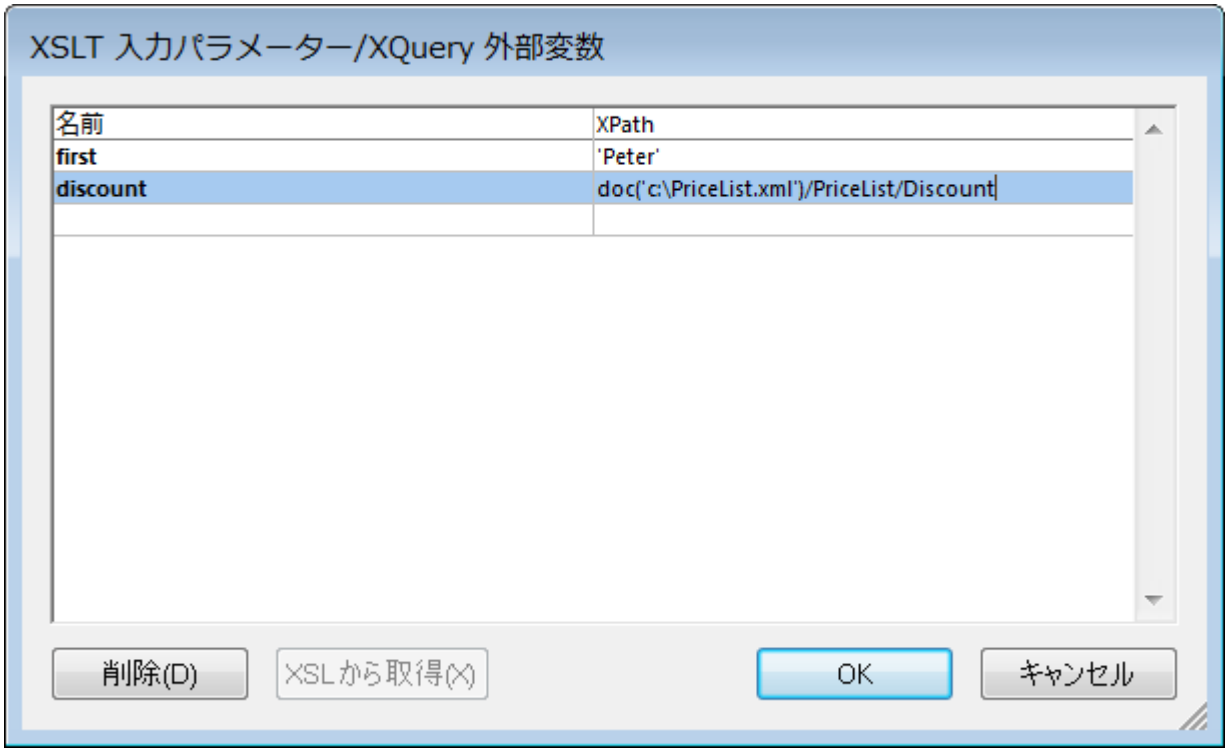
XSLT 入力パラメーター/XQuery 外部変数 ダイアログ内で country という名前のパラメーターが作成され、値が与えられると、新規のパラメーターの値は、XSLT 変換内のパラメーターにパスされます(*上のスクリーンショット参照*)。この値は変換のためにXSLT スタイルシート内のパラメーター country にパスされます。このようにして、異なるパラメーターへ異なる値をランタイムにパスすることができます。

メモ

- XSL:FO 変換コマンド(「XSL/XQuery | XSL:FO 変換」)を使用する場合、ダイアログにて入力された値はスタイルシートへ渡されません。これらのパラメーターをPDF 出力にて使用するには、まずXSLT 変換コマンド(「**XSL/XQuery | XSL 変換**」)を使用して XML から FO ドキュメントへの変換を行い、その後「XSL:FO 変換」コマンド(XSL/XQuery | XSL:FO T変換)によりFO からPDF への変換を行う必要があります。
- 内蔵されているAltova XSLT エンジン以外のXSLT プロセッサーを使用する場合、ダイアログにて入力されたパラメーターは外部プロセッサーへ渡されません。

外部 XQuery 変数の使用

外部 XQuery 変数に対して入力する値は、引用句無しのXPath 条件式か、引用句により分離されたテキスト文字列となります。外部変数のデータ型は、XQuery ドキュメントの変数宣言にて指定されます。



メモ 外部 XQuery 変数がダイアログにて一度入力されると、そのエントリーが明示的に削除されるかアプリケーションが再起動されるまで、その後の実行でもそれらの変数が使用され続けます。ダイアログにて入力された変数はアプリケーションレベルで指定されており、その後行われる実行のたびに関連する XQuery ドキュメントへ渡されます。これは以下を意味します:

- 変数は特定のドキュメントに割り当てられているのではありません。
- ダイアログにて入力された変数は、アプリケーション（XMLSpy）の終了時に削除されます。

外部 XQuery 変数の使用例

以下の例では、変数 `$first` が XQuery ドキュメントにて宣言され、FLW OR ステートメントの return 句にて使用されます:

```
xquery version "1.0";
declare variable $first as xs:string external;
let $last := "Jones"
return concat($first, " ", $last )
```

（XSLT 入力パラメーター/XQuery 外部変数ダイアログにて入力された）外部変数に `Peter` がセットされている場合、XQuery により `Peter Jones` という値が返されます。以下の点に注意してください:

- XQuery ドキュメントの変数宣言にある `external` キーワードにより、この変数が外部変数であるということが認識されます。
- 静的な変数の型定義はオプションです。変数の宣言時に変数のデータ型が定義されていない場合、変数の値は `xs:untypedAtomic` に割り当てられます。
- XQuery ドキュメントにて外部変数が宣言されているが、その変数名に対して外部変数が与えられない場合、エラーとなります。
- 外部変数が宣言され、XSLT 入力パラメーター/XQuery 外部変数ダイアログにて入力された場合、実行されるのは XQuery ドキュメントのスコープ内となります。XQuery ドキュメントにて新たな変数が同じ名前で宣言されている場合、その変数によりスコープ内の外部変数が一時的にオーバーライドされます。例えば、以下にある XQuery ドキュメントでは、（外部変数により `$first` に対して `Peter` という値が渡されているにも関わらず）`Paul Jones` という値が返されます。

```
xquery version "1.0";
```

```
declare variable $first as xs:string external;
let $first := "Paul"
let $last := "Jones"
return concat($first, " ", $last )
```

28.8.5 XQuery/Update 実行

XSL/XQuery | XQuery/Update 実行 コマンドにより XQuery (1.0/3.1) または XQuery Update (1.0/3.0) ドキュメントが実行されます。選択された XQuery または XQuery Update ファイルにより、XQuery 実行 または XQuery 更新が実行されます。XMLSpy は、ドキュメントの型 (XQuery または XQuery Update) を[オプションダイアログのファイル型](1458) セクション内で定義された[ファイル型との関連](1458) をベースにして認識します。

使用する XQuery エンジン (1.0 または 3.1) は、ドキュメント内のバージョンの宣言をベースに自動的に選択されます。ドキュメント内にバージョン宣言が存在しない場合、[オプションダイアログの XQuery タブ](1482) 内で指定されているデフォルトのバージョンが使用されます。XQuery、XQuery Update、または XML ファイルがアクティブな場合、**XQuery/Update 実行** コマンドが呼び出されます。XML ファイルから呼び出されると、XML ファイルと関連付けられる XQuery ファイルを要求するダイアログが開かれます。グローバルリソース、または、URL ([参照](1132) ボタンをクリックして) またはファイルを選択することにより、(**ウィンドウ** ボタンをクリックして) XMLSpy 内のウィンドウで開くことができます。
メモ: [プロジェクトウィンドウ](110) アイテムのコンテキストメニューでコマンドを使用することができます。

RaptorXML 2021 を使用した自動検証

RaptorXML は、XML 検証、XSLT 変換、と XQuery 変換のための Altova のスタンドアロンアプリケーションです。Java プログラムと .NET 内の COM インターフェイスを使用して、アプリケーションコマンドラインから使用することができます。このため、RaptorXML を使用して XQuery 実行タスクを実行することができます。例えば、ドキュメントのセット上で XQuery を実行するために RaptorXML を呼び出すバッチファイルを作成し、テキストファイルに出力を送ることができます。詳細に関しては、RaptorXML ドキュメンテーション を参照してください。

28.8.6 バックマッピングの有効化

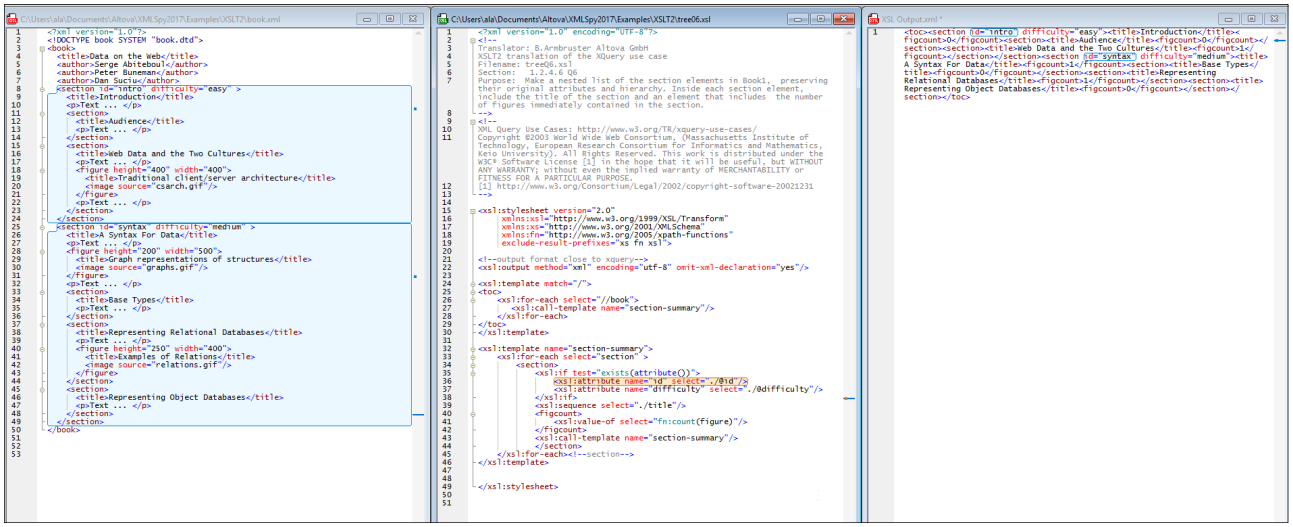
メインツールバーで使用することのできる、このコマンドは、バックマッピング機能をオンに切り替えます。

(このコマンドを使用して) バックマッピングが有効化されると、XSLT 変換 と XQuery 実行が実行され、結果ドキュメントが元の XSLT+XML、または、XQuery+XML ドキュメントにバックマップされます。これは、結果ドキュメント内のノードをクリックすると、その特定の結果ノードを生成した XSLT 命令 と XML ソースデータ がハイライトされます (*下のスクリーンショットを参照してください*)。XSLT 変換、または、XQuery 実行が結果ドキュメントの異なる箇所をどのように作成するかをチェックする際に役にたちます。XSLT/XQuery ドキュメント、または、XML ドキュメント内でノードをクリックし 2 つのドキュメントの対応する箇所を確認することもできます。

メモ HTML 以外の全ての型の結果ドキュメントはテキストビュー内で開くことができます。HTML 結果ドキュメントは、ブラウザービューで開かれますが、テキストビューに切り替えることができます。結果ドキュメントがブラウザービューで開かれると、結果ドキュメント内で選択することによりバックマッピングを使用することができます。XML、または、XSLT/XQuery ドキュメント内で選択する場合、バックマッピングを使用することはできません。

メモ Authentic View 526 または プロジェクト 変換 1194 として実行される変換のためには バックマッピング を使用することはできません。

下のスクリーンショットは XSLT 変換の バックマッピング を表示しています。3 つすべてのドキュメント― XML+XSLT+Result― は、上下に表示されています。section/@id 属性 in 結果ドキュメント 内で生成する XSLT 命令がクリックされています。この結果、この命令により生成された全ての結果ノードと、結果ノードを生成した XML ソースデータが ハイライト されています。結果ドキュメント、または、XML ドキュメント内のノードをクリックして他の 2 つのドキュメント内の対応するノードを ハイライト することもできます。



XSLT/XQuery バックマッピングの有効化 コマンドをクリックすると、変換後に ドキュメント ウィンドウの分割を問うダイアログが表示されます。このオプションを選択すると、3 つのドキュメント が上のスクリーンショットに表示されるように横に並べられて表示されます。

以下の点に注意してください:

- ディスクロケーションからロードされた XML ドキュメント のみが表示されます。一時的なツリーは表示されません。
- XQuery 実行、結果ドキュメントが XML ソースからデータを取得することなく作成される場合も存在します。この場合、XML ファイルは、バックマッピング に含まれません。この結果は表示されません。
- 複数の XML ファイルがデータソースとして使用される場合、変換、または、実行 プロセス内での最初の発生が表示されます。
- バックマッピング は バックマップされていない 変換/実行 より先、より遅くより多くのメモリを消費します。サイズの大きなファイルと作業する際この点に注意してください。
- (分割されている、または、分割されていない場合の) 結果ドキュメント のコンテキスト メニューには「次のコンテキスト ノードに移動する」と「ソース 命令に移動する」)、コマンドが含まれており、XML と XSLT/XQuery ドキュメント 内の対応するノードに移動することができます。

バックマッピングツールバー

バックマッピングツールバー(下のスクリーンショット) には次のアイコンが含まれています:

- ブラウザービュー でマウスをポイントすると HTML を ハイライトする: 結果ドキュメント がブラウザービューで表示されると、結果ドキュメント 内のコンテンツを選択することにより バックマッピング を使用することができます。XML、または、XSLT/XQuery ドキュメント 内で選択しても バックマッピング を使用することはできません。ブラウザービュー内では、以下の 2 つの方法で バックマッピング のコンテンツを選択することができます: (i) ブラウザービュー内のコンテンツをクリックする。または、(ii) コンテンツにマウスをポイントする。この切り替えコマンドを使用して 2 つの選択メソッドを切り替えます。(例えば、ラジオボタン、、または、コンボボックスをクリックするなど) ブラウザービュー内のコンテンツのクリックが結果ドキュメント 内での変更を引き起こす場合、マウスのポイントは役に立ちます。

- *バックマッピング セッションの終了*: バックマッピング セッションを終了します。

バックマッピング セッションの終了

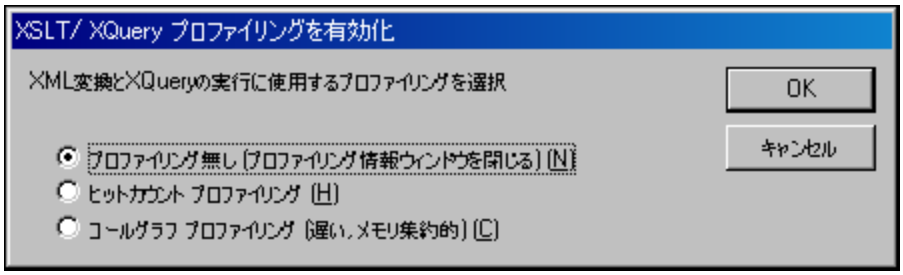
バックマッピング セッションの作動中、**バックマッピングセッションの終了** アイコンがバックマッピング ツール バーに表示されます (*上を参照*)。バックマッピングのセッションを終了するにはこのアイコンをクリックします。バックマッピングセッションは分析のためのみに使用してください。ドキュメントを編集する場合、編集の前にバックマッピングを終了することが奨励されます。

テキスト ハイライトの色

現在選択されている (バックマップされたコンテンツである) アクティブなバックマッピングの色、および、(選択されていない バックマップされたコンテンツである) アクティブではない バックマッピングは、テキスト フォント 設定のその他のカテゴリ (ツール | オプション | テキスト フォント 1470) で設定することができます

28.8.7 XSLT/XQuery プロファイリングの有効化

「XSLT/XQuery **プロファイリングの有効化**」コマンドにより XSLT/XQuery プロファイリングの有効化コマンドが表示されます。このダイアログにより、XSLT 変換や XQuery 実行時に実行されるインストラクションに必要な時間を分析するツールであるプロファイラー 492 が有効になります。



28.8.8 XSL の割り当て

「XSL/XQuery | XSL の割り当て...」コマンドにより、XSLT スタイルシートが XML ドキュメントへ割り当てられます。このコマンドをクリックすることで、目的の XSLT ファイルを指定するためのダイアログが表示されます。ファイルの選択は (参照 1132 ボタンをクリックして) グローバル リソースや URL、(**ウィンドウ**ボタンをクリックして) XMLSpy にて既に開かれているファイルからも行うことができます。

XML ドキュメントに xml-stylesheet 処理命令が挿入されます::

```
<?xml-stylesheet type="text/xsl" href="C:¥workarea¥recursion¥recursion.xslt"?>
```

メモ:...に対する相対パスにするチェックボックスを有効にすることで、ファイルへの パスを相対パスにすることができます。

28.8.9 XSL-FO の割り当て

「XSL/XQuery | XSL: FO の割り当て」コマンドにより FO ドキュメントへの変換を行うための XSLT スタイルシートが XML ドキュメントへ割り当てられます。このコマンドにより、XSL または XSLT ファイルを指定するためのダイアログが表示され、適切な処理命令が XML ドキュメントに挿入されます。

...に対する相対パスにするチェックボックスを有効にすることで、ファイルへのパスを相対パスにすることができます。ファイルの選択は(参照[1132] ボタンをクリックして) グローバルリソースや URL、(ウィンドウボタンをクリックして) XMLSpy にて既に開かれているファイルからも行うことができます。

メモ: XML ドキュメントには2つの XSLT ファイルを割り当てることができます。1つが標準的な XSLT 変換のためのもので、もう1つが FO への XSLT 変換で使用されます。

28.8.10 サンプル XML ファイルの割り当て

「XSL/XQuery | サンプル XML ファイルの割り当て」コマンドにより XML ファイルを XSLT ドキュメントへ割り当てます。このコマンドにより、XSLT ファイルにて XSL 変換がなされたときに使用される XML ファイルを指定した処理命令が挿入されます:

```
<?altova_samplexml C:¥workarea¥html2xml¥article.xml?>
```

メモ: 「...に対する相対パスにする」チェックボックスを有効にすることで、ファイルへのパスを相対パスにすることができます。ファイルの選択は(参照[1132] ボタンをクリックして) グローバルリソースや URL、(ウィンドウボタンをクリックして) XMLSpy にて既に開かれているファイルからも行うことができます。

28.8.11 XSL へ移動

「XSL/XQuery | XSL へ移動」コマンドにより、割り当てられている XSLT ドキュメントが開かれます。XML ドキュメントに、以下のような (XSLT 割り当てを行う) 処理命令が含まれている場合、

```
<?xml-stylesheet type="text/xsl" href="Company.xsl"?>
```

「XSL へ移動」コマンドにより XSLT ドキュメントが XMLSpy にて開かれます。

28.8.12 ソース命令に移動

バックマッピング[1285] の結果ドキュメント内でノードを選択する場場合、「ソース命令に移動」をクリックすると結果ドキュメント内の選択されたノードを生成した XSLT または XQuery ドキュメント内のソース命令にカーソルが移動されます。例えば、希望する場合命令を直接編集することが可能です。

バックマッピングの詳細に関してはメニューコマンド「XSL | バックマッピングの有効化」1285 を参照してください。

28.8.13 コンテキストノードに移動

バックマッピング1285 の結果ドキュメント内でノードを選択する場合、「**コンテキストノードに移動**」をクリックすると結果ドキュメント内で選択されたノードのコンテンツを生成したXMLドキュメント内のデータノードにカーソルが移動されます。例えば、希望する場合データノードを編集することが可能です。

バックマッピングの詳細に関してはメニューコマンド「XSL | バックマッピングの有効化」1285 を参照してください。

28.8.14 デバッガーの開始/実行

Alt+F11

「XSL/XQuery | **デバッガーの開始/実行**」コマンドによりXSLT/XQueryドキュメントの処理が開始または再開されます。ブレークポイントがセットされている場合、そこで処理が停止します。トレースポイントがセットされている場合、ステートメントの終了ノードに到着した時点で、それらステートメントの出力がトレースウィンドウに表示されます。デバッガーセッションが開始されていない場合、このボタンによりセッションが開始され、ファイルにある最初の処理ノードにて停止します。セッションが既に開始されている場合、XSLT/XQueryドキュメントの最後、または次のブレークポイントまで処理が行われます。

28.8.15 デバッガーの停止

「XSL/XQuery | **デバッガーの停止**」コマンドによりデバッガーが停止します。デバッガーセッションが動作している中でデバッガーを終了するのとは少し違い、デバッグセッションの最中にドキュメントの編集を行ったり、同じデバッグセッションにて別のファイルを使用する様な場合に便利な機能となります。デバッガーを終了させると、デバッガーを再度開始して、XSLT/XQueryドキュメントの最初からやり直す必要があります。

28.8.16 デバッガーの再起動

「XSL/XQuery | **デバッガーの再起動**」コマンドにより、出力ウィンドウの内容がクリアされ、現在選択されているファイルにてデバッグセッションが再度開始されます。

28.8.17 デバッグセッションの終了

「XSL/XQuery | デバッグセッションの終了」コマンドによりデバッグセッションが終了され、デバッグセッションが開始される以前にアクティブであった通常のXMLSpy ビューに切り替わります。XSLT/XQuery デバッガー設定 473 ダイアログでは、デバッグセッション中に開かれていた出力ドキュメントを開かれたままにするかどうかを設定することができます。

28.8.18 ステップイン

F11

「XSL/XQuery | ステップイン」コマンドにより、スタイルシートにある全てのノードならびにXPath 条件式の内部を処理します。このコマンドは停止されているデバッガーを再度開始するためにも使用することができます。

28.8.19 ステップアウト

Shift+F11

「XSL/XQuery | ステップアウト」コマンドにより、親ノードから見た次の兄弟ノードに移動するか親ノードから見た1つ上のレベルにある次のノードに移動します。

28.8.20 ステップオーバー

Ctrl+F11

「XSLT/XQuery | ステップオーバー」コマンドにより、現在のノードから同じレベルにある次のノードに移動するか、現在のノードより先上のレベルにある次のノードへ移動します。このコマンドは停止されているデバッガーを再度開始するためにも使用することができます。

28.8.21 現在実行中のノードを表示

「XSL/XQuery | **現在実行中のノードを表示**」コマンドにより、XSLT/XQuery ドキュメント内にある現在実行中のノードならびに XML ドキュメント内にある対応するコンテキストノードを表示/選択します。このコマンドは XSLT スタイルシートや XML ファイルのコードを表示/マーキングするために他のタブをクリックした後、それまで処理を行っていた場所に戻る際に便利な機能になります。

28.8.22 ブレークポイントの挿入/削除

F9

「XSL/XQuery | **ブレークポイントの挿入/削除**」コマンドにより、現在のカーソル位置にブレークポイントが挿入、または既にあるブレークポイントが削除されます。インラインブレークポイントは XSLT/XQuery や XML ドキュメントの両方で定義することができ、処理がどこで停止するかを決定することができます。赤色の点線がブレークポイントをセットしたノードの上に表示されます。終了ノードにブレークポイントをセットすることはできず、XSLT ドキュメントの属性に対するブレークポイントは無視されます。このコマンドは右クリックして表示されるコマンドからも利用することができます。

28.8.23 トレースポイントの挿入/削除

Shift+F9

「XSL/XQuery | **トレースポイントの挿入/削除**」コマンドにより、XSLT/XQuery ドキュメント内のカーソル位置にトレースポイントが挿入、または削除されます。デバッグ中にトレースポイントがセットされたステートメントの終了ノードにたどり着くと、そのステートメントの結果がトレースウィンドウに表示されます。青色の点線がトレースポイントをセットしたノードの上に表示されます。終了ノードにトレースポイントをセットすることはできません。このコマンドは右クリックして表示されるコンテキストメニューからも利用することができます。

28.8.24 ブレークポイントの有効化/無効化

Ctrl+F9

「XSL/XQuery | **ブレークポイントの有効化/無効化**」コマンドにより、既に定義されたブレークポイントを有効/無効化することができます。ブレークポイントが無効化されると、赤色のブレークポイントが灰色で表示されます。デバッガーは無効化されたブレークポイントでは停止しません。ノード名までカーソルを移動させ、**ブレークポイントの有効/無効化**コマンドをクリックして、ブレークポイントを有効/無効化することができます。このコマンドは右クリックにより表示されるコンテキストメニューからも利用できます。

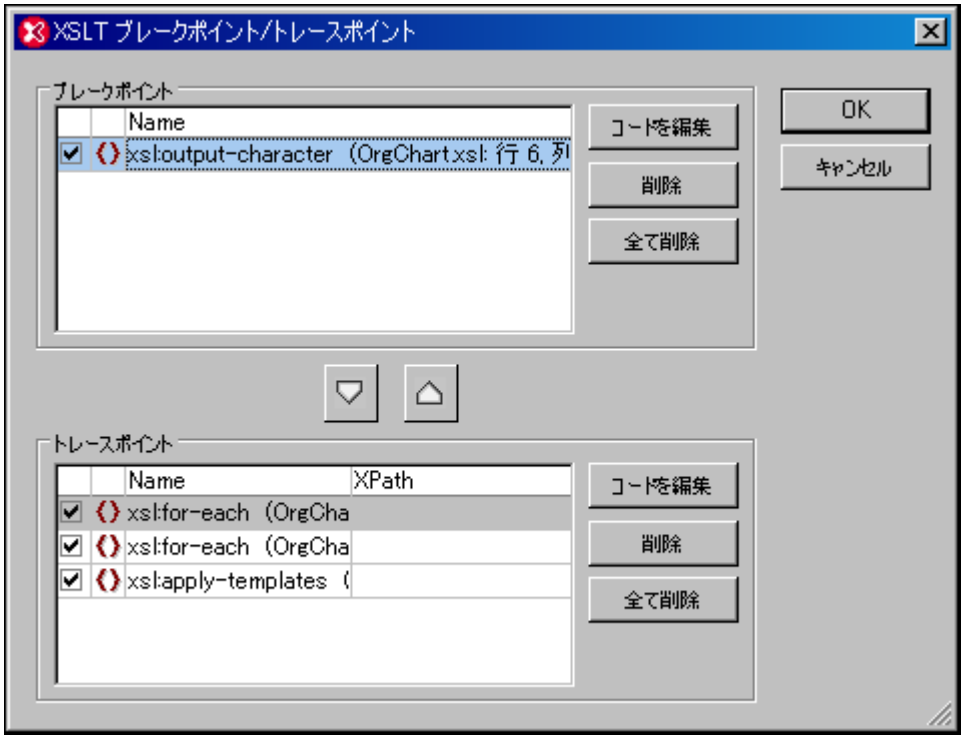
28.8.25 トレースポイントの有効化/無効化

Ctrl+Shift+F9

「XSL/XQuery | トレースポイントの有効化/無効化」コマンドにより、既に定義されたトレースポイントを有効/無効化することができます。トレースポイントが無効化されると、青色のトレースポイントが灰色で表示されます。無効化されたトレースポイントにおいて出力は行われません。ノード名までカーソルを移動させ、トレースポイントの有効/無効化コマンドをクリックして、トレースポイントを有効/無効化することができます。このコマンドは右クリックにより表示されるコンテキストメニューからも利用できます。

28.8.26 ブレークポイント/トレースポイント

「XSL/XQuery | ブレークポイント/トレースポイント」により、XSLT ブレークポイント/トレースポイントダイアログが表示され、現在のデバッグセッションにおいて全てのファイルで定義されているブレークポイントやトレースポイントが(無効になっているブレークポイントやトレースポイントも含めて)表示されます。



各ブレークポイントやトレースポイントの隣にあるチェックボックスは、それらが有効になっているかどうかを表します。「削除」ボタンをクリックすることで、ハイライトされているブレークポイントやトレースポイントを削除することができます。「全て削除」ボタンをクリックすることで、表示されている全てのブレークポイントまたはトレースポイントが削除されます。「コードを編集」ボタンをクリックすると、ファイル内にあるブレークポイント/トレースポイントへ直接ジャンプすることができます。

下矢印 ボタンを使用することで、ハイライトされたブレークポイントを「**トレースポイント**」ペインへ、上矢印 ボタンにより、ハイライトされたトレースポイントを「**ブレークポイント**」ペインへ移動することができます。
トレースポイントペインの**XPath** カラムでは、各トレースポイントに対して XPath をセットすることができます。

28.8.27 デバッグウィンドウ

「XSL/XQuery | **デバッグウィンドウ**」コマンドのサブメニューには、XSLT/XQuery デバッガーにある様々な情報ウィンドウが用意されています。リストにある情報ウィンドウを選択することで、XSLT/XQuery デバッガーインタフェース上にあるウィンドウの表示/非表示を選択することができます。このコマンドは、デバッグセッション中にのみ使用することができます。

28.8.28 デバッグ設定

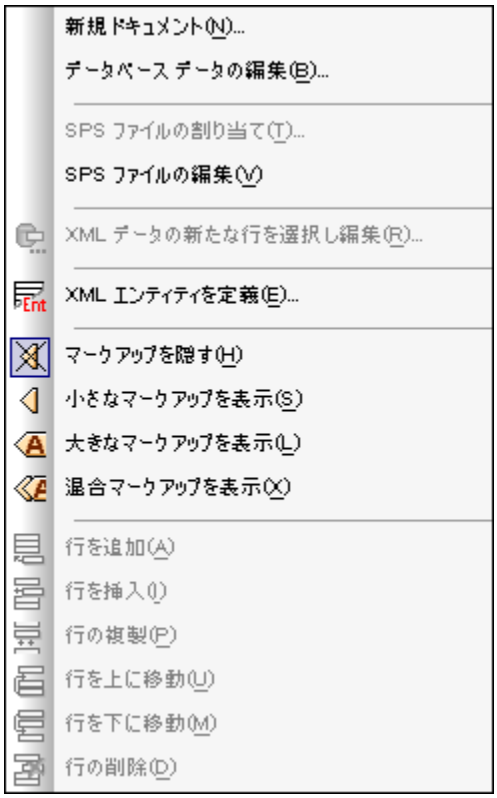
「XSL/XQuery | XSLT/XQuery **の設定**」コマンドにより、デバッグ設定ダイアログ[473] が開かれ、デバッガーのユーザーオプションを選択することができます。詳細はXSLT/XQuery デバッガー[467] セクションを参照下さい。

28.9 Authentic メニュー

Authentic View を使用することで、**Altova StyleVision** により作成された**StyleVision Power Stylesheet**（.sps ファイル）をベースにしたドキュメントの編集を行うことができます！スタイルシートにはXML ファイルを Authentic View にて視覚的に表示するための情報が収められています。表示するための情報に加えて、StyleVision Power Stylesheet によりXML ファイルへデータを書き込むこともできます。データはXSLT スタイルシートに用意されている機能により動的に処理され、その結果は Authentic View にて直ちに表示されます。

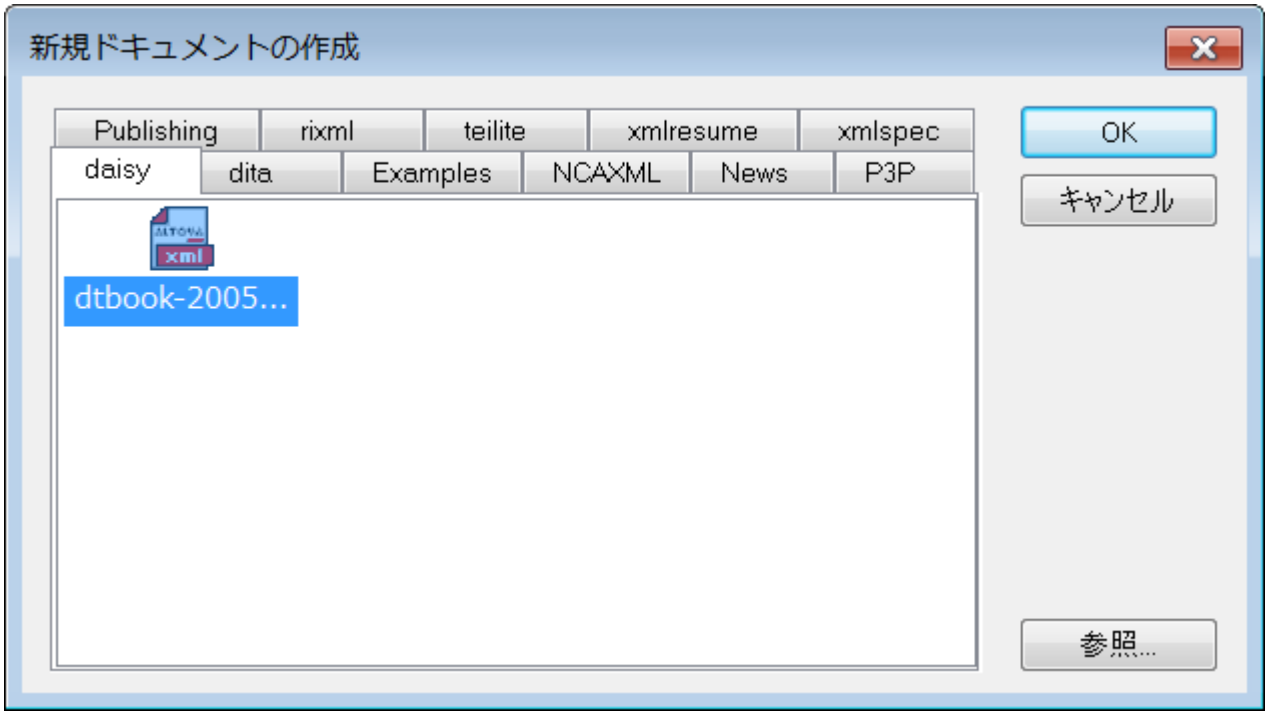
他にも StyleVision Power Stylesheet を使うことでデータベースにあるデータの編集を行うこともできます StyleVision Power Stylesheet にはデータベースへ接続するための情報が含まれており、データベースから得られたデータを Authentic View にて表示し、編集後のデータをデータベースへ書き込むことができます。

Authentic メニューコマンドにはXML ドキュメントを Authentic View にて編集するためのコマンドが収められています。のチュートリアルは、 Authentic View チュートリアル[528] セクションを参照ください。



28.9.1 新規ドキュメント

このコマンドにより Authentic View 内で新規のXML ドキュメントテンプレートを開く事ができます。XML ドキュメントテンプレートは、StyleVision Power Stylesheet (.sps ファイル)をベースにしています。新規のドキュメントの作成 ダイアログ内で StyleVision Power Stylesheet（SPS ファイル）を選択する事により開かれます（*下のスクリーンショット*）。 SPS を選択して、「OK」をクリックすると、SPS ファイルのために定義されたXML ドキュメントテンプレートが Authentic View 内で開かれます。



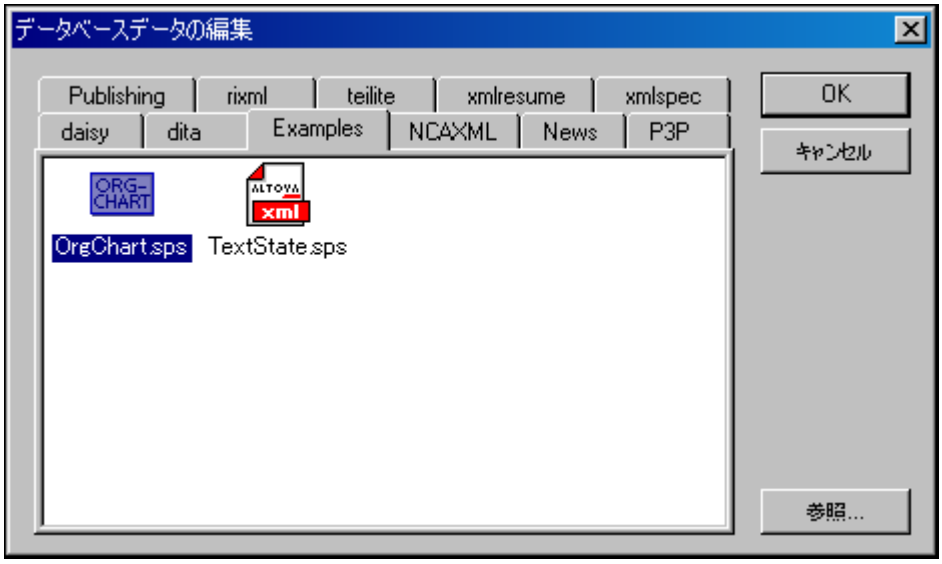
新規のドキュメントの作成 ダイアログは、一般的に使用されるDTD またはスキーマをベースにした、XMLドキュメントテンプレートの選択を提供します。または、テンプレート XML ファイルが割り当てられているカスタムメイドのSPS ファイルを参照することができます。SPS ファイルは、Altova StyleVision を使用して作成されます。アプリケーションにより、DTD またはXML スキーマをベースにしたXMLドキュメントテンプレートをデザインすることができます。StyleVision 内で必要な SPS をデザインすると、XML ファイルは、（ StyleVision 内で SPS へのテンプレート XML ファイルとして割り当てられます。XML ファイル内のデータは、XMLSpy のAuthentic View 内で開かれた新規のドキュメントテンプレートの開始データを与えます。

新規のXMLドキュメントテンプレートは、ですから、テンプレート XML ファイルとして選択されたSPS とXML ファイルのデータ内で定義されたドキュメントプレゼンテーションプロパティを持つようになります。Authentic View ユーザーは、XMLドキュメントテンプレートをグラフィカルなW YSIW YG インターフェイスで編集することができ、XMLドキュメントとして保存することができます。

28.9.2 データベースの編集

「**Authentic | データベースの編集...**」コマンドにより、データベースのビューをAuthentic View にて開くことができます。DB への接続方法やDB の表示方法、そしてAuthentic View にて許されている編集方法などは、StyleVision Power Stylesheet に記述されます。**データベースの編集**コマンドで開かれるのは、そのようなDB をベースにしたStyleVision Power Stylesheetです。このコマンドにより、DB への接続が確立され、DB のデータが（XML を介して）Authentic View にて表示されます

データベースの編集... コマンドにより、データベースデータの編集ダイアログが表示されます。



目的のSPS ファイルを選択してください。この操作によりDB への接続が行われ、DB が編集できる状態でAuthentic View に表示されます。Authentic View にて表示されているDB ビューのデザインはStyleVision Power Stylesheet に含まれています。

注意: **データベースの編集**コマンドを使ってDB をベースにしていないStyleVision Power Stylesheet や、StyleVision 2005 よりも前のStyleVision で作成されたDB ベースのStyleVision Power Stylesheet を開こうとした場合、エラーが表示されます。

メモ: StyleVision Power Stylesheet はAltova StyleVision により作成されます。

28.9.3 StyleVision スタイルシートの割り当て

StyleVision スタイルシートの割り当てコマンドにより、StyleVision Power Stylesheet が**XML ドキュメント**に割り当てられ、そのXML ドキュメントをAuthentic View にて閲覧ならびに編集することができるようになります。XML ファイルに割り当てられるStyleVision Power Stylesheet は、XML ファイルがベースとしているスキーマファイルをベースにしなければなりません。

StyleVision Power Stylesheet をXML ファイルへ割り当てるには:

1. XML ファイルがアクティブな状態で、「**Authentic** | StyleVision **スタイルシートの割り当て**...」コマンドを選択します。
2. この操作により、XML への割り当てを行う StyleVision Power Stylesheet ファイルを指定します。
3. 「**OK**」をクリックして、目的の SPS スタイルシートを XML ドキュメントに挿入します。「...**に対する相対パスにする**」チェックボックスを有効にすることで、ファイルへのパスを相対パスにすることができます。ファイルの選択は(参照 1132 ボタンをクリックして) グローバルリソースやURL、(**ウィンドウボタン**をクリックして) XMLSpy にて既に開かれているファイルからも行うことができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<?altova_sps HTML-Orgchart.sps?>
```

StyleVision Power Stylesheet がHTML-Orgchart.sps という名前で、XML ファイルと同じディレクトリ内に収められていることが分かります。

メモ: : 以前のバージョンの Altova 製品では、ターゲットが xmlspysps という名前とともに処理命令を使用しており、処理命令は <? xmlspysps HTML-Orgchart.sps?> というような形式になりました。最新の Altova 製品では、このような古いバージョンの処理命令も Authentic View の表示に使用することができます。

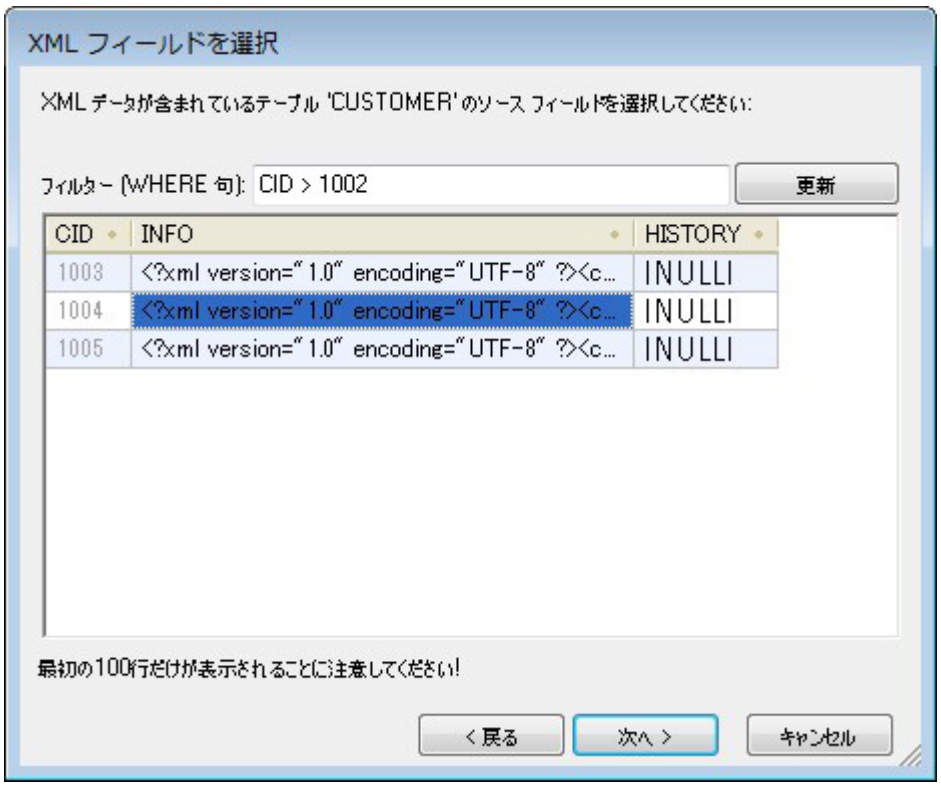
28.9.4 StyleVision スタイルシートの編集

Authentic | 編集 StyleVision スタイルシート コマンドは Authentic View のみで使用することができ、すなわち StyleVision Power Stylesheet が XML ドキュメントに割り当てられています。StyleVision を開始し StyleVision 内ですぐに StyleVision Power Stylesheet を編集することを許可します。

28.9.5 XML データの新たな行を選択し編集

「**XML データの新たな行を選択し編集**」コマンドにより、IBM DB2 といった XML DB にあるテーブルから新たに行を選択することができます。新たな行は Authentic View に表示され、編集を行った後に DB への書き込みを行うことができます。

XML DB が XML データソースとして使用される場合、XML データカラムのセルに収められている XML ドキュメントが Authentic View に表示されます。**XML データの新たな行を選択し編集**コマンドにより、XML カラム上の他のセル(または行)にある XML ドキュメントを選択することができます。このコマンドを選択することで、XML フィールドの選択ダイアログが表示され、XML カラムに含まれているテーブルが表示されます。



フィルタリングによりこのテーブルに表示される内容を制限することができます。フィルタリングに使用されるのは SQL の WHERE 句です(例えば、CID>1002 といった WHERE 句の中に入る条件になります)。**更新**ボタンをクリックして、ダイアログの表示内容を更新することができま

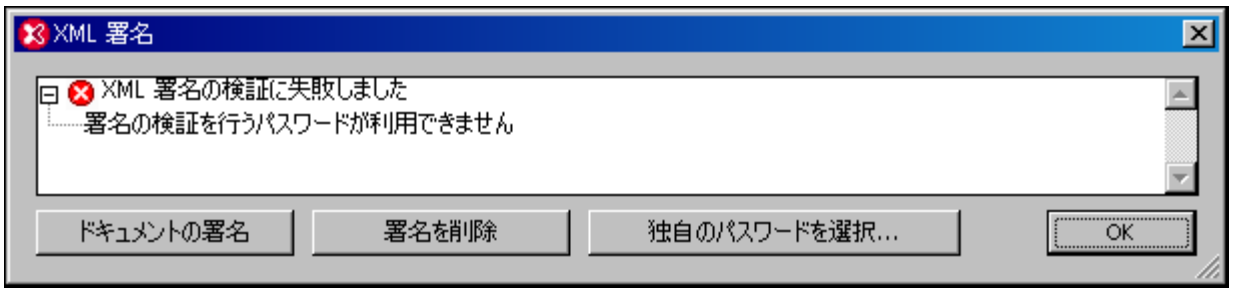
す。上のスクリーンショットではフィルタリングされた結果を見ることができます。目的のXMLドキュメントが表示されているセルを選択して**OK**をクリックします。選択されたセルのXMLドキュメントがAuthentic Viewにロードされます。

28.9.6 XML 署名

関連付けられているSPS にてXML 署名が有効になっている場合、Authentic ビューで「**XML 署名**」コマンドを使用することができるようになります。Authentic ツールバーにある「**XML 署名**」ツールバーアイコン からもXML 署名コマンドへアクセスすることができます。

検証と独自の証明書/パスワード

「**XML 署名**」コマンドをクリックすると、署名の検証処理が開始されます。ドキュメント内に署名が存在しない場合、その結果がXML 署名ダイアログに表示され、表示されているボタンからAuthentic View ユーザーはドキュメントの署名を行うことができるようになります(以下のスクリーンショットを参照)。



Authentic Viewにて独自の証明書/パスワードを選択することができるようにオプションが指定されている場合、「**独自の署名を選択**」または「**独自のパスワードを選択**」ボタンがダイアログにて表示されます。認証において証明書が使用されるかパスワードが使用されるかは、SPS デザイナーが署名の設定を行った際に指定されます。署名は証明書またはパスワードをベースにしたものになります。ダイアログにボタンが表示されている場合、そのボタンをクリックすることで、Authentic View が証明書を選択、またはパスワードを入力することができるようになります。Authentic View ユーザーの選択はメモリ上に保管されており、そのセッションでのみ有効になります。証明書の選択またはパスワードの入力を行った後にドキュメントやアプリケーションが閉じられると、証明書/パスワードの設定も、SPS に保存されたオリジナルの設定に戻されます。

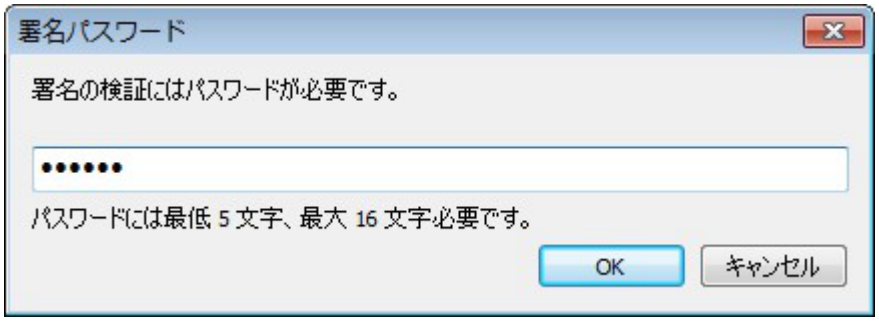
検証ならびに認証情報

単一のドキュメントに対して検証処理が行われた場合、2つのケースが考えられます。認証情報(署名またはSPS にて)が利用できる場合、検証処理が直接実行され、その結果が表示されます(以下のスクリーンショットを参照)。

(C) 2015-2021 Altova GmbH

認証情報は証明書の鍵情報または署名時に使用したパスワードとなります。XML ドキュメントが署名されたときに、証明書の鍵情報が署名の内に収められるか、またパスワードをベースにした署名の場合、パスワードがSPS の中に収められるかは、SPS デザイナーにより指定されます。どちらの場合でも認証を行うことができ、検証処理はAuthentic View ユーザーからの入力を必要とすることなく直接実行されます。

もう1つのケースは、認証情報が署名の中に存在しない(証明書の場合)、またはSPS ファイルの中に存在しない(パスワードの場合)というものです。この場合、認証情報(パスワードか証明書の場所)を入力するよう求められます(以下のスクリーンショットを参照)。



28.9.7 XML エンティティの定義

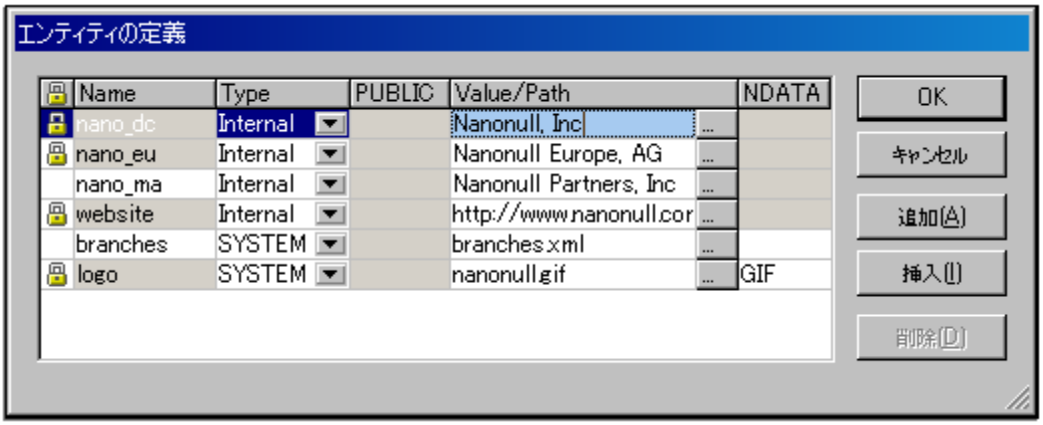
Authentic View 内で使用するエンティティを定義することができます。定義後、これらのエンティティはエンティティの入力ヘルパー内に表示され、コンテキストメニューの「**エンティティの挿入**」サブメニュー内に表示されます。エンティティの入力ヘルパー内のエンティティをダブルクリックすると、カーソル挿入ポイントにエンティティが挿入されます。

テキスト文字列、XML フラグメント、または、他の外部リソースをドキュメント内の複数の場所で使用する場合、エンティティは役にたちます。必要とされるデータのための短い名前であるエンティティを定義するには、エンティティの定義 ダイアログを使用します。エンティティを一度定義すると、そのエンティティをドキュメントの複数箇所で使用することができるようになります。この機能により、編集時間の短縮や、メンテナンスコストを大幅に軽減することができます。

ドキュメントで使用することのできるエンティティには、XML データ(テキスト文字列またはXML ドキュメントのフラグメント)の**解析対象実体**(解析対象エンティティ)と、バイナリファイル(通常はイメージや音楽などのマルチメディアオブジェクト)の様な非 XML データを扱う**解析対象外実体**(解析対象外エンティティ)と、大きく分けて2つの種類があります。各エンティティには名前と値が与えられています。解析されたエンティティの場合、エンティティはXML データのためのプレースホルダです。エンティティの値 はXML データ 自身、または、データを含む .xml ファイルをポイントするURI です。解析されていないエンティティの場合、エンティティの値は非-XML データ ファイルをポイントするURI です。

エンティティを定義する方法:

1. 「**Authentic | XML エンティティの定義**」をクリックします。エンティティの定義ダイアログが表示されます。



2. 名前フィールドにエンティティ名を入力します。この値がエンティティ入力ヘルパーに表示される名前となります。
3. エンティティの型を型フィールドのドロップダウンリストから入力します。3つある種類から選択を行います。テキストが使用される **Internal** エンティティがXMLドキュメント自身内に保管されています。**PUBLIC** または **SYSTEM** では、エンティティのリソースが、XMLファイルの外に配置され、それぞれPUBLIC 識別子とSYSTEM 識別子を使用することで、リソースが特定されます。システム識別子ではリソースの場所を表すURI が記述されます。公開識別子はプロセッサーがリソースを識別するためのロケーションに影響されない識別子です。公開とシステム識別子の両方を指定すると、公開識別子はシステム識別子、システム識別子を使用して解決します。
4. PUBLIC がType として選択されている場合、PUBLIC フィールド内にリソースの公開識別子を入力します。Type にてInternal またはSYSTEM を選択した場合、PUBLIC フィールドが無効になります。
5. **Value/Path** フィールドでは、以下のどれかを入力することができます:

   - エンティティの種類がInternal の場合、エンティティの値となるテキスト文字列を入力してください。エントリに引用符を入力しないでください。入力する引用符はテキスト文字列の一部として扱われます。
   - SYSTEM がエンティティの型である場合、参照ボタンを使用してリソースのURI 、または、ローカルネットワーク上のリソースを入力します。リソースが参照可能エンティティである場合、参照されるリソースはXML ファイル(つまり **.xml** 拡張子を持ったファイル)である必要があります。他にも、GIF ファイルのようなバイナリファイルを使用することもできます。
   - エンティティの型がPUBLIC の場合、このフィールドにシステム識別子を追加で入力する必要があります。

6. NDATA エントリはプロセッサーにこのエンティティを解析せず適切なプロセッサーに送信されるように命令します。そのためNDATA フィールドは解析対象外エンティティに対してのみ使用されます。

ダイアログ機能

対応するボタンをクリックすることでエンティティの追加、挿入、そして削除を行うことができます。カラムのヘッダーをクリックすることで、エンティティをアルファベット順に並び替え。1回クリックすることで昇順、2回クリックで降順の並び替えとなります。

ダイアログボックスの大きさならびにカラムの幅を変更することができます。エンティティが1度 XML ドキュメント内で使用されると、そのエンティティはロックされ、エンティティの編集ダイアログにて編集することができなくなります。ロックされたエンティティは、最初の列内でロックシンボルにより表示されます。エンティティをロックすることにより、エンティティに対してXMLドキュメントが有効であることを保証することができます(エンティティが参照されており、定義されていない場合は、ドキュメントは無効になります)。

エンティティの複製はフラグされます。

制約

- 他のエンティティ内に含まれているエンティティは、ダイアログ、Authentic View、XSLT 出力にて解決されず、そのようなエンティティのアンパサンドはエスケープされた形(つまり `&`)で表示されます。

- 外部エンティティは、そのエンティティがイメージである場合とENTITY または ENTITIES 型の属性値として入力された場合を除き Authentic View にて解決されません。このようなエンティティは、SPS から生成された XSLT にてドキュメントが処理されたときに解決されます。

28.9.8 マークアップの表示

Authentic XML ドキュメント内のマークアップをコントロールするオプションを使用するサブメニューがマークアップコマンドに搭載されています。これらのオプションは下で説明されています。

マークアップを隠すコマンドにより、Authentic View に表示されているマークアップシンボルを非表示にします。

小さなマークアップを表示コマンドにより、小さなマークアップシンボルを Authentic View にて表示します。

大きなマークアップを表示コマンドにより、大きなマークアップシンボルを Authentic View にて表示します。

混合マークアップを表示コマンドにより、異なるマークアップシンボルが Authentic View で表示されます。StyleVision Power Stylesheet では、デザイン時にドキュメントの各要素/属性に対して、大きなマークアップ、小さなマークアップ、またはマークアップ無しというオプションを指定することができます。混合マークアップのモードでは、カスタマイズされたマークアップが Authentic View に表示されます。

28.9.9 リッチエディット

リッチエディットコマンドへマウスを移動させることで、リッチエディットのマークアップに関するコマンドが含まれるサブメニューが表示されます（以下のスクリーンショットを参照）。このサブメニューに含まれるコマンドは、Authentic View にてリッチエディットコンポーネントの要素内にカーソルが配置された状態で選択可能になります。

リッチエディットメニューのテキストスタイルプロパティは、リッチエディットコマンドがクリックされた時に選択されたテキストへ適用されます。選択されたテキストにおけるフォント、フォントの太さ、装飾、大きさ、色、背景色、そして配置方法を Authentic View ユーザーは指定できます。

28.9.10 行の追加/挿入/複製/削除

行の追加コマンドにより、Authentic View の現在アクティブなテーブルに新たな行を追加します。

行の挿入コマンドにより、Authentic View の現在アクティブなテーブルに新たな行を挿入します。

行の複製コマンドにより、Authentic View の現在アクティブなテーブル行を複製します。

行の削除コマンドにより、Authentic View の現在アクティブなテーブル行を削除します。

(C) 2015-2021 Altova GmbH

28.9.11 上に/下に移動

行を上に移動コマンドにより、Authentic View における現在のテーブル行を、1行上に移動します。

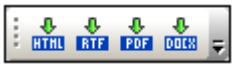
行を下に移動コマンドにより、Authentic View における現在のテーブル行を、1行下に移動します。

28.9.12 HTML、RTF、PDF、Word 2007+ ドキュメントの生成

以下にある4つのコマンドにより、PXF ファイル内部に収められた Authentic View XML ドキュメントから出力ドキュメントを生成することができます:

- HTML ドキュメントの生成
- RTF ドキュメントの生成
- PDF ドキュメントの生成
- Word 2007+ ドキュメントの生成

これらコマンドへは Portable XML Form (PXF) ツールバーからもアクセスすることができます(以下のスクリーンショットを参照)。

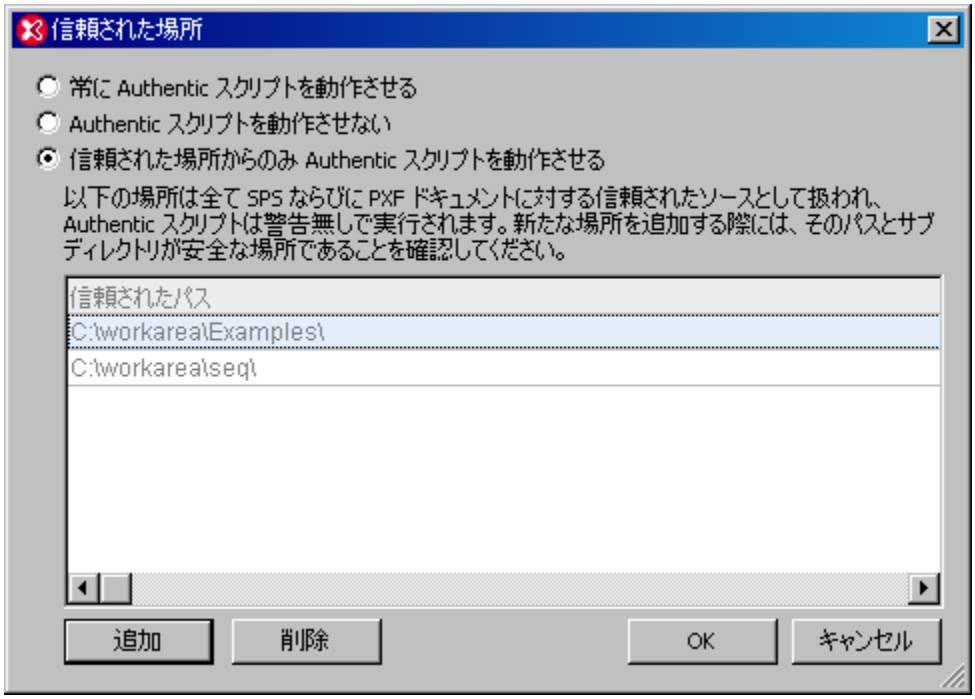


個々のコマンドまたはボタンをクリックすることで、HTML、RTF、PDF、または DocX 形式の出力がそれぞれ生成されます。

これらのボタンは PXF ファイルが Authentic View にて開かれた時に有効になります。PXF ファイル内にそれぞれの出力形式に対応した XSLT ファイルが含まれることで、個々のコマンドならびにボタンが有効になります。例えば、HTML ならびに RTF を出力する XSLT スタイルシートが PXF ファイル内部に含まれている場合、HTML ならびに RTF 出力を行うためのコマンドならびにツールバーボタンが有効になり、PDF と DocX (Word 2007+) 出力を行うコマンドは無効状態となります。

28.9.13 信頼された場所

信頼された場所コマンドを選択すると、(以下のスクリーンショットにある) 信頼された場所ダイアログが表示され、SPS 内にあるスクリプトのセキュリティー設定を指定することができます。スクリプトを含む SPS をベースにしている XML ファイルを Authentic ビューで開いた場合、このダイアログで指定された内容に従い、スクリプトの実行が許可されます(または許可されません)。



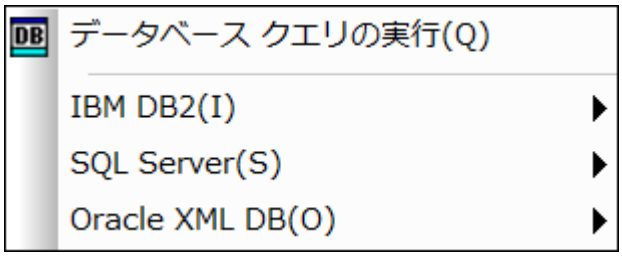
以下にある3つのオプションが指定可能です:

- Authentic ビューにてファイルが開かれた時に、Authentic スクリプトを常に動作させる。
- Authentic ビューにてファイルが開かれても、Authentic スクリプトを動作させない。
- 信頼された場所にある Authentic スクリプトだけを動作させる。信頼された場所(フォルダー)のリストが下部のペインに表示されます。**追加**ボタンを使用することで、フォルダーの参照とリストへの追加を行うことができます。リストからエントリーを削除するには、リストからエントリーを選択した後、**削除**ボタンをクリックしてください。

(C) 2015-2021 Altova GmbH

28.10 DB メニュー

DB メニューは(DB) オペレーションのためのメニューです。下のスクリーンショットに示されているように、下にリストされているメニューが含まれています。DB メニューのサブメニュー内のコマンドの説明は、このセクションのサブセクションを参照してください。

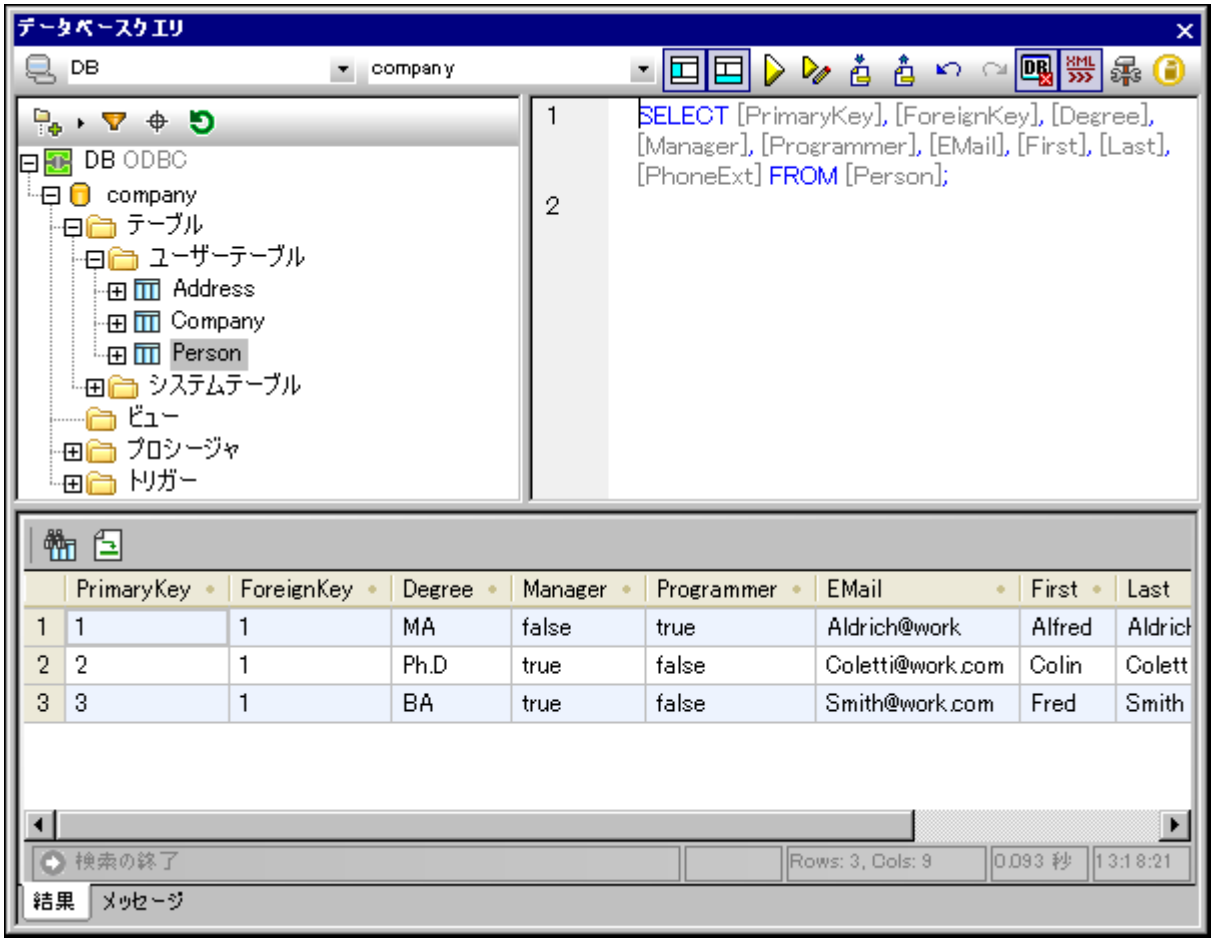


- クエリデータベース[1305] : 各種データベースの問い合わせを行うことができます。
- IBM DB2[1318] : DB2 に特化した機能がサポートされます。
- SQL サーバー[1324] : SQL Server データベースの管理を行うためのコマンドが収められます。.
- Oracle データベース[1326] : Oracle データベースを利用するためのコマンドが収められています。

このセクションで記述されている全てのコマンドを実行するには、データベースへの接続を行う必要があります。詳細に関しては[データベースへの接続](847)を参照してください。

28.10.1 データベースをクエリする

データベースクエリの実行コマンドでは、データベースクエリウィンドウが開かれます(以下のスクリーンショットを参照)。クエリウィンドウが開かれると、「DB | データベースクエリの実行」コマンドまたは、データベースクエリツールバーアイコン により、表示を切り替えることができます。



データベースクエリウィンドウの概要

データベースクエリウィンドウは、以下のパーツにより構成されています:

- 左上にある ブラウザーペイン[1310] により、接続情報ならびにデータベーステーブルが表示されます。
- 右上にある クエリペイン[1313] によりクエリが入力されます。
- タブ化されている 結果/メッセージペイン[1316] が下部に表示されます。結果ペインにはクエリの結果が結果グリッドと呼ばれるテーブルに表示されます。メッセージペインでは、警告、エラーを含むクエリの実行に関する情報が表示されます。

データベースクエリウィンドウの上部にはツールバーが用意されています。ここでは以下にある2つのツールバーアイコンに注目してください。他のツールバーアイコンに関する情報については、クエリペイン: 詳細と機能[1313] を参照ください。

ブラウザーペインの表示を切り替えます。

結果/メッセージペインの表示を切り替えます。

(C) 2015-2021 Altova GmbH

クエリデータベースメカニズムの概要

クエリデータベースは、以下のように動作します。詳細に関しては、このセクション以下にあるサブセクションを参照ください。

1. データベースクエリウィンドウにてデータベースへの接続が確立[1307]されます。サポートされているデータベースには、MS Access 2000 ならびに 2003、Microsoft SQL Server、Oracle、MySQL、Sybase、そして IBM DB2 があります。
2. 接続されたデータベースは、ブラウザーペイン[1310]にて表示され、最適な表示を行うための設定を施すこともできます。
3. 選択されているデータベースに対して適切な構文で記述された query[1315] が、Query ペイン[1315], に入力され、クエリが実行されます。
4. クエリの結果[1316] は、様々なフィルターにより閲覧、編集、そして DB への保存することができます。

28.10.1.1 データソース

データベースに対してクエリを実行するには、まず目的のデータベースへ接続する必要があります。このセクションでは、以下の方法について記述します:

- データベースへの接続を行い、
- 複数ある既存の接続から、目的のデータソースとルートオブジェクトを選択

データベースへの接続

セッションにて初めて(または既存のデータベース接続がない状態で) **データベースクエリの実行**コマンドをクリックすると、クイック接続ダイアログが表示され、データベースへの接続を行うことができます。作成した接続をそれ以降も使用する場合、データベースクエリウィンドウにあるクイック接続アイコン をクリックしてください。

クイック接続ダイアログからデータベースへ接続する方法については、データソースへの接続[847]セクションを参照ください。

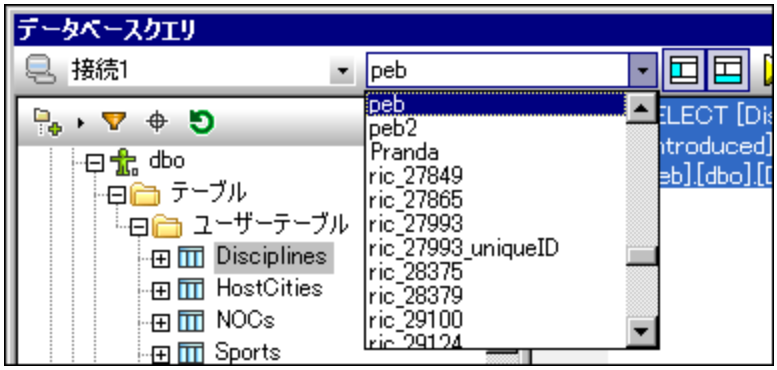
次のデータベースはサポートされています。各データベースの使用できるルート オブジェクトもリストされています。Altova は他の ODBC/ADO データベースへのサポートの向上のための努力をしていますが、以下にリストされたデータベースのみで接続とデータ処理のテストに成功しています。使用中の Altova アプリケーションが、64 ビット バージョンの場合、接続されているデータベース特定の 64 ビット データベース ドライバーにアクセスしてください。Altova アプリケーションは IBM iSeries データベースのロジカル ファイルをサポートし、ロジカル ファイルを表示します。

データベース	メモ
Firebird 2.5.4, 3.0	
IBM DB2 8.x, 9.1, 9.5, 9.7, 10.1, 10.5	
IBM Db2 for i 6.1, 7.1, 7.2, 7.3, 7.4	ロジカルファイルはサポートされ、ビューとして表示されています。
IBM Informix 11.70, 12.10, 14.10	
MariaDB 10.2, 10.3, 10.4, 10.5	
Microsoft Access 2003, 2007, 2010, 2013, 2016, 2019	ドキュメントの作成時（2019 年9 月初旬）Access 2019 のために使用可能な Microsoft Access Runtime が存在しませんでした。Microsoft Access 2016 Runtime がインストールされており、データベースが「大量」のデータ型を使用しない場合のみ Altova 製品から Access 2019 にデータベース接続することができます。

データベース	メモ
Microsoft Azure SQL Database	SQL Server 2016 codebase
Microsoft SQL Server 2005, 2008, 2012, 2014, 2016, 2017, 2019 Microsoft SQL Server on Linux	
MySQL 5.0, 5.1, 5.5, 5.6, 5.7, 8.0	
Oracle 9i, 10g, 11g, 12c, 18, 19	
PostgreSQL 8.0, 8.1, 8.2, 8.3, 9.0.10, 9.1.6, 9.2.1, 9.4, 9.5, 9.6, 10, 11, 12.1	PostgreSQL 接続は、ODBC または JDBC などのインターフェイス(ドライバ)によりネイティブとドライバーベースの接続でサポートされています。ネイティブな接続はドライバを必要としません。
Progress OpenEdge 11.6	
SQLite 3.33.0	SQLite 接続は、ネイティブとしてサポートされます。SQLite データベースファイルへの直接の接続です。 別途のドライバーは必要ありません。Authentic ビューでは、SQLite からのデータを編集することはできません。Authentic ビューで、SQLite からのデータを保存する場合は、既知の制限についてのメッセージが通知されます。
Sybase ASE 15, 16	
Teradata 16	

目的のデータソースを選択

既存の接続ならびにルートオブジェクトが、データベースクエリウィンドウのツールバーにある2つのコンボボックスにてそれぞれ表示されます(以下のスクリーンショットを参照)。



上のスクリーンショットでは、StyleVision DB という名前のデータベースが選択されています。このデータベースにて利用することのできるルートオブジェクトのうち、ALTOVA_USER が選択されています。データベースならびにルートオブジェクトはブラウザーペインにて表示されます。

28.10.1.2 ブラウザーペイン：DB オブジェクトの閲覧

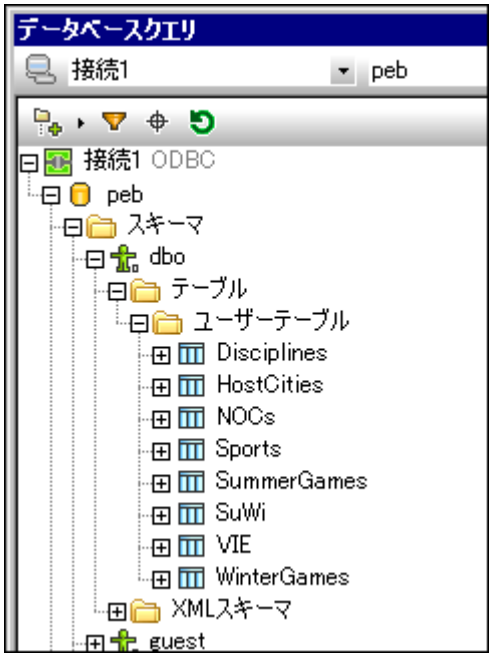
ブラウザーペインでは、選択されたデータベースにあるオブジェクトの概要が表示されます。この概要には、どのカラムが主キーや外部キーかといった、データベース制約情報も表示されます。IBM DB2 バージョン 9 データベースでは、登録された XML スキーマも別のフォルダーにてブラウザービューに表示されます。

このセクションでは以下のことがらに関する記述を行います:

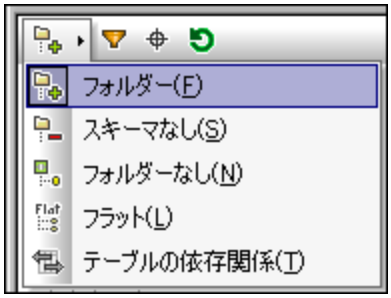
- ブラウザーペインにて利用可能な レイアウト 1310
- データベースオブジェクトの フィルタリングの方法 1311
- データベースオブジェクトの 検索の方法 1312

ブラウザーペインレイアウト

デフォルトのフォルダーレイアウトでは、データベースオブジェクトが階層構造で表示されます。選択されたオブジェクトにより、右クリックにより表示されるコンテキストメニューのオプションが変化します。



ブラウザーのレイアウトを選択するには、ブラウザーペインのツールバーにあるレイアウトアイコンをクリックし、ドロップダウンリストからレイアウトを選択します（以下のスクリーンショットを参照）。選択されたレイアウトによりアイコンが変化することにご注意してください。



利用可能なレイアウトは以下のとおりです:

- *フォルダー* : データベースオブジェクトを、オブジェクトの種類ごとにフォルダー形式の階層構造で表示します。デフォルトの設定です。
- *スキーマなし* : フォルダーレイアウトに似ていますが、データベーススキーマフォルダーは表示されません。そのため、テーブルはデータベーススキーマによりカテゴライズされません。
- *フォルダーなし* : フォルダーを使用すること無く、データベースオブジェクトを階層構造で表示します。
- *フラット* : データベースオブジェクトを、型ごとに最初の階層構造レベルにおいて割り振ります。例えば、個々のカラムが対応するテーブルに含められるのではなく、全てのカラムが特定のカラムフォルダーに収められて表示されます。
- *テーブルの依存関係* : 他のテーブルとのリレーションシップに従い、テーブルをカテゴライズします。外部キー付きのテーブル、外部キーにより参照されているテーブル、そして他のテーブルに対してリレーションシップを持たないテーブルがカテゴリーとして表示されます。

テーブルをユーザーテーブルならびにシステムテーブルで並べ替えるには、スキーマなし、またはフラットレイアウトへの切り替えを行い、テーブルフォルダーを右クリックして、**ユーザーとシステムテーブルの並べ替え**を選択します。ユーザーテーブルならびにシステムテーブル内のテーブルがアルファベット順に並べ替えられます。

データベースオブジェクトのフィルタリング

ブラウザーペインでは、(フォルダーなしと、テーブルの依存関係レイアウトを除き) スキーマ、テーブル、そしてビューを、名前や名前の一部によりフィルタリングすることができます。オブジェクトのフィルタリングは文字の入力により行われます。フィルタリングにおいて、大文字と小文字は区別されません。

以下の操作により、ブラウザーペインにてオブジェクトのフィルタリングを行うことができます:

1. ブラウザーペインのツールバーにあるフォルダーコンテンツのフィルターアイコンをクリックします。フィルターアイコンが、テーブルならびにビューフォルダーの隣に表示されます(以下のスクリーンショットを参照)。

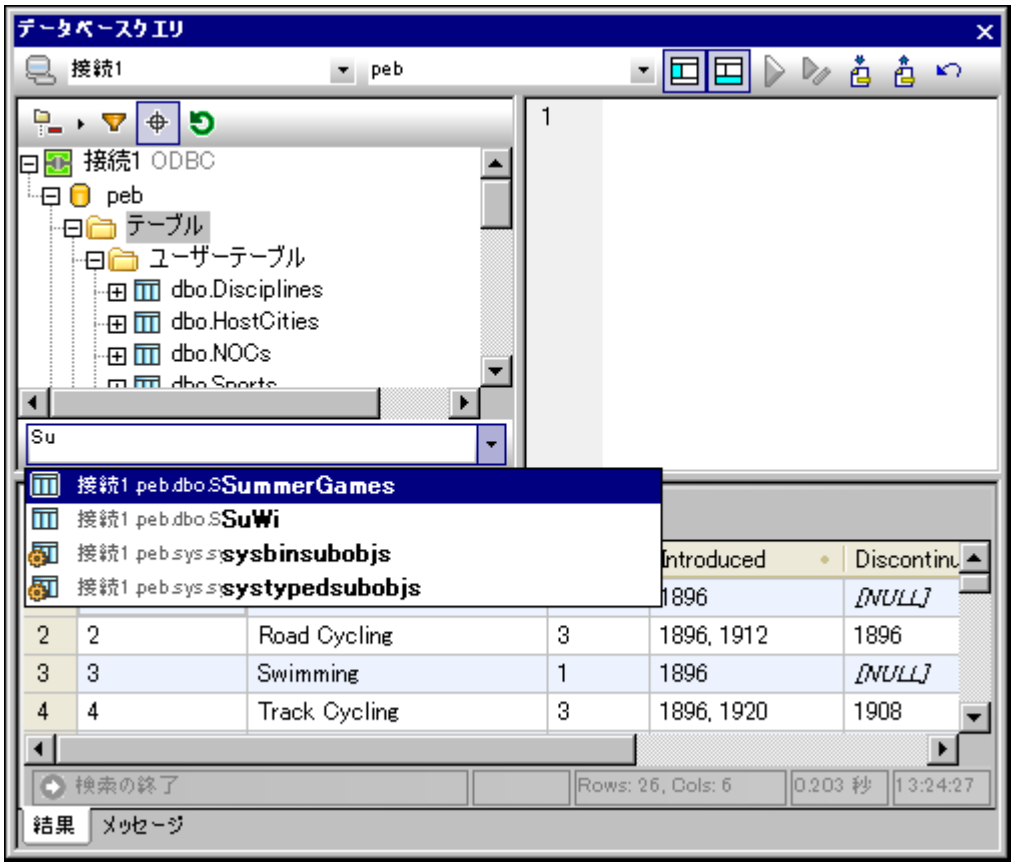
2. 新たに表示される入力フィールドにて、フィルタリングを行う文字列を入力します(以下のスクリーンショットでは、テーブルフォルダーに対して Su というフィルタリングが行われています)。入力しながらも、フィルタリングは行われます。

3. 新たに表示される入力フィールドにて、フィルタリングを行う文字列を入力します(以下のスクリーンショットでは、テーブルフォルダーに対して Su というフィルタリングが行われています)。入力しながらも、フィルタリングは行われます。

データベースオブジェクトの検索

以下にあるように、ブラウザーペインのオブジェクトロケーターを使用するで、特定の名前を持つデータベースアイテムを検索することができます。

1. ブラウザーペインのツールバーにて、オブジェクトロケーターアイコンをクリックします。ブラウザーペインの下部にドロップダウンリストが表示されます。
2. リストの入力フィールドに検索文字列を入力します。以下のスクリーンショットでは、例として Su という文字列が入力されています。ドロップダウンリストの矢印をクリックすることで、その検索文字列を含む全てのオブジェクトが表示されます。



3. リスト内のオブジェクトをクリックして、ブラウザーペインにて確認します。

(C) 2015-2021 Altova GmbH

28.10.1.3 クエリペイン: 説明と機能

クエリペインは、選択されたデータベースに対してクエリを実行するためのインテリジェントな SQL エディターです。クエリを入力した後は、データベースクエリウィンドウの実行コマンドをクリックすることでクエリが実行され、その結果と実行メッセージが結果/メッセージペイン[1316]にて表示されます。クエリの操作方法については、クエリペイン: クエリの使用方法[1315]を参照ください。このセクションでは、以下に示されるクエリペインの主要な機能を記述します。

- データベースクエリツールバーにある SQL エディターアイコン
- SQL エディターオプション
- SQL ステートメントの自動補完
- SQL スクリプトの領域定義
- SQL スクリプト内へのコメント挿入
- ブックマークの使用

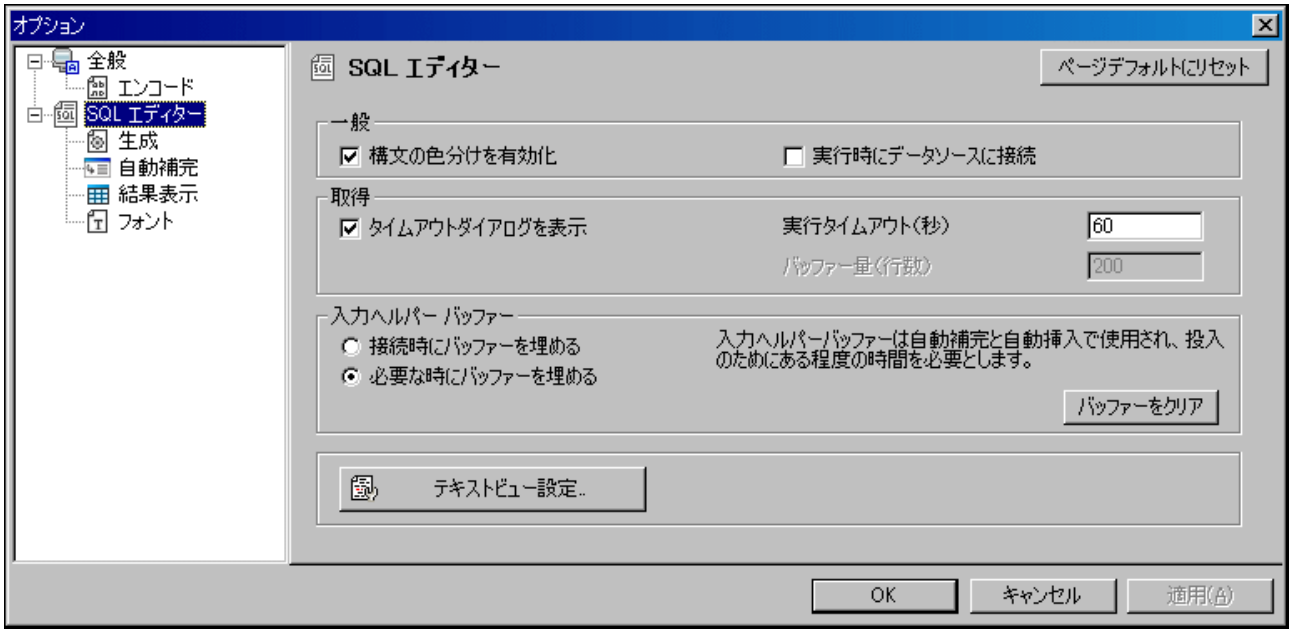
データベースクエリツールバーの SQL エディターアイコン

データベースクエリウィンドウのツールバーでは以下のアイコンが用意されており、SQL エディターを使用する際に利用することができます:

	実行	現在選択されている SQL ステートメントを実行します。スクリプトに複数のステートメントが含まれており、そのうちどれも選択されていない場合、全てのステートメントが実行されます。
	データ編集の実行	実行コマンドと同じですが、結果タブに表示される結果を編集することができます。
	SQL ファイルのインポート	SQL ファイルを SQL エディターにて開きます。
	SQL ファイルのエクスポート	SQL クエリを SQL ファイルに保存します。
	元に戻す	SQL エディターでなされた変更を回数に制限なく元に戻します。
	やり直し	SQL エディターでなされた「元に戻す」コマンドからやり直しを行います。
	XML を開く際に DB クエリを隠す	XML ドキュメントが開かれたときに、DB クエリウィンドウを隠すかどうか選択します。
	XML 保存時に自動コミット	編集された XML ドキュメントを XMLSpy にて保存する際に、このボタンが有効になっていれば、DB へのコミットが自動的に行われます。そうでない場合、結果ペインから明示的にコミットを行う必要があります。
	オプション	SQL エディターのオプションダイアログを開きます。
	SQL スクリプトを DatabaseSpy で開く	Altova の DatabaseSpy にて SQL スクリプトを開きます。

オプション

データベースクエリツールバーのオプションアイコンをクリックすることで、オプションダイアログが表示されます(以下のスクリーンショットを参照)。左側にあるペインから設定のページを選択し、そのページのオプションを指定することができます。ページデフォルトにリセットボタンをクリックすることで、そのページのオプションを、デフォルトの設定にリセットすることができます。

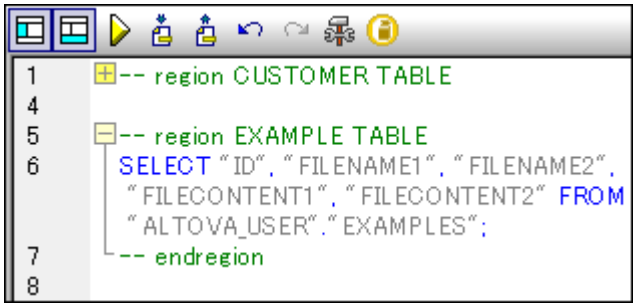


重要な設定を以下に記します:

- 全般 / エンコード : 新たな SQL ファイルや、エンコーディングを特定できない既存 SQL ファイルのエンコーディングならびにバイトオーダーマーク(BOM) をセットするオプションです。既存 SQL ファイルのエンコーディングを特定できた場合、エンコーディングを変更すること無く、開かれたファイルは保存されます。
- SQL エディター : 構文による色付けや実行時のデータソース接続などを有効にすることができます。クエリの実行にはタイムアウトを設定することができ、指定された値を超えて時間が経過すると、タイムアウトの変更を促すダイアログが表示されます。入力ヘルパーのバッファーは、データソースへの接続時、またはそれが最初に求められた際に埋められるよう指定することができます。テキストビュー設定ボタンにより XMLSpy のテキストビューオプションウィンドウが開きます。.
- SQL エディター / SQL 生成 : オブジェクトをブラウザーペインからクエリペインへドラッグすると、アプリケーションにより SQL ステートメントが生成されます。SQL ステートメント生成に関するオプションは、SQL 生成タブにて指定することができます。データベースペインからデータベースの種類を選択することで、各種データベースに対して個別のオプションを指定することができます。全てのデータベースに適用チェックボックスにチェックを入れることで、現在選択されているオプションが、全種類のデータベースに対して適用されます。オプションには、セミコロンをステートメントに追加したり、識別子をエスケープ文字により囲むなどがあります。エディターによりセミコロンを追加 チェックボックスが有効になっている場合、SQL ステートメントが SQL エディターにて生成された際に、セミコロンが追加されます。JDBC 接続により Oracle データベース、IBM iSeries、そして DB2 データベースへ接続し、データの編集を行うには、このチェックボックスのチェックを外す必要があります。
- SQL エディター / 結果ビュー : 結果タブのオプションを指定します。
- SQL エディター / フォント : テキストエディターならびに結果ビューのフォントを指定します。

SQL スクリプトの領域を定義

領域とは、SQL スクリプトにて選択され、1つのまとまりとして宣言された箇所のことです。領域により、縮退や展開、スクリプトの一部だけ非表示にすることが可能になります。領域を他の領域の中にネストすることもできます。領域の開始位置と終了位置は、-- region ならびに -- endregion コメントにより、それぞれ定義されます。オプションとして、-- region の後ろに文字列を入力することで、領域に名前を付けることができます(以下のスクリーンショットを参照)。



領域を挿入するには、領域に含めるステートメントを選択し、右クリックにて表示されるコンテキストメニューから、**領域の挿入**を選択します。展開/縮退可能な領域が作成されます。必要な場合は、領域に名前を追加してください。上のスクリーンショットで表示されている行番号にも注目してください。領域を削除するには、-- region ならびに-- endregion を削除してください。

SQL スクリプトにコメントを挿入

SQL スクリプトに表示されているテキストはコメントアウトすることができます。これらコメントはスクリプトの実行時にスキップされます。

- ブロックのコメントアウトを行うには、ブロックを選択して、右クリックで表示されるコンテキストメニューから、**ブロックコメントを挿入/削除**を選択します。ブロックコメントを削除するには、コメントを選択し、右クリックから**ブロックコメントを挿入/削除**を選択します。
- 行や、行の一部をコメントアウトするには、行コメントを挿入する場所にカーソルを移動し、右クリックから**行コメントを挿入/削除**を選択します。行コメントを削除するには、コメントを選択し、右クリックから**行コメントを挿入/削除**を選択します。

ブックマークの使用

特定の行にブックマークを挿入し、ドキュメント内のブックマークによりナビゲーションを行うことができます。ブックマークを挿入するには、ブックマークする行にカーソルを移動させ、右クリックにより表示されるコンテキストメニューから、**ブックマークの挿入/削除**を選択します。次や前のブックマークに移動するには、右クリックから**次のブックマークへ移動**または**前のブックマークへ移動**を選択します。ブックマークを削除するには、削除するブックマークの行にカーソルを移動させ、右クリックから**ブックマークの挿入/削除**を選択します。全てのブックマークを削除するには、右クリックから**全てのブックマークを削除**を選択します。

28.10.1.4 クエリペイン：クエリの実行

データベースへの接続を行うと、SQL エディターにて SQL スクリプトを入力、実行することができます。このセクションでは以下のことがらについて記述します:

- SQL エディターにて SQL スクリプトを入力する方法
- データベースクエリウィンドウにて SQL スクリプトを実行する方法

このセクションでは以下のアイコンが使用されます:

	クエリの実行	現在選択されている SQL ステートメントを実行します。スクリプトに複数のステートメントが含まれており、そのうちどれも選択されていない場合、それら全てが実行されます。
	データ編集の実行	実行コマンドと同じですが、(結果タブに表示される) 結果を編集することができます。

SQL ファイルのインポート SQL ファイルを SQL エディターにて開きます。

SQL エディターにてSQL ステートメントならびにスクリプトを作成

以下のGUI 操作からSQL ステートメントやスクリプトを作成することができます:

- ドラッグアンドドロップ : ブラウザーペインからSQL エディターへオブジェクトをドラッグします。そのオブジェクトをデータベースから問い合わせるためのSQL ステートメントが生成されます。
- コンテキストメニュー : ブラウザーペインにあるオブジェクトをクリックし、「新規 SQL エディターの表示 | SELECT」を選択します。
- マニュアル入力 : SQL ステートメントをSQL エディターに直接入力します。自動補完機能を使って編集を行うことができます。
- SQL スクリプトのインポート : データベースクエリウィンドウのツールバーにあるSQL ファイルのインポートアイコンをクリックします。

SQL ステートメントの実行

SQL エディターのSQL スクリプトに2つ以上のSQL ステートメントが含まれている場合、実行するステートメントを選択して、データベースクエリウィンドウのツールバーにある実行アイコンまたはデータ編集の実行アイコンをクリックします。SQL スクリプトが選択されていない場合は、スクリプト内にある全てのステートメントが実行されます。データベースのデータが取得され、結果タブ[1316]、のグリッドに表示されます。データ編集の実行が選択された場合、結果タブのグリッドに表示されたデータを編集することができます[1316]。実行に関するメッセージは、メッセージタブ[1316]に表示されます。

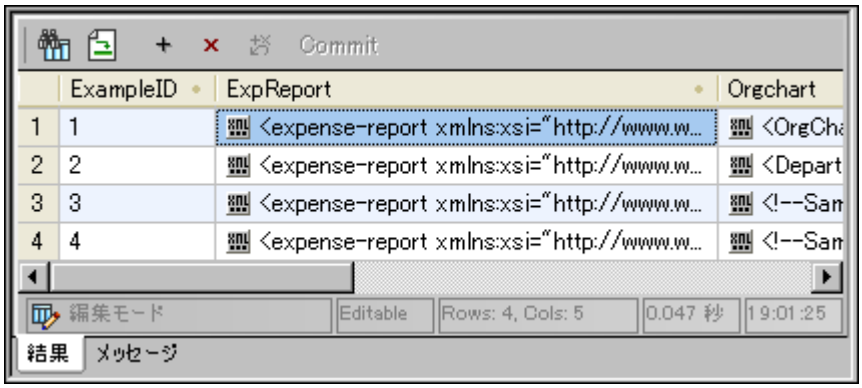
28.10.1.5 結果とメッセージ

結果/メッセージペインには2つのタブが含まれています:

- 結果タブ[1316] にはクエリにより取得されたデータが表示されます。
- メッセージタブ[1318] にはクエリの実行に関するメッセージが表示されます。

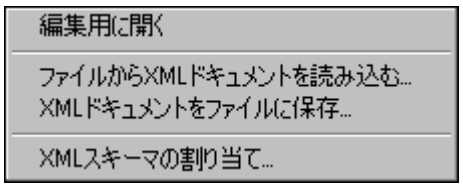
結果タブ

クエリにより取得されたデータは、グリッド形式で結果タブに表示されます(*以下のスクリーンショットを参照*)。**クエリの実行** コマンドによりクエリの結果が得られた場合、結果タブ内のXML ドキュメントはXML アイコン により表示されます(*下のスクリーンショット*)。 **データ編集の実行** コマンドが使用された場合 XML ドキュメントはXML アイコンにより表示されます。



次の操作は、コンテキスト結果タブ内の適切な場所を右クリックするとポップアップする、コンテキストメニューを使用して、結果タブで実行することができます。

- *列の並べ替え*: 記録が並べ替えられる列を右クリックし、[**並べ替え | 昇順/降順/既定値に復元**] を選択します。。
- *クリップボードにコピー*: この手順には2 つのステップがあります: (i) データ範囲の選択、(ii) 選択範囲のコピー。データは複数の方法で選択することができます。: (i) 列のヘッダーまたは行番号をクリックしてれつまたは行を選択、(ii) 個別のセルを選択 (**Shift** キーおよびまたは **Ctrl** キーを使用して複数のセルを選択します); (iii) セルを右クリックして、[**選択 | 行/列/すべて**] を選択します。。選択が決定されると、右クリックして [**選択されたセルのコピー**] を選択します。この操作により、選択対象がクリップボードにコピーされ、他のアプリケーションの処理に渡すことができるようになります。
- *新規の行の追加*:編集を行えるようクエリが実行された場合、結果ペインを右クリックすることで、**行の追加** コマンドへアクセスすることができます。
- *行の削除*:新しい行の追加 :編集を行えるようクエリが実行された場合、結果ペインを右クリックすることで、**行の削除**コマンドへアクセスすることができます。
- *レコードの編集* :編集を行えるようクエリが実行された場合、各フィールドを編集することができます。変更をコミットするには、結果ペインのツールバーにある **コミット** ボタンをクリックしてください。
- *XML レコードの編集*: この機能はIBM DB2, SQLServer、PostgreSQL (8.3 またはは以降)、および Oracle (9 またはは以降) データベースと主キーを持つDB テーブルのためにサポートされています。データ編集の実行コマンドにより編集可能な XML フィールドが取得された場合、結果グリッドにて編集可能 XML アイコン をクリックすることで、XML 編集メニューが表示されます(以下のスクリーンショットを参照)。ツリーから XML フィールドを右クリックして、**データ編集**を選択することでも XML フィールドを開くことができます。



編集用に開くコマンドにより、XML ドキュメントが XMLSpy ウィンドウで開かれ、編集可能アイコン内のドットが赤色に変化 します。ドキュメントが開かれたときに、データベースクエリツールバーにある XML の変更を自動コミットアイコン が有効になっている場合、このドキュメントが保存されるとともに、XML に対して行われた変更が自動的にデータベースへコミットされます。アイコンが有効になっていない場合、XML に対して保存された変更を、結果ペインにあるコミットボタンによりコミットする必要があります。XML ドキュメントとデータベースクエリウィンドウを切り替えるには、「**DB | データベースクエリの実行**」コマンドをクリックする必要があります。**ファイルから XML ファイルを読み込む** コマンドにより、外部の XML ドキュメントをデータベースの選択されたフィールドへロードすることができます。**XML ドキュメントをファイルに保存** コマンドでは、選択されたデータベースフィールドの XML ドキュメントを、ユーザーが選択する場所へ保存することができます。XML スキーマの割り当てコマンドを選択すると、XML スキーマの選択 ダイアログ[1322] が表示され、XML ドキュメントへ割り当てる XML スキーマを選択することができます。この割り当て情報はデータベースへ保存されます。XML スキーマの割り当てに関する詳細は、「 IBM DB2 | XML スキーマの割り当て[1322] のセクションを参照ください。
- *NULL のセット*、*デフォルトのセット*、*このセルの変更を取り消す* :編集を行えるようクエリが実行された場合、セルを右クリックすることで、そのセルに対して NULL 値や、(デフォルト値が定義されている場合) セルのデフォルト値をセットすることができます。セルに対して行われた変更は、**このセルの変更を取り消す**コマンドにより、表示されている値をDB に収められている値に戻すことができます。

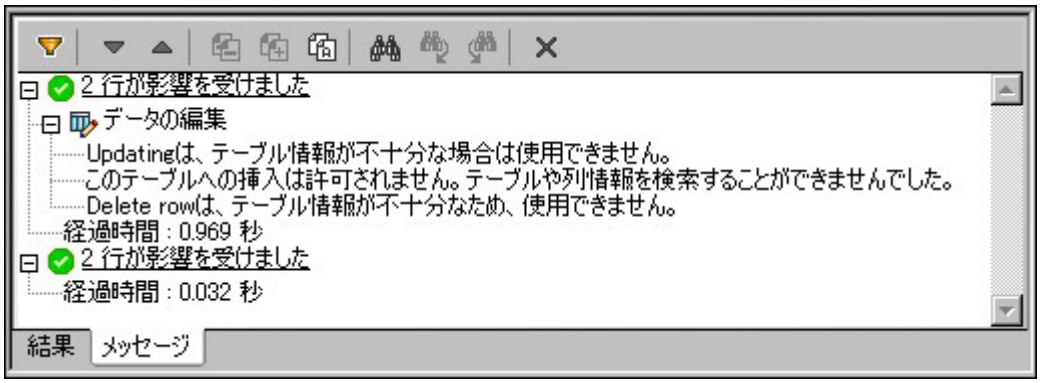
フォーミュラタブには以下のツールバーアイコンがあります:

	ステートメントへ移動	表示されている結果を取得するためのSQL エディターにてステートメントがハイライトされます。
	検索	結果ペインにあるテキストを検索します。XML ドキュメントのコンテンツも検索されます。
	新しい行の追加	新たな行を結果グリッドへ追加します。
	行の削除	現在選択されている行を結果グリッドから削除します。

	結果グリッドでの変更を全て元に戻す	結果グリッドでなされた全ての変更を元に戻します。
Commit	コミット	結果グリッドでなされた全ての変更をデータベースへコミットします。

メッセージタブ

メッセージタブにはそれまでに実行されたSQL ステートメントの情報がエラーや警告メッセージとともに表示されます。



メッセージタブのツールバーにはビューのカスタマイズやナビゲーション、クリップボードへのメッセージコピーを行うためのアイコンが含まれています。[フィルター] アイコンにより、特定の種類のメッセージの表示の切り替えをオンまたはオフします。「次」ならびに「前」アイコンにより、選択箇所を移動することができます。表示されているメッセージは、子コンポーネントを含めて(または含めないで)クリップボードにコピーすることができ、その後ドキュメントへの貼付けを行うことができます。[検索] 機能により、検索用語を指定し、この用語をリスト内上下に検索することができます。更に、クリアアイコンを使用すると、メッセージペインに表示されているコンテンツをクリアすることができます。

メモ ツールバーアイコンコマンドは、コンテキストメニューコマンドとしても使用することができます。

28.10.2 IBM DB2

IBM DB2 メニューアイテムには、(i) IBM DB2 データベースにおけるスキーマの登録や登録取り消し(XML スキーマの管理[1318])ならびに(ii) XML ファイル検証で使用されるスキーマの割り当て(XML スキーマの割り当て[1322])を行うためのコマンドが含まれています。

これらのコマンドを実行するには、IBM DB2 データベースへの接続を行う必要があります。データベースへの接続方法については、IBM DB2 (ODBC) への接続[883] セクションを参照ください。このセクションでは、IBM DB2 データベース内にてスキーマを管理する方法と、XML スキーマをDB XML ファイルに割り当てる方法について記述します。

メモ: データベースクエリウィンドウ[1305] の結果グリッドには、IBM DB2データベース内にあるXML ファイルを処理するための重要な機能が収められています。この機能を使うことで、XML ファイルを開き、DB XML ファイルへロード、DB XML ファイルを外部的に保存したり、XML スキーマをDB XML ファイルへ割り当てるといった操作を行うことができます。

28.10.2.1 XML スキーマの管理

XML スキーマの管理機能により、個々のデータベーススキーマから、IBM DB2 データベースへスキーマを追加したり、削除することができます。以下の操作によりスキーマの管理を行うことができます:

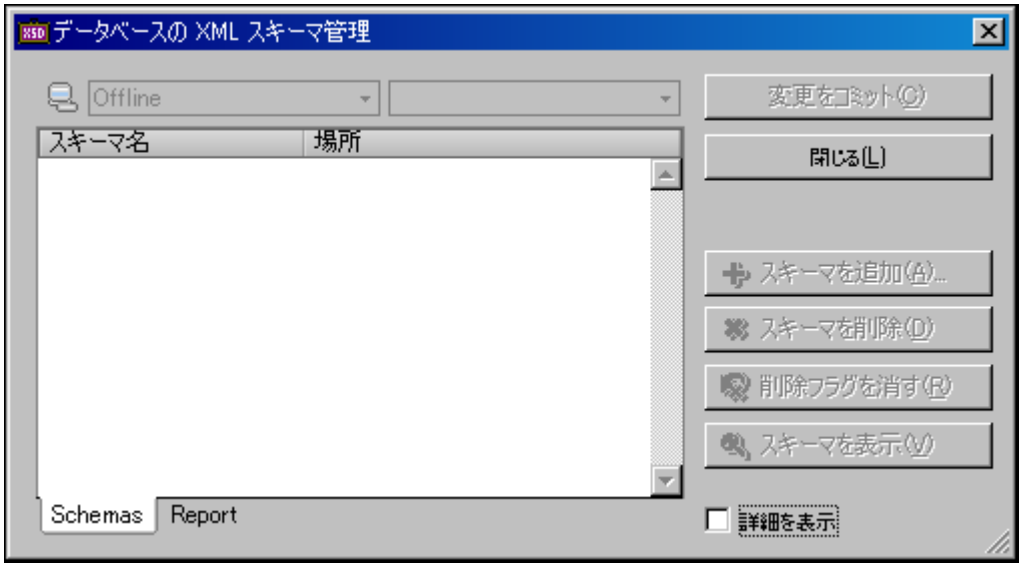
- IBM DB2 データベースへの接続

- XML スキーマを追加または削除するためのデータベーススキーマを選択
- スキーマの管理操作を行う

これらステップに関する詳細を以下に記します。

IBM DB2 データベースへの接続

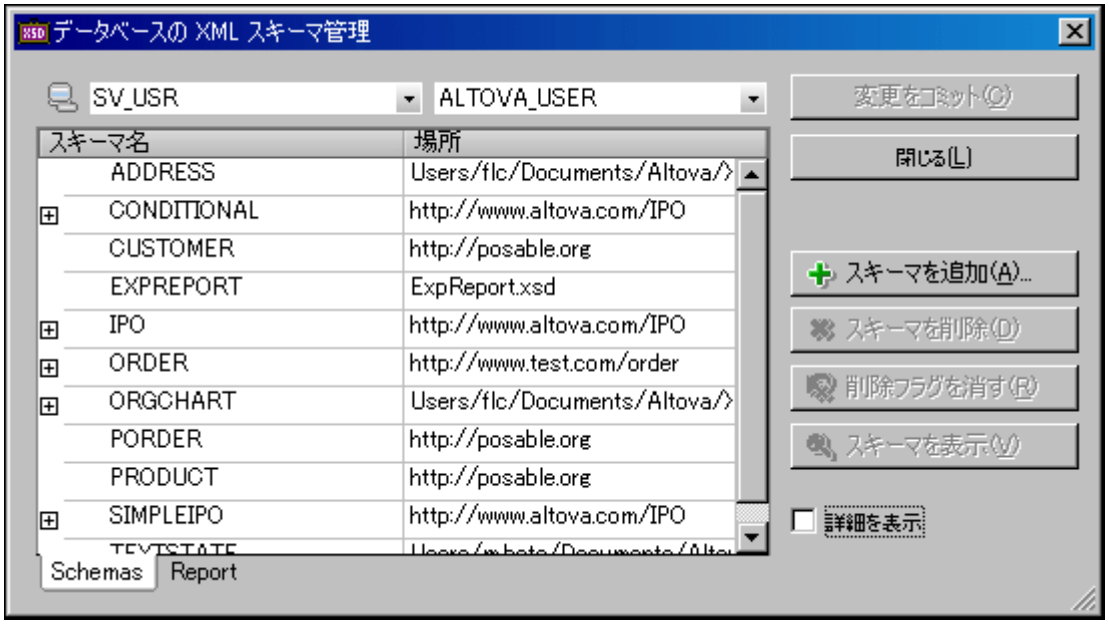
「XML スキーマの管理」コマンドをクリックすることで、データベースの XML スキーマ管理ダイアログが表示されます（*以下のスクリーンショットを参照*）。



目的のデータベースに対して接続がなされていない場合、まずは接続を行う必要があります。既に接続が確立されている場合、その情報がデータベースコンボボックスに表示されます。接続処理を開始するには、ダイアログのクイック接続アイコン をクリックします。クイック接続ダイアログの使用方法については、データベースへ接続 847 セクションを参照ください。

XML スキーマのリストを表示

IBM DB2 データベースへの接続が確立されると、データベースがコンボボックスの中に表示されます（*以下のスクリーンショットを参照*）。複数のデータベースへの接続が行われている場合、このコンボボックスから目的のデータベースを選択することができます。以下のスクリーンショットでは、SV_USR データベースが選択されています。



右にあるコンボボックスには、現在選択されているIBM DB2 データベースにあるデータベーススキーマのリストが表示されます。データベーススキーマがこのコンボボックスで選択されると、そのデータベーススキーマに登録されている全てのXML スキーマがメインペインにて表示されます。上のスクリーンショットでは、`Altova_User` データベーススキーマに登録された全てのXML スキーマが、配置されている場所とともに表示されています。詳細を表示チェックボックスにチェックを入れることで、更に詳細な情報がメインペインに表示されます。

XML スキーマの管理

選択されたデータベーススキーマに対して登録されているスキーマがメインペインにて表示されます。XML スキーマのリストが表示されると、スキーマをリストに加えたり、リストからスキーマを削除することができます。

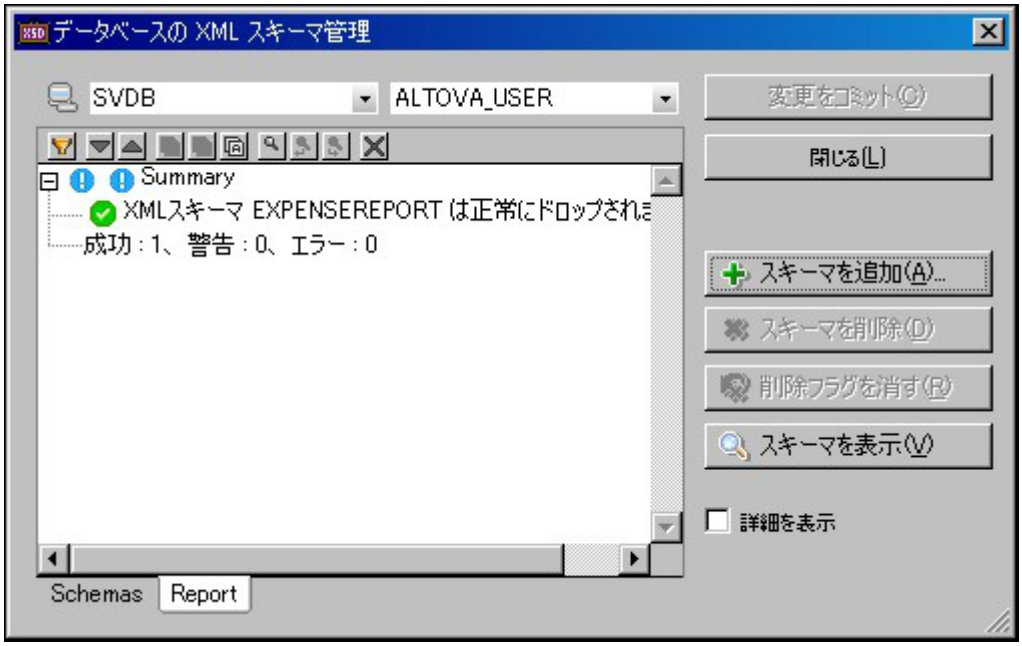
スキーマを追加するには、**「追加」**ボタンをクリックし、目的のスキーマファイルを指定します。選択されたスキーマファイルがリストに表示されます。**「変更をコミット」**ボタンをクリックすることで、新たに追加されたスキーマがデータベーススキーマに登録されます。

スキーマを削除するには、メインペインにてスキーマを選択し、**「スキーマを削除」**ボタンをクリックします。削除フラグがそのスキーマに割当てられ、変更が次にコミットされる際に、そのスキーマが削除されることを示します。フラグが立てられたスキーマを選択し、**「削除フラグを消す」**ボタンをクリックすることで、削除フラグを取り消すことができます。**「変更をコミット」**ボタンをクリックすると、削除フラグが立てられている全てのスキーマに対して、データベーススキーマの登録解除が行われます。

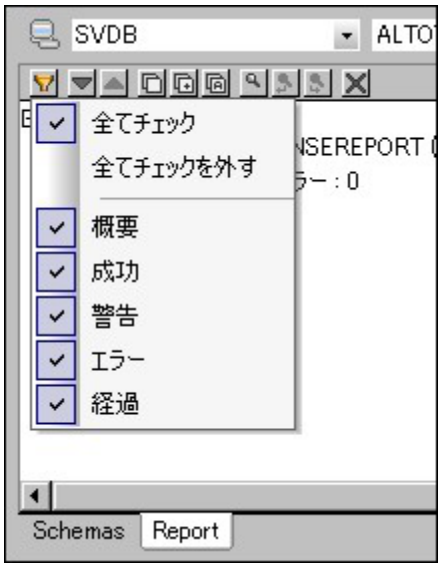
スキーマを表示ボタンをクリックすることで、スキーマがXMLSpy で開かれます。XML スキーマ管理ダイアログを閉じるには、**「閉じる」**ボタンをクリックしてください。

レポート

「変更をコミット」ボタンをクリックすると、加えられた変更に従うかたちでデータベースが変更されます。コミット処理のレポートは、レポートペインに表示され、処理が正常に行われたかの確認や、エラーが発生した際のデバッグに役立てることができます(*以下のスクリーンショットを参照*)。新たなレポートメッセージは、それまでのメッセージに続くかたちで表示されます。新たなレポートメッセージは、それまでのメッセージに続くかたちで表示されます。



レポートペインにあるツールバーには、表示のカスタマイズやナビゲーション、クリップボードにレポートをコピーしたり、テキストの検索やクリアを行うためのアイコンが含まれています(*以下のスクリーンショットを参照*)。



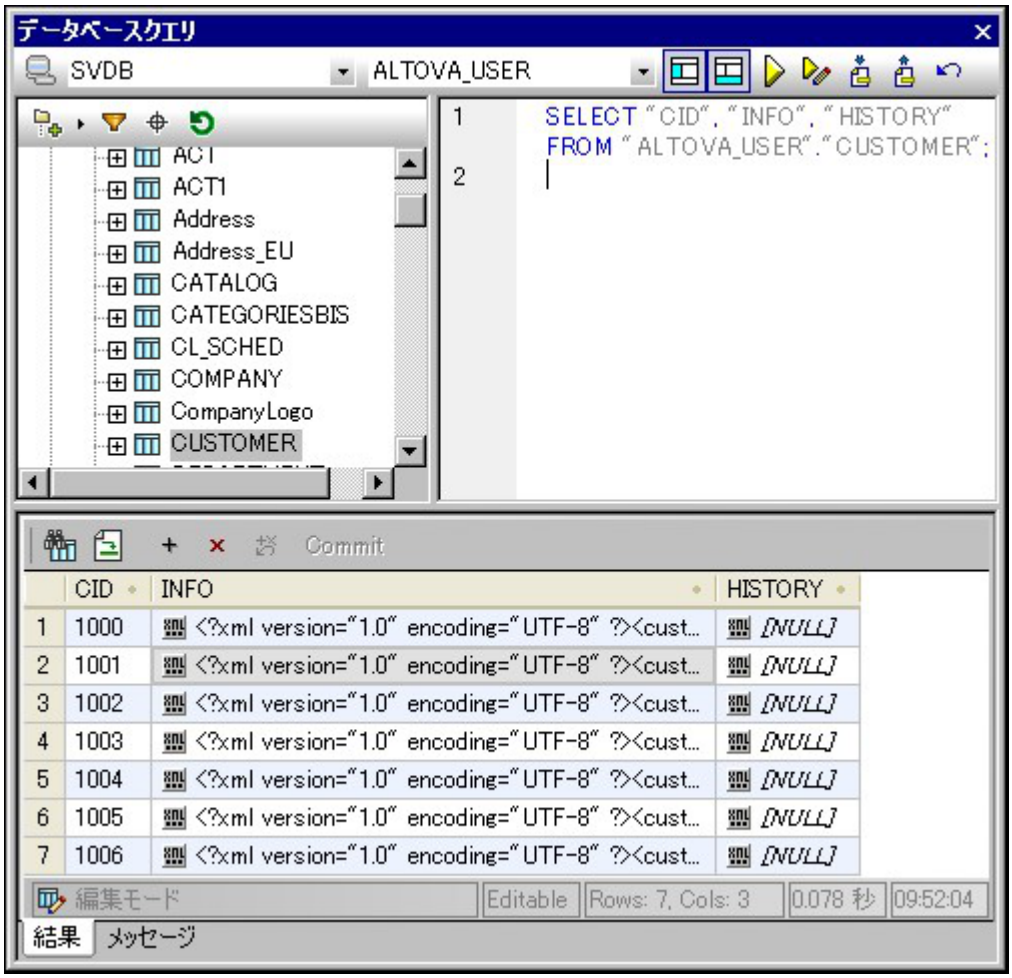
[フィルター] アイコンにより、特定の種類のメッセージの表示の切り替えをオンまたはオフします。「次」ならびに「前」アイコンにより、選択箇所を移動することができます。表示されているメッセージは、子コンポーネントを含めて(または含めないで)クリップボードにコピーすることができ、その後ドキュメントへの貼付けを行うことができます。[検索] 機能により、検索用語を指定し、この用語をリスト内上下に検索することができます。[クリア] アイコンにより、レポートペインの内容をクリアすることができます。

28.10.2.2 XML スキーマの割り当て

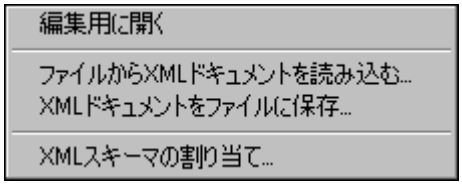
XML スキーマの割り当てコマンドにより、データベースクエリウィンドウの結果グリッドから、開かれている XML ファイルへスキーマを割り当てることができます。割り当てが行われると、そのスキーマに対して XML ファイルの検証を行うことができます。DB に対する割り当ては、XML ファイルが XMLSpy にて保存されたときに行われます。

DB XML ファイルを編集のために開く

データベースクエリウィンドウにてクエリが作成され、XML DB に対して編集を行うために実行されると、データベースクエリウィンドウの下部にある結果グリッドからデータベース内部にある XML ファイルへアクセス、そして編集することができます（以下のスクリーンショットを参照）。



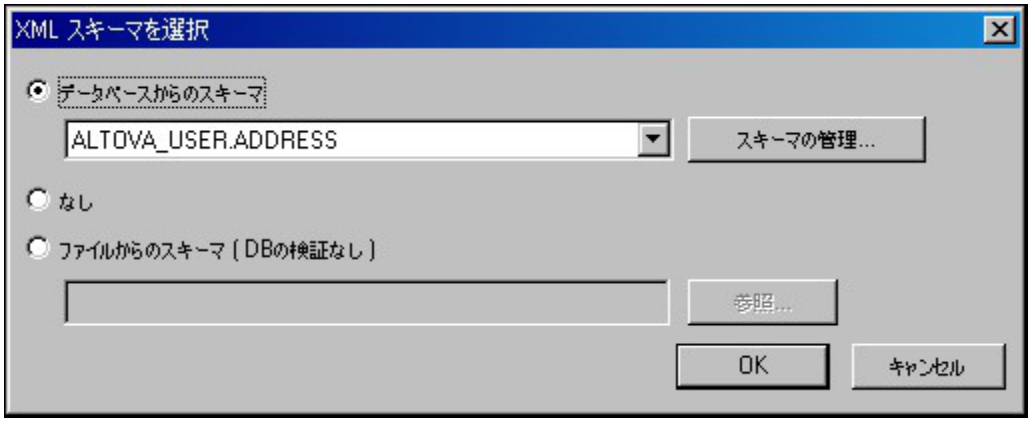
XML アイコン をクリックすることで、以下のメニューが表示されます。



「開いて編集」コマンドを選択すると、そのXMLドキュメントがXMLSpy で開かれ、編集を行うことができます。

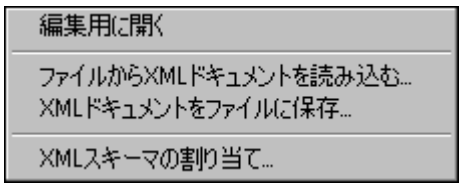
スキーマを DB XML ファイルへ割り当てる

DB XML ファイルが編集のためにXMLSpy で開かれると、「IBM DB2 | XML スキーマの割り当て」コマンドが有効になります。XMLドキュメントがXMLSpy でアクティブになっている状態で、「XML スキーマの割り当て」コマンドをクリックすると、XML スキーマを選択ダイアログが表示されます(以下のスクリーンショットを参照)。



(データベースからのスキーマコンボボックスにあるドロップダウンリストにて表示される) データベースに保管されている中からスキーマを選択することができる他にも、参照ボタンから外部ファイルを選択することもできます。「OK」をクリックすると、スキーマがXML ファイルへ割り当てられます。割り当て情報はXML ファイルへは書き込まれないことにご注意してください。ファイルを開く際にデータベースクエリツールバーにあるXML の変更を自動コミットアイコン が有効になっていた場合、XMLSpy にてXML ファイルが保存されるとともに、スキーマの割り当てがデータベースへ保存されます。スキーマの割り当て情報はデータベースへ書きこまれ、XML ファイルへは書き込まれないことにご注意してください。

メモ: データベースクエリウィンドウの結果グリッドには、XML スキーマの割り当てコマンドが用意されており、スキーマを DB XML ファイルへ割り当てることもできます(以下のスクリーンショットを参照)。



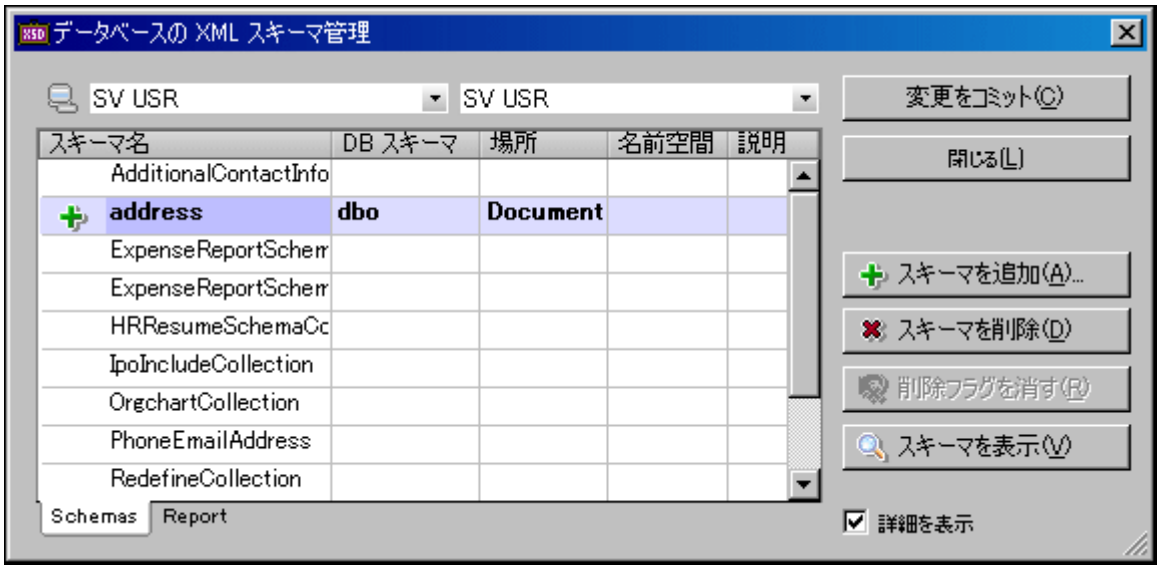
これら2つの XML スキーマ割り当てコマンドにおける違いは、「DB | IBM DB2」メニューのコマンドでは、XML ファイルの編集をしながらも XML スキーマの割り当てを行うことが可能で、XML ドキュメントの編集中にスキーマの割り当てを変更し、そのまま XML ドキュメントの検証を行うことが可能だということです。

28.10.3 SQL Server

SQL Server メニューアイテムには、XML スキーマの管理コマンドが収められています。

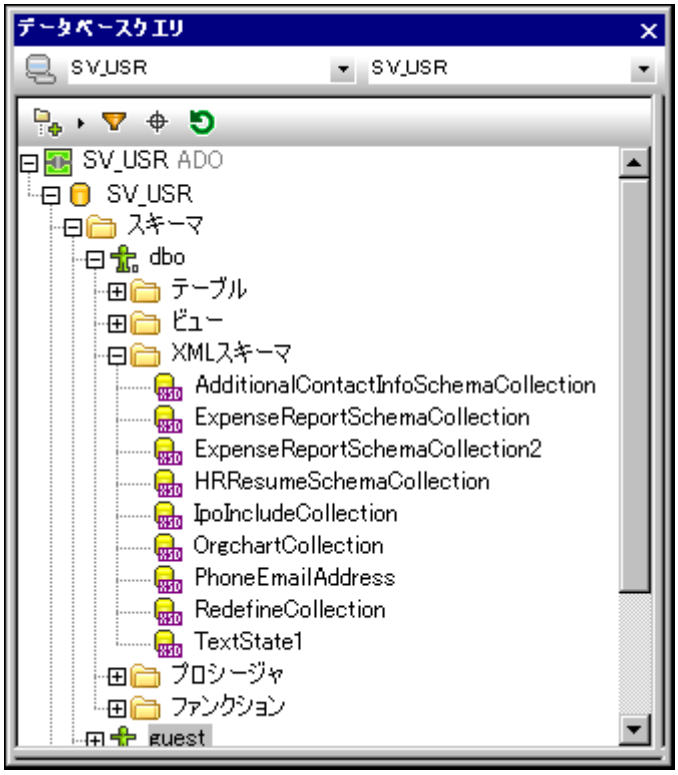
28.10.3.1 Manage XML Schemas

データベースの XML スキーマ管理ダイアログでは、XML データベースのスキーマリポジトリへ XML スキーマを追加したり削除することができます。データベースへの接続が行われると、XMLSpy に搭載されているデータベースの XML スキーマ管理ダイアログにて XML スキーマの管理を行うことができます。



ダイアログボックスに表示されているクイック接続アイコン を使用することで、クイック接続ウィザード 848 からデータソースへの接続することができます。1つ以上の接続が現在存在する場合、必要な接続は左側のコンボボックスから選択することができます。必要なルートオブジェクトをコンボボックスの右側から選択することができます。ルートオブジェクトのためのレポジトリ内にある全ての XML スキーマは、ダイアログボックスに表示されています。名前、場所、各スキーマの名前空間のためのオプションがリストされます。

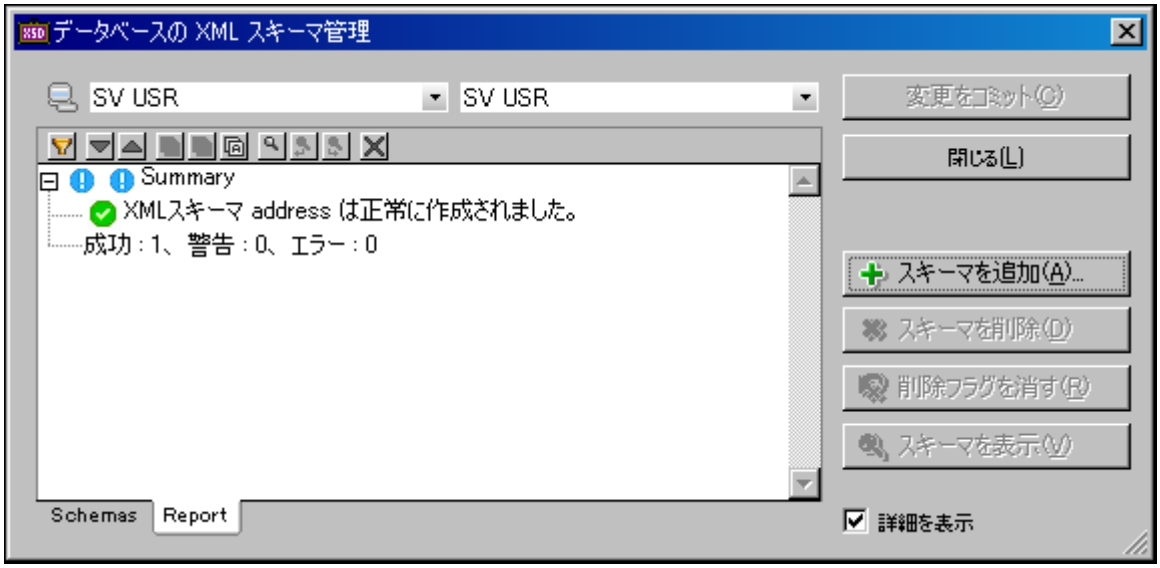
保管されているスキーマを、データベースクエリウィンドウ内で閲覧することができますが、管理することはできません（下のスクリーンショット）。スキーマを管理するには、データベースのための XML スキーマ管理ダイアログを使用します。



XML スキーマ管理ダイアログで以下を行うことができます:

- **スキーマの追加** ボタンを使用してスキーマを追加します。選択されたスキーマは、リストに追加され、追加のためにマークされます。
- 「**スキーマをドロップ**」ボタンを使用して、リスト内のスキーマを削除のためにマークすることができます。「**削除フラグを消す**」ボタンを使用すると、ドロップフラグが削除されます。
- **スキーマをビュー** ボタンをクリックして、スキーマビュー内で選択されたスキーマを開きます。
- 「**変更をコミット**」ボタンを使用して追加とドロップ(削除) の変更をコミットします。

変更がコミットされた後、レポートタブ内でコミットアクションに関するレポートを確認することができます(下のスクリーンショット)。



28.10.4 Oracle XML DB

XMLSpy はOracle XML Db データベースへの接続およびクエリを許可しています。

次のデータベース機能がサポートされています:

- XML スキーマを Oracle XML Db に追加および登録します。Oracle XML DB クライアントを XMLSpy を使用してスキーマを登録するためにインストールする必要があります。
- スキーマを開き削除します
- XPath ステートメント (DBUri) を使用したQuery データベース
- (W ebDAV を使用した)参照 XML ドキュメント
- データベース内に保存されたスキーマをベースにして作成された XML ドキュメント

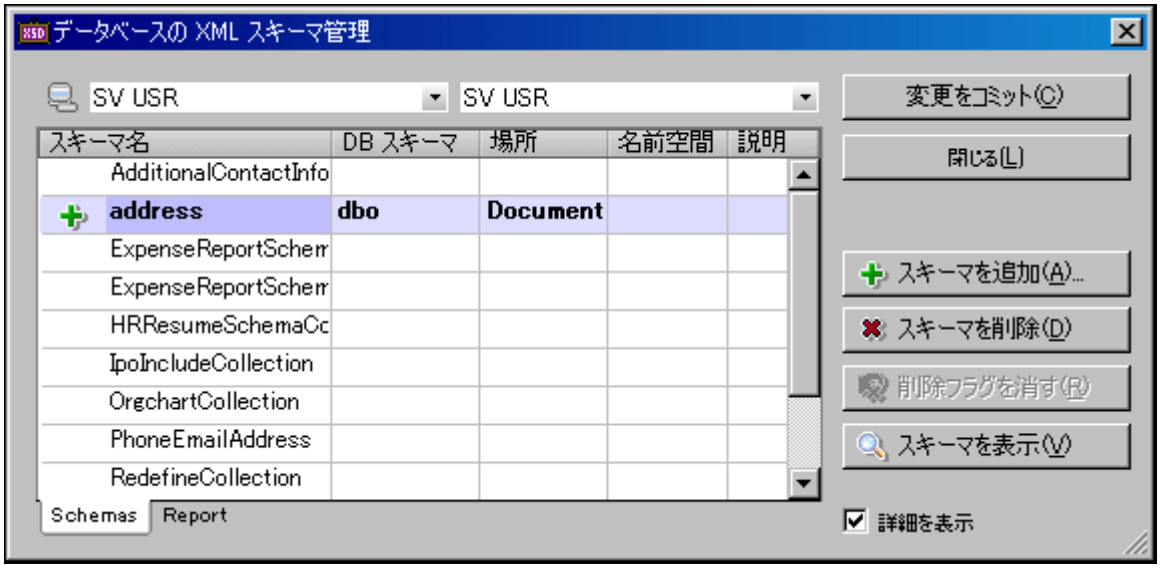
一般的なインストールのプロセス:

- XMLSpy をダウンロードし、インストールします。
- Oracle サーバーを(必要な場合)インストールします。
- Oracle データベースを作成します

28.10.4.1 XML スキーマの管理

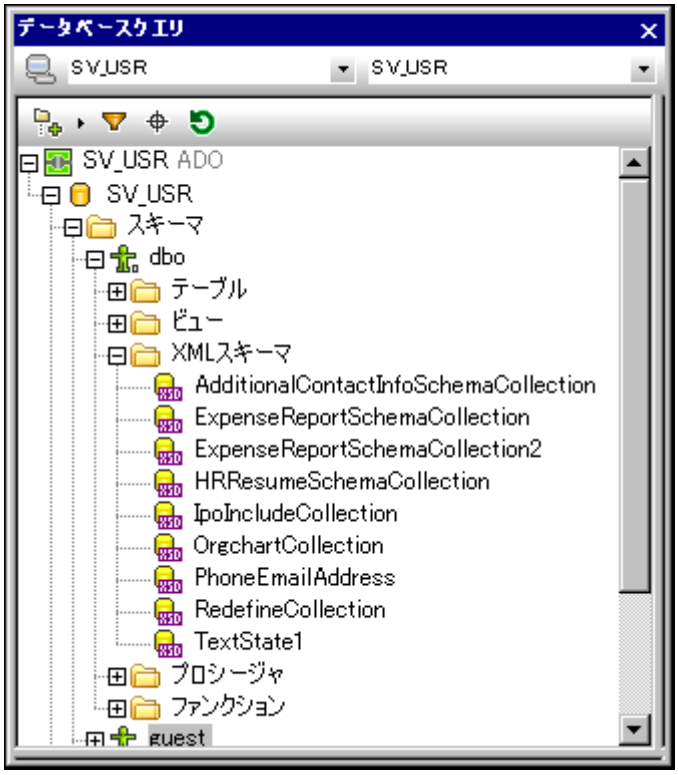
データベースの XML スキーマ管理ダイアログでは、XML データベースのスキーマリポジトリへ XML スキーマを追加したり削除することができます。データベースへの接続が行われると、XMLSpy に搭載されているデータベースの XML スキーマ管理ダイアログにて XML スキーマの管理を行うことができます。

(C) 2015-2021 Altova GmbH



ダイアログボックスに表示されているクイック接続アイコン を使用することで、クイック接続ウィザード 848 からデータソースへの接続することができます。1つ以上の接続が現在存在する場合、必要な接続は左側のコンボボックスから選択することができます。必要なルートオブジェクトをコンボボックスの右側から選択することができます。ルートオブジェクトのためのレポジトリ内にある全てのXML スキーマは、ダイアログボックスに表示されています。名前、場所、各スキーマの名前空間のためのオプションがリストされます。

保管されているスキーマを、データベースクエリウィンドウ内で閲覧することができますが、管理することはできません（下のスクリーンショット）。スキーマを管理するには、データベースのためのXML スキーマ管理ダイアログを使用します。

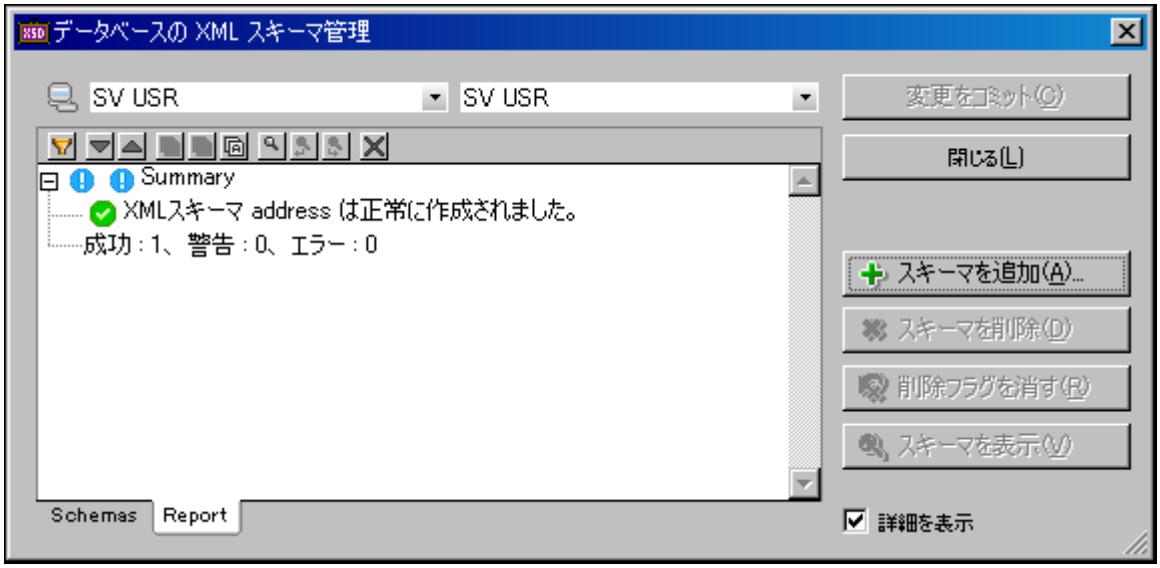


XML スキーマ管理ダイアログで以下を行うことができます:

- **スキーマの追加** ボタンを使用してスキーマを追加します。選択されたスキーマは、リストに追加され、追加のためにマークされます。
- 「**スキーマをドロップ**」ボタンを使用して、リスト内のスキーマを削除のためにマークすることができます。「**削除フラグを消す**」ボタンを使用すると、ドロップフラグが削除されます。
- **スキーマをビュー** ボタンをクリックして、スキーマビュー内で選択されたスキーマを開きます。
- 「**変更をコミット**」ボタンを使用して追加とドロップ(削除) の変更をコミットします。

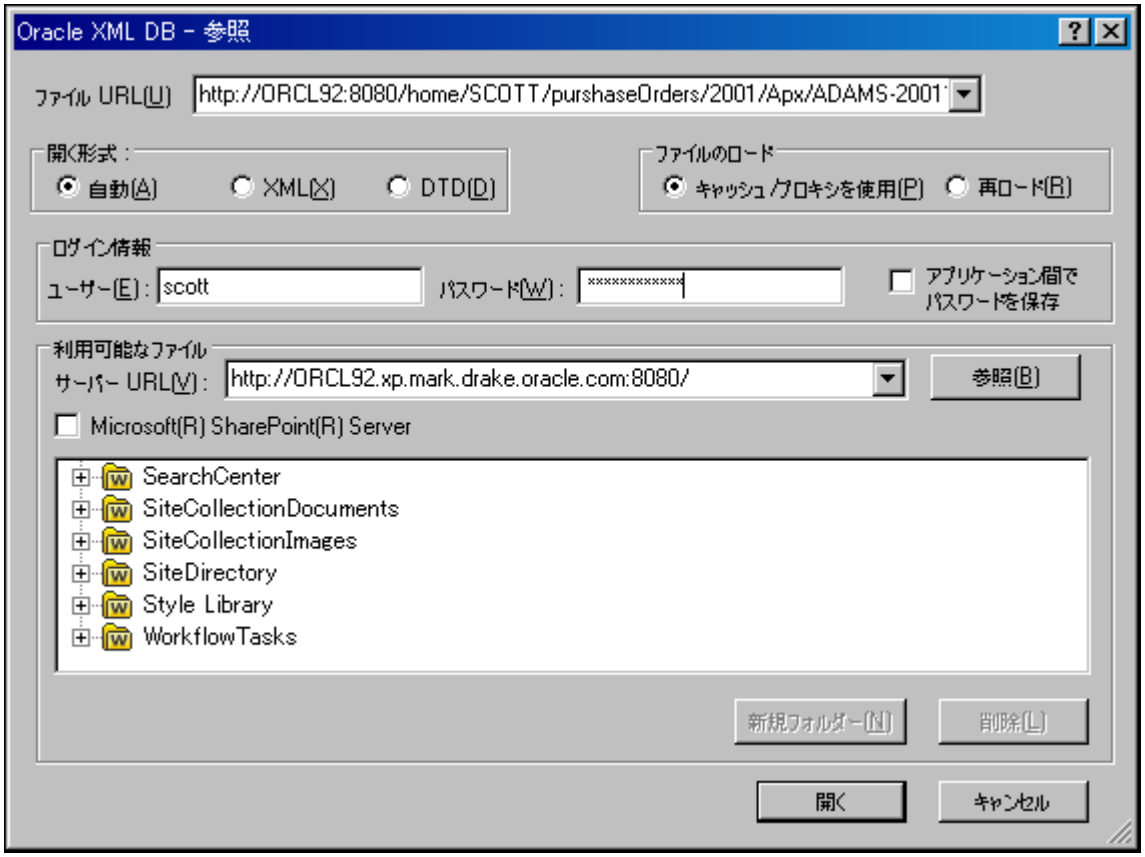
変更がコミットされた後、レポートタブ内でコミットアクションに関するレポートを確認することができます(下のスクリーンショット)。

(C) 2015-2021 Altova GmbH



28.10.4.2 参照 Oracle XML ドキュメント

このコマンドにより、サーバー上で使用することのできる XML ドキュメントを参照することができます。以前にクエリされたデータベースまたはリストされたスキーマが存在する場合、サーバーの詳細は、自動的に入力されます。それ以外の場合は、手動で情報を入力します。

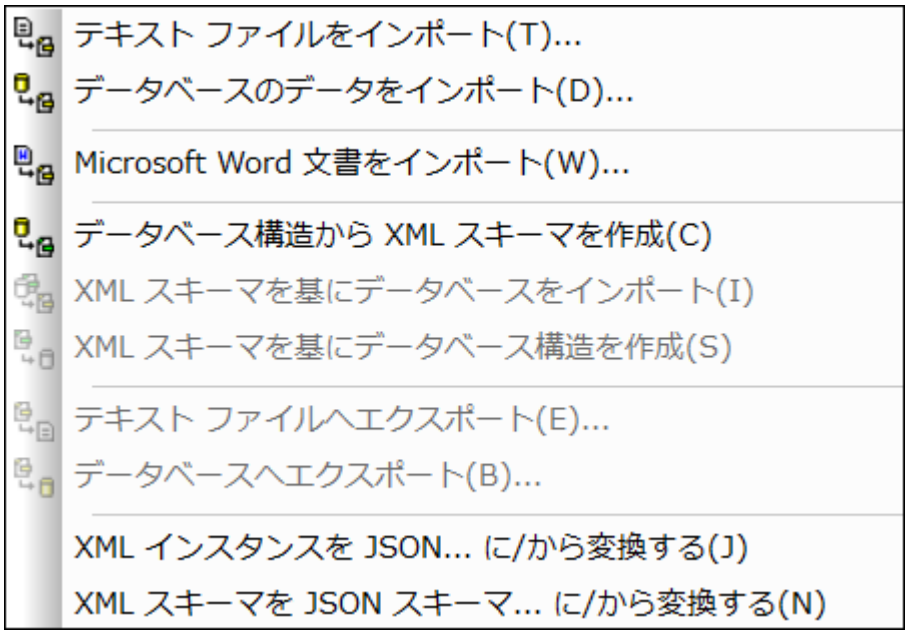


特定のXML ファイルをビューするために、ツリービューを使用します。ツリービューで開くファイルをダブルクリックします。ファイル を「開く」をクリックしても同じ結果が得られます。「**新規のフォルダー**」ボタンは、新規のフォルダーを追加します。「**削除**」ボタンは、現在選択されているXML ファイルを削除します。

(C) 2015-2021 Altova GmbH

28.11 変換メニュー

変換 メニューでは、以下に示される操作を行うための、パワフルなデータ交換機能が提供されます:

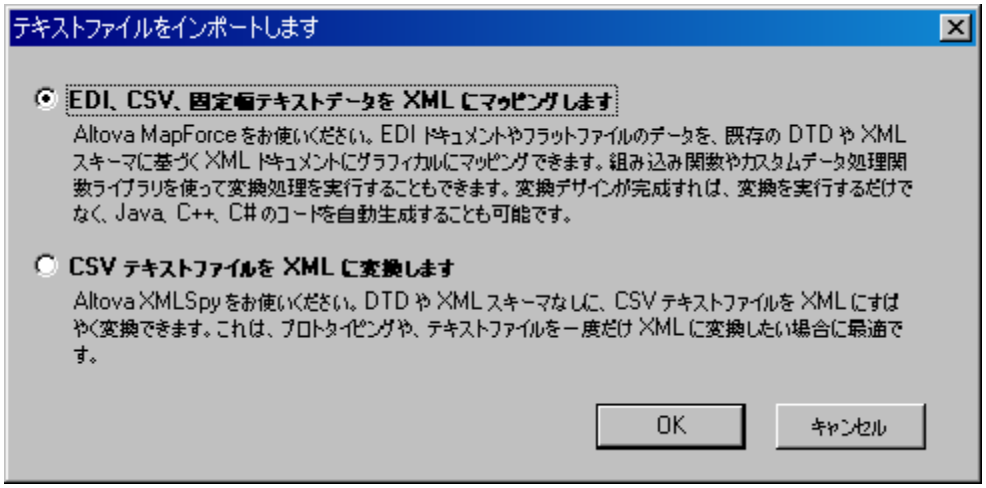


- テキスト、ワードプロセッサードキュメント、データベース、XML ファイルのインポートならびにエクスポート
- 既存の XML スキーマをベースにした[データベースのインポート](1343)
- 既存データベースの構造をベースに [XML スキーマの作成](1339)
- 既存の XML スキーマをベースに [データベース構造を作成](1344)。
- [XML インスタンスと JSO インスタンス間](1352) と[XML スキーマと JSON スキーマ間の](1355) の変換。

28.11.1 テキストファイルのインポート

このコマンドにより、構造化テキストファイルを XMLSpy へインポートし、直ちに XML フォーマットへ変換することができます。古いシステムからレガシーなデータをインポートするためにこの機能を利用することができます。以下にテキストファイル内のデータを XML ドキュメントとしてインポートする為の手順を記します。

1. 「**変換 | テキストファイルのインポート**」を選択します。以下のダイアログが表示されます:

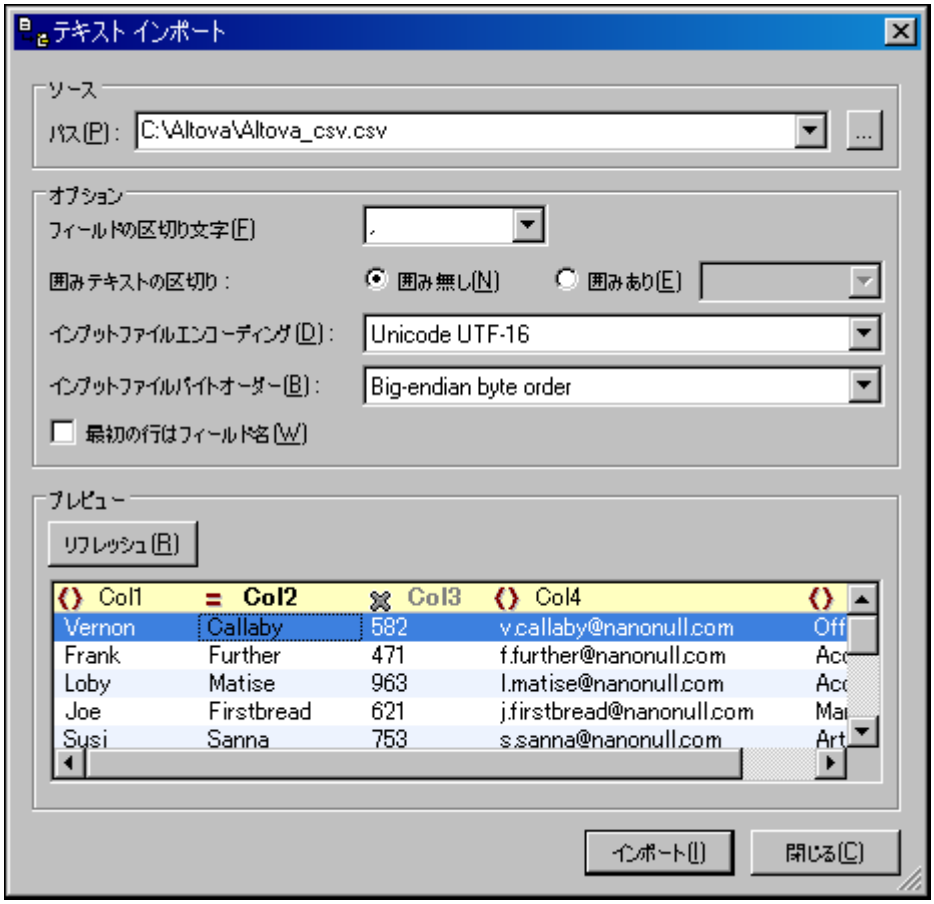


2. 以下から選択を行います:

- EDI、CSV、固定長テキストデータを XML へマッピング(このオプションを選択するには Altova MapForce がインストールされていなければなりません)
- CSV テキストファイルを XML へ変換

3. 「OK」をクリックします。テキストインポートダイアログが表示されます(以下のスクリーンショットを参照)。以下にこのダイアログの使用方法を記します。

(C) 2015-2021 Altova GmbH



パス

インポートするファイルへのパスをパス テキストボックスへ入力するか、テキストボックスの右側にある参照ボタンを使用してファイルを選択します。ファイルが選択されると、XML ファイルのプレビューが、グリッドビュー形式で*プレビュー* ペインに表示されます。このダイアログにおけるオプションの変更は、プレビューへ直ちに反映されます。

区切り文字

テキストファイルをインポートするには、ファイル内部のカラムやフィールドを分割している区切り文字を指定する必要があります。標準的な行の区切り(CR、LF、または CR+LF)は <%XMLSPY% により自動的に識別されます。

文字列の引用符

レガシーなシステムからエクスポートされたテキストファイルの中には、数値データとの違いを明確にするために、文字列の値を引用句により囲んでいるものもあります。このような場合、ファイルの中で使用されている引用句を指定することで、データがインポートされた際にそれら引用句を自動的に削除することができます。

エンコード

データは（XML ドキュメントで通常使用される）ユニコード[2094] へ変換されるため、ファイル内で現在使用されているエンコーディングを指定する必要があります。

バイトオーダー

16ビットまたは32ビットのユニコード（UCS-2、UTF-16、または UCS-6）ファイルをインポートする場合、リトルエンディアンとビッグエンディアンバイトオーダーから選択を行うことができます。

最初の行はフィールド名

テキストファイル内にある最初の行にてフィールド名が記述されることもあります。この場合、このチェックボックスにチェックを入れてください。

プレビュー

プレビューペインでは、カラムヘッダーの名前をダブルクリックして編集を行うことで、カラム名を変更することができます。カラムヘッダーは、XML ドキュメントにおける要素または属性名となります。各カラムのヘッダーにあるアイコンをクリックすることで、カラムが要素なのか属性なのか、また XML ドキュメントにインポートするのかを選択することもできます。上のスクリーンショットでは、Col1 が要素、Col2 が属性としてインポートし、Col3 をインポートしないようにセットされています。

オプションの指定が終わったら、「**インポート**」をクリックします。インポートされたデータが XML ドキュメントに変換され、グリッドビューにて表示されます。

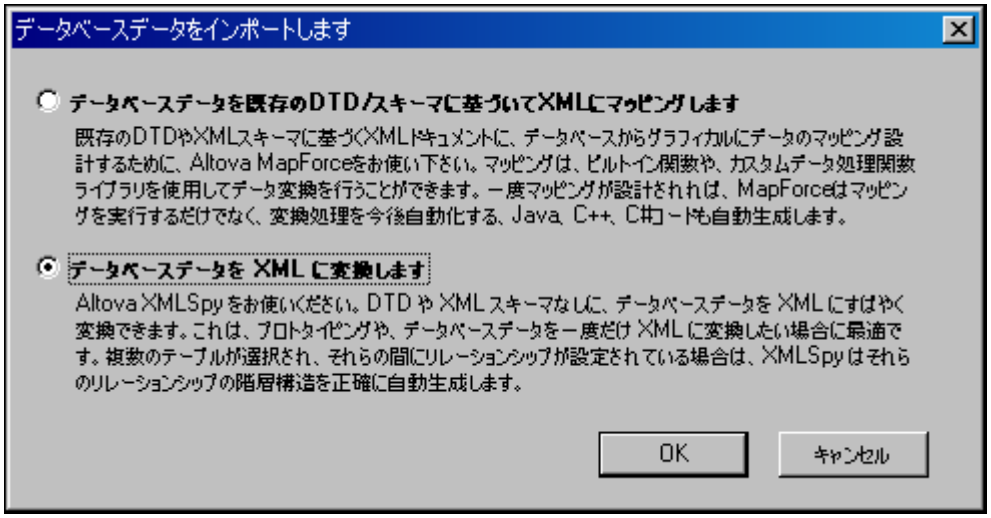
28.11.2 データベースのデータをインポート

「**データベースのデータをインポート**」コマンドにより、様々な種類のデータベースからデータを XML ファイルにインポートすることができます。インポートメカニズムは、以下のような2つのステップにより構成されます:

1. データベースへの接続が作成されます。詳細に関しては[データベースへ接続](847)を参照してください。
2. [インポートするデータを選択](1335)します。

以下の操作によりデータベースのデータをインポートします:

1. データベースのデータをインポートコマンドをクリックすると、データベースデータをインポートするためのダイアログが表示されます（以下のスクリーンショットを参照）。

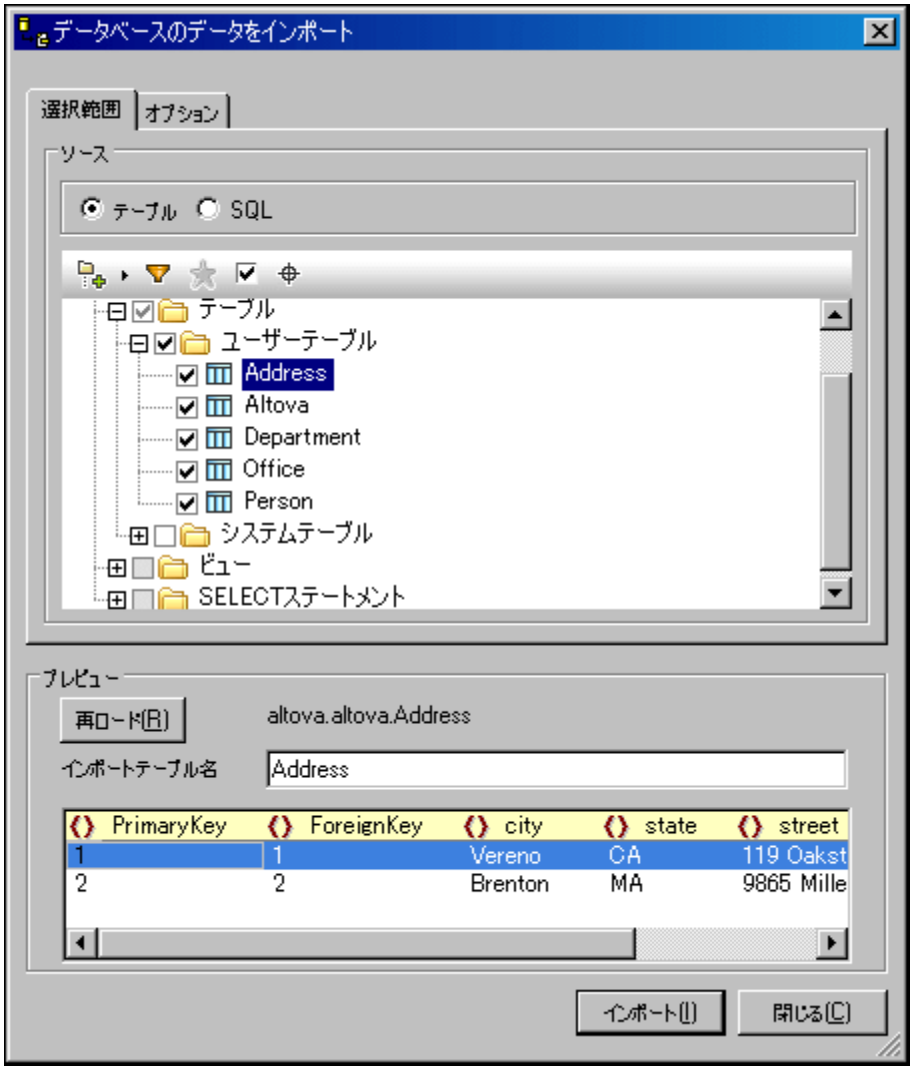


2. データベースのデータを XML に変換を選択して、OK をクリックします。データソースへの接続ダイアログが表示されます。データベースを既存の DTD/スキーマに基づいて XML にマッピングオプションを使ってマッピングを行うには、Altova MapForce が必要になります。
3. データソースへの接続ダイアログにて、データベースへの接続を確立します。異なる種類のデータベースに対する接続の方法については、データベースへ接続 847 セクションを参照ください。
4. データベースへの接続が確立されると、データベースのデータをインポートダイアログにて表示されているタブやウィンドウにより、データベースにあるデータのインポートを始めます。以下にこれらオプションに関する説明を記述します。オプションの設定を完了すると、インポートボタンをクリックしてデータベースのデータをインポートします。

データの選択とインポートオプション

インポートオプションを指定するデータベースのデータをインポートダイアログは、2つのパートにより構成されています:

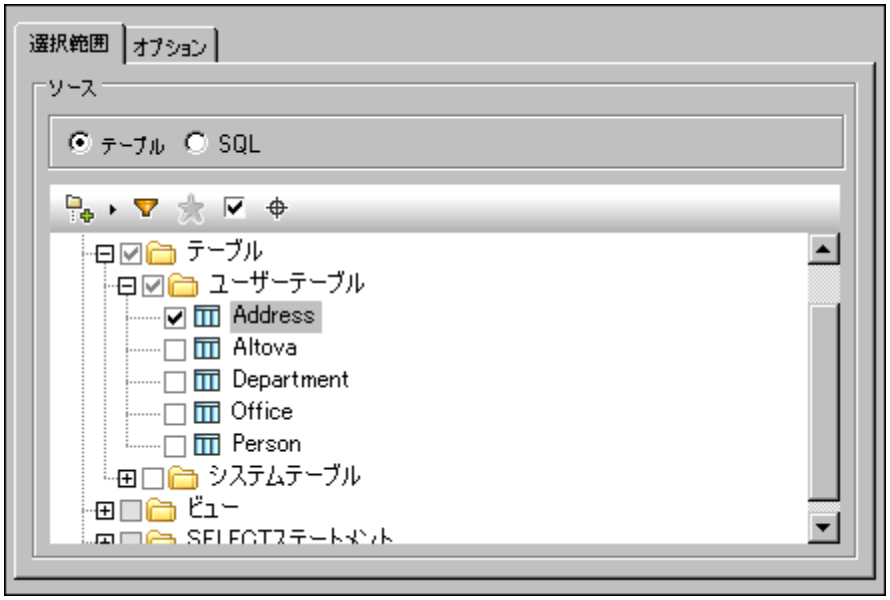
- 選択範囲とオプションという2つのタブがある上部パート。
- データの選択範囲とインポートオプションに従ったデータを表示するプレビューウィンドウがある下部パート。



選択範囲タブ

(上のスクリーンショットで示される) 選択範囲タブでは、(以下のスクリーンショットにある) ソースペインによりデータベース上のテーブルや、目的のテーブルを選択するためのSQL ステートメントが表示されます。

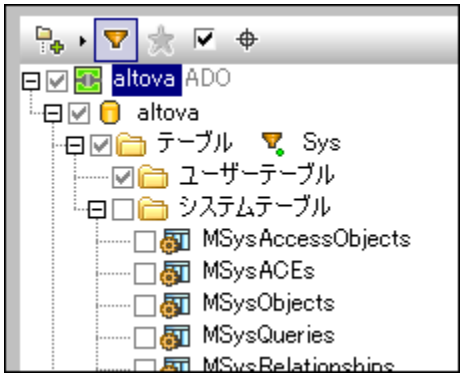
(C) 2015-2021 Altova GmbH



テーブルビューでは、テーブルのチェックボックスにチェックを入れることで、インポートするテーブルをデータベースから選択することができます(上のスクリーンショットを参照)。テーブルのコンテンツはプレビューペインに表示されます。テーブルの選択は、プレビューペインにて更にフィルタリングすることもできます。

データベース構造の表示方法を変更したり、フィルタリングを行うこともできます。ソースペインにあるレイアウトアイコン により、データベースオブジェクトを、(i) オブジェクトの型をベースにしたフォルダー、(ii) スキーマフォルダー無しで、オブジェクトの型をベースにしたフォルダー、(iii) フォルダー無しの階層構造、(iv) 他のテーブルとのリレーションシップをベースにしたテーブルのカテゴリーにより管理することができます。

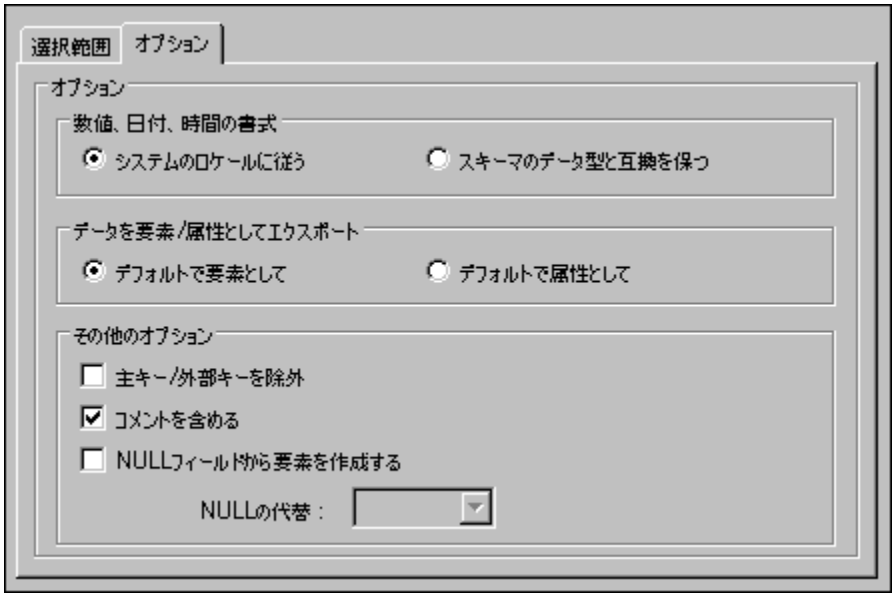
フォルダーコンテンツのフィルター アイコンにより、選択されたフォルダーに対してフィルタリングを行うことができます(以下のスクリーンショットにあるテーブルフォルダーを参照)。テーブルの隣にあるフィルターアイコンをクリックすることで、フィルタリングの種類を選択することができます。以下のスクリーンショットでは、Sys というテキストが名前に含まれているオブジェクトだけを表示するようにフィルタリングが行われています。



オブジェクトロケーターアイコン をクリックすると、検索フィールドの様に動作するテキストフィールドが表示されます。テキスト文字列を入力すると、入力された文字列を含む全てのオブジェクトが表示されます。ドロップダウンリストにて表示されているオブジェクトを選択することで、ツリー内の対応するオブジェクトがハイライトされます。

オプションタグ

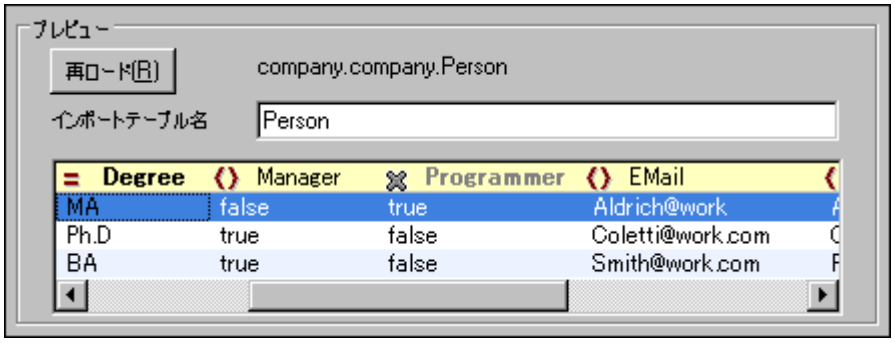
(以下のスクリーンショットで示される)オプションタブでは、数値、日付、時間の値をインポートする方法を指定することができます。要素や属性としてデータをインポートするのか、コメントやNULL フィールドもインポートの中に含まれるかを指定することができます。



NULL フィールドをインポートする場合、その代替となるXML の値を入力することができます。

プレビューペイン

(以下のスクリーンショットで示される)プレビューペインでは、選択範囲タブにて現在選択されているテーブルの構成が表示されます。選択範囲タブにて新しいテーブルが選択された場合、再ロードボタンによりプレビューが更新されます。



要素または属性としてフィールドをインポートするように指定することができ、また、列名の左横のシンボルをクリックして、インポートされないように指定することもできます。要素、属性、またはオプションの無視から選択することができます。上のスクリーンショットでは、city フィールド、が例えば、要素としてインポートされるようにセットされていますが、street フィールドは属性としてインポートされるように設定されています。

データ型の変換

データベースのデータ型からXML スキーマのデータ型への変換に関する情報については、付録[2081]を参照ください。

28.11.3 Microsoft Word ドキュメントをインポート

このコマンドにより任意のW ord ドキュメントをインポートし、XML フォーマットへ変換することができます。このオプションを使用するには、お使いのコンピューターにMicrosoft W ord またはMicrosoft Office がインストールされていなければなりません。

このコマンドを選択すると、ファイルを開くダイアログが表示されます。インポートするW ord ドキュメントを選択してください。
XMLSpy により、CSS スタイルシートとともにXML ドキュメントが生成されます。W ord における各段落から、Microsoft W ord 内の段落スタイルに対応した名前のXML 要素が生成されます。

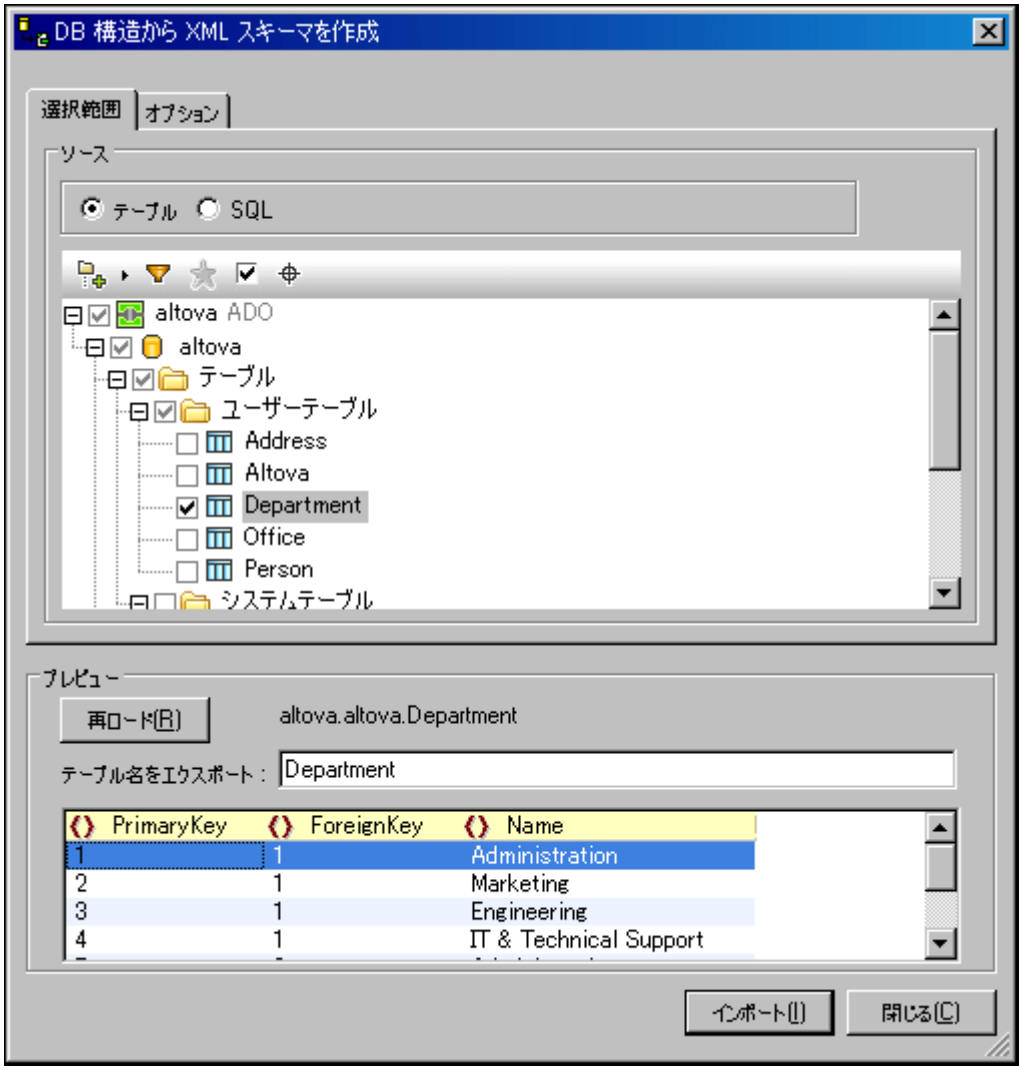
28.11.4 データベース構造から XML スキーマを作成

「**データベース構造から XML スキーマを作成**」コマンドにより、様々な種類のデータベース構造からXML スキーマを作成することができます。XML スキーマの作成メカニズムは、2つのステップにより構成されます:

1. データベースへの接続が確立されます。
2. データベースのデータ選択や XML スキーマのオプションを指定します。

以下の操作により、データベース構造からXML スキーマを作成することができます:

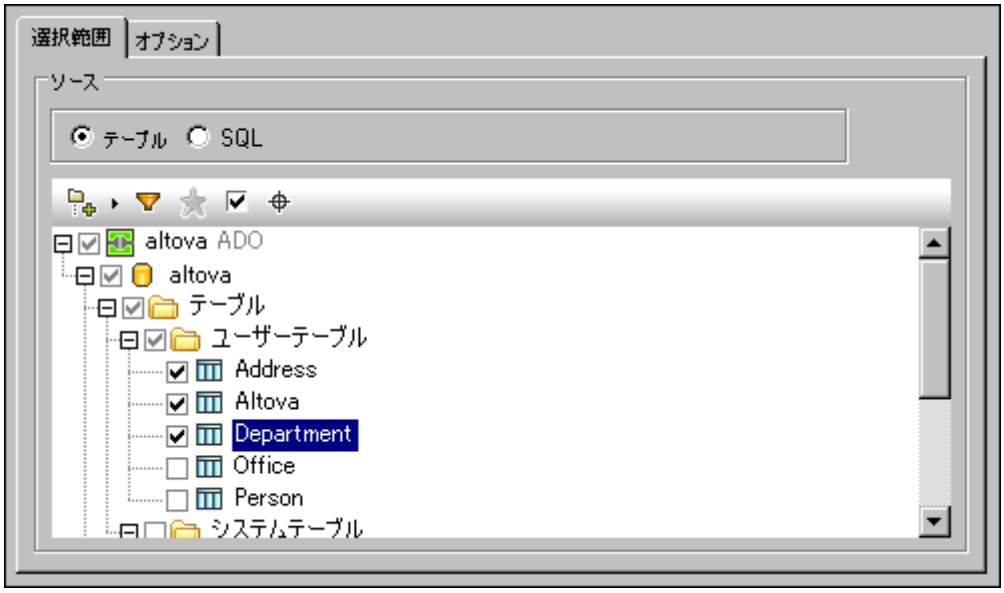
1. **データベース構造から XML スキーマを作成**コマンドをクリックすると、データベース接続ウィザードボックスが表示されます。
2. データベースへの接続ウィザードにてデータベースへの接続を確立します。異なる種類のデータベースに対する接続の方法についてはデータベースへ接続 847 セクションを参照ください。
3. データベースへの接続が確立されると、DB 構造から XML スキーマを作成ダイアログが表示され、タブやペインによりインポートするデータベース構造を選択することができます（以下のスクリーンショットを参照）。オプションについては以下を参照ください。**インポート**ボタンをクリックすることで、データベースのデータがインポートされます。



選択範囲タブ

(上のスクリーンショットで示される) 選択範囲タブでは、(以下のスクリーンショットにある) ソースペインによりデータベース上のテーブルや、目的のテーブルを選択するためのSQL ステートメントが表示されます。.

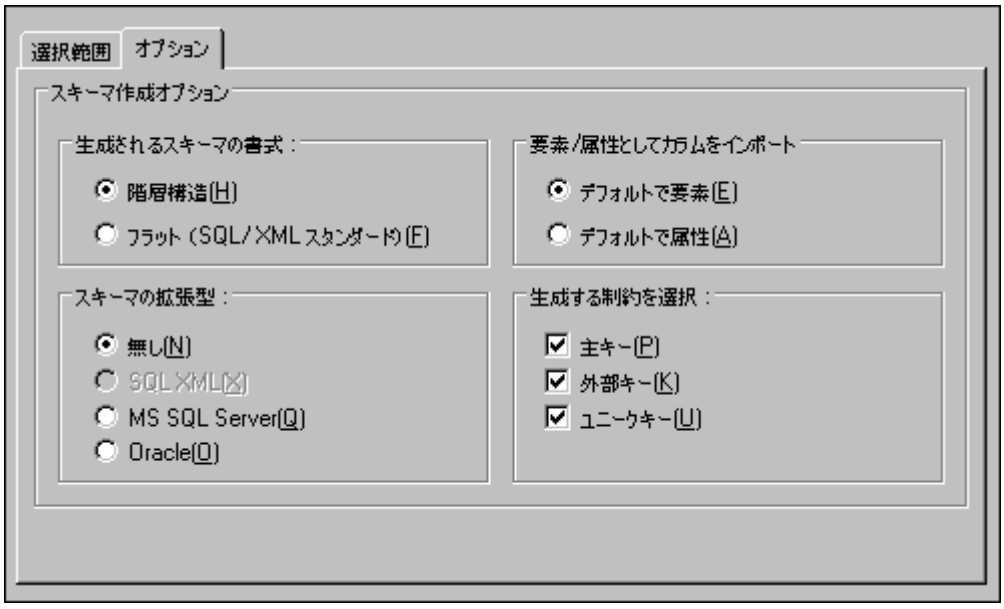
(C) 2015-2021 Altova GmbH



テーブルビューでは、テーブルのチェックボックスにチェックを入れることで、インポートするテーブルをデータベースから選択することができます(上のスクリーンショットを参照)。テーブルのコンテンツはプレビューペインに表示されます。テーブルの選択は、プレビューペインにて更にフィルタリングすることもできます。データベースデータのインポート[1334]のセクションに記述されている方法を使用することで、ソースペインの表示方法を変更することもできます。

オプションタグ

(以下のスクリーンショットにある)オプションタブでは、スキーマの書式、その拡張型、カラムを要素または属性としてインポートするのか、スキーマにてデータベースの制約を生成するかを指定することができます。



スキーマの書式:フラット(SQL/XML 標準)か階層構造スキーマを選択することができます。

- **フラット** スキーマモデルはXML-Related Specification (SQL/XML) という名前のISO-ANSI 作業草案(これ以降は「SQL/XML 作業草案」と表記)をベースにしています。SQL/XML 作業草案では、データベースを XML へマッピングする方法が定義されます。スキーマにおいてリレーションシップは同一性制約を使って定義され、要素への参照は行われません。そのため、スキーマはデータベースのツリー型ビューにも似たフラットな構造となります。ANSI ストアで仕様を購入することができますSQL/XML 作業草案に関するより詳細な情報は、以下のリンクを参照ください: www.sqlx.org。
- **階層構造** スキーマモデルでは、何らかのオブジェクトに依存しているテーブルがインデントされた子要素としてコンテンツモデルに表示され、ツリービューの様な形式でテーブルの依存関係が視覚的に表示されます。テーブルの依存関係は同一性制約タブにも表示されます。スキーマにおいてテーブルはグローバル要素として表示され、テーブルのカラムはこれらグローバル要素の子要素または属性となります(カラムを要素とするか属性とするかはユーザーにより決定されます)。リレーションシップは階層構造形式で作成され、あるテーブルにある外部キーのフィールドは、そのテーブルを表すグローバル要素への参照となります。

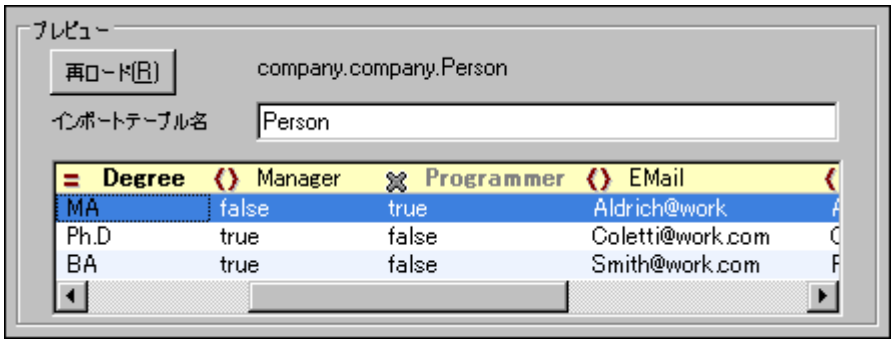
スキーマの拡張型: スキーマ拡張情報とはデータベースから読み取られ、スキーマ内部では注釈データまたは属性として埋め込まれる情報のことです。スキーマを生成するにあたり、4つの拡張型オプションから選択を行うことができます: (i) 拡張情報無し、(ii) SQL/XML 拡張、(iii) MS SQL Server 拡張、(iv) Oracle 拡張。それぞれのオプションについて以下に記述します:

- 無し: データベースから新たに取得する情報はありません。
- SQL XML: : SQL/XML 拡張は、フラット構造でスキーマの生成を行うときのみ使用することができます。拡張情報は SQL/XML 作業草案に従ったかたちで注釈内に保管されます。(INCITS/ISO/IEC 9075-14-2008).
- **MS SQL Server:** Microsoft SQL Server を選択することで、SQL Server 拡張が生成されます。詳細については、SQL サーバーブックオンライン とMSDN のXSD スキーマの注釈付けについて を参照ください。以下にある注釈データのサブセットがスキーマにて生成されます: sql:relation、sql:field、sql:datatype、sql:mapped。
- **Oracle:** : Oracle データベースを使用していると、Oracle 拡張がデフォルトで選択されます。拡張情報は属性内に保管されます。これら属性に関する情報については、Oracle のオンラインドキュメント. を参照ください。以下にある属性のサブセットが現在生成されます: SQLName、SQLType、SQLSchema。.

メモ: SQL Server ならびに Oracle 拡張は、それぞれ対応するデータベース以外のデータベースに対して生成することができます。この機能は、別のデータベースにて作業しながら、将来 SQL Server または Oracle にて処理されるスキーマを生成するような場面で利用することができます。

プレビューペイン

(以下のスクリーンショットで示される)プレビューペインでは、選択範囲タブにて現在選択されているテーブルの構成が表示されます。選択範囲タブにて新しいテーブルが選択された場合、再ロードボタンによりプレビューが更新されます。



要素または属性としてフィールドをインポートするように指定することができ、また、列名の左横のシンボルをクリックして、インポートされないように指定することもできます。要素、属性、またはオプションの無視から選択することができます。上のスクリーンショットでは、`city` フィールド、が例えば、要素としてインポートされるようにセットされていますが、`street` フィールドは属性としてインポートされるように設定されています。

データ型の変換

データベースのデータ型からXML スキーマのデータ型への変換に関する情報については、付録[2081]を参照ください。

28.11.5 XML スキーマを元にデータベースをインポート

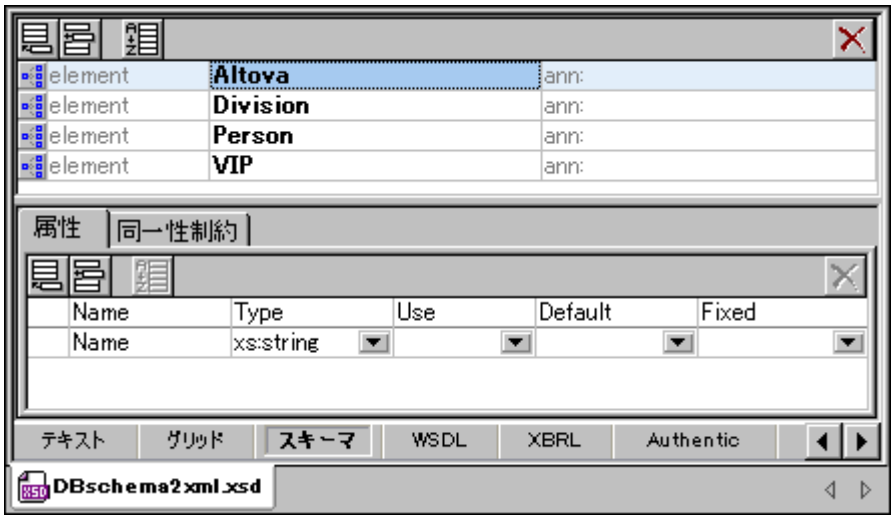
「XML スキーマを元にデータベースをインポート」コマンドにより、与えられた XML スキーマに従う形で妥当かつ、データベースからインポートされたデータを含む XML ドキュメントが作成されます。この機能は以下のデータベースにてサポートされます:

- Microsoft Access 2000 と2003
- Microsoft SQL Server
- Oracle
- MySQL
- Sybase
- IBM DB2

インポートされるデータはデータベースにて選択されているテーブルにより決定されます。スキーマビューにてアクティブになっているドキュメントが、インポートにて使用される XML スキーマとなり、データベースへの接続が行われます。インポートを行うテーブルを選択し、インポートをクリックします。データが XML ドキュメントへインポートされ、データのインポート時にアクティブだった XML スキーマに従う形でドキュメントは構成されます。

以下の例では、MS Access データベース上にあるデータが、スキーマビューでアクティブな XML スキーマに従ってにインポートされます。インポートを行うためのステップを以下に示します:

1. スキーマファイルをスキーマビューにて選択します(以下のスクリーンショットを参照)。



2. メニューコマンド「XML スキーマを元にデータベースをインポート」を選択します。データソースへ接続[848] ダイアログが表示されます。
3. Microsoft Access (ADO) オプションを選択し、次へをクリックします。
4. 参照をクリックし、データベースファイルを選択します。次へをクリックします。

5. XML スキーマに基づく DB インポートダイアログにて、テーブルタグへ移動し、インポートを行うテーブル(例えば Altova)を選択し、「インポート」をクリックします。テーブルが XML ドキュメントへインポートされ、グリッドビューにて表示されます。

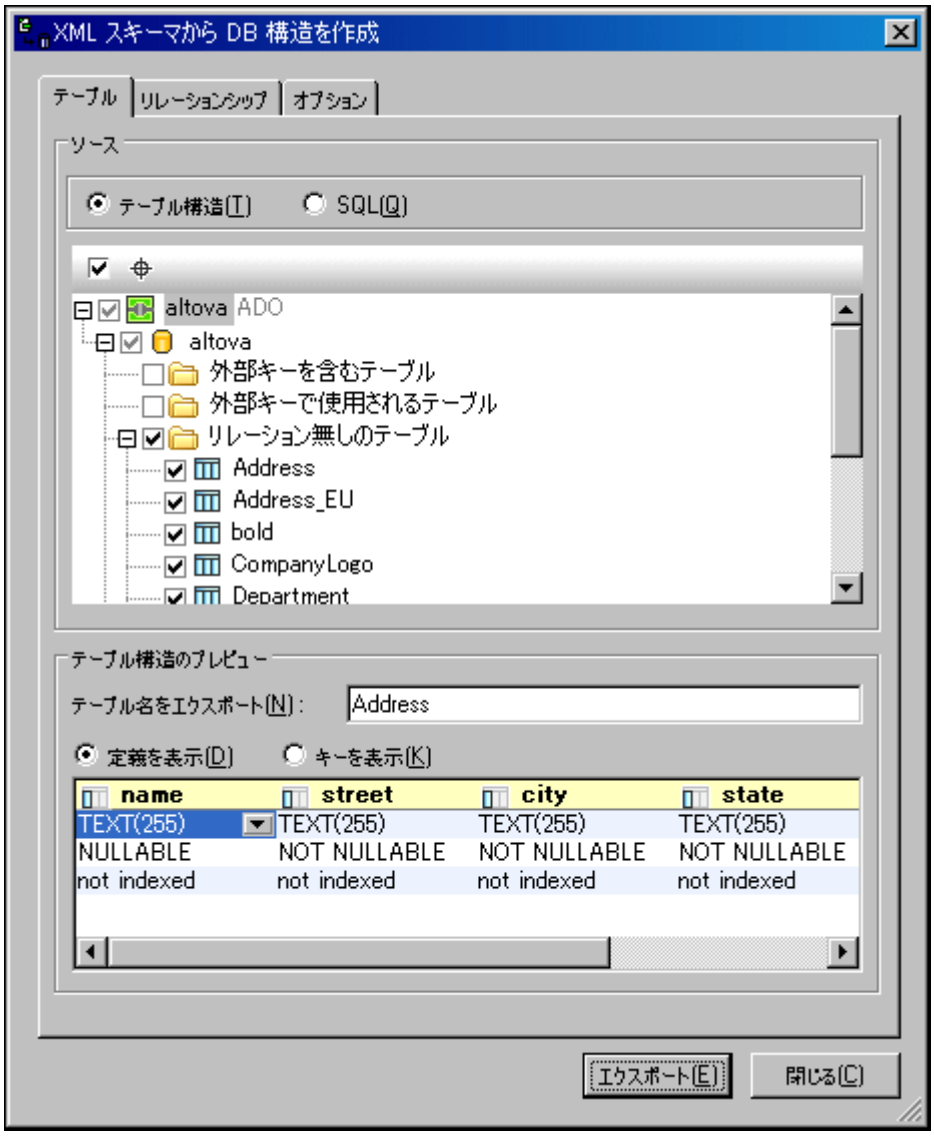
データ型の変換

データベースのデータ型からXML スキーマのデータ型への変換に関する情報については、付録[2081]を参照してください。

28.11.6 XML スキーマを元にデータベース構造を作成

XMLSpy では、既存のスキーマファイルを元に、空のデータベース(またはスケルトンデータベース)を作成することができます。以下に記される方法は、各種データベースに対して使用することができます。

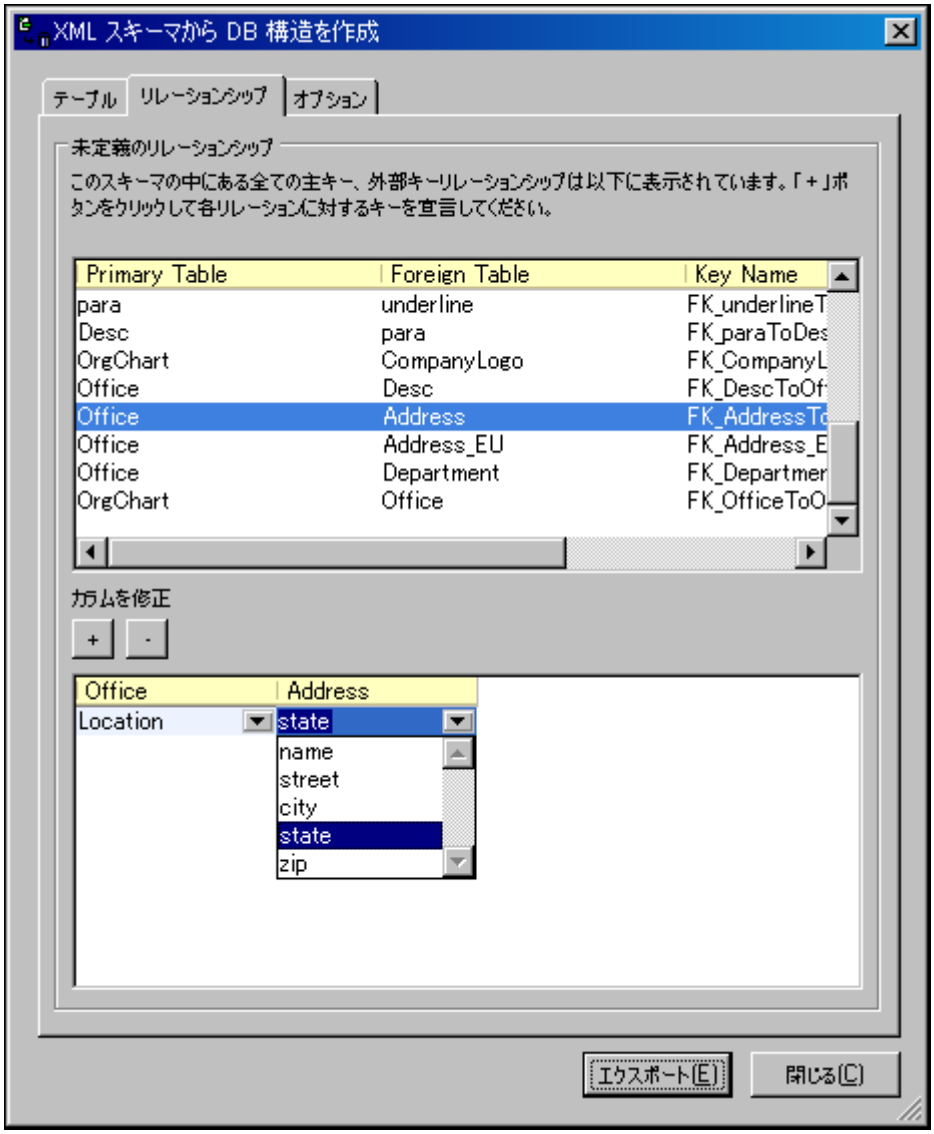
1. スキーマファイルをスキーマ/WSDL ビューにて開きます。
2. メニューコマンドから「**変換 | XML スキーマを元にデータベース構造を作成**」を選択します。データソースへの接続[848] ダイアログが表示され、データベース(DB) への接続を行います。
3. リレーショナルデータベースへの接続を行うには、データソースへの接続[847] のセクションに記述された方法を使用します。例えば、Microsoft Access データベースへ接続するには、Microsoft Access ラジオボタンを選択し、データベースを選択する操作を続けます。スキーマ構造の作成には、既存のデータ
4. XML スキーマから DB 構造を作成ダイアログにて、スキーマから作成されたテーブルがツリー構造で表示されます。例えば以下のスクリーンショットでは、Address テーブルが作成され、エクスポートされる対象として選択されています。エクスポートしないテーブルは、隣にあるチェックボックスやコンテキストメニューからチェックを外してください。



リレーションシップと共にDB テーブルを作成する

DB 構造の元となるXML スキーマにて同一性制約によるリレーションシップが定義されている場合、これらのリレーションシップが自動的に作成され、テーブル構造にて表示されます。外部キーを含むテーブルならびに外部キーで使用されるテーブル以下に、リレーションシップを含むテーブルが表示されます。

リレーションシップタブでは、テーブルリレーションシップを作成ならびに修正することができます。このタブでは、主キー/外部キーが取りうる全てのリレーションシップが表示されます(以下のスクリーンショットを参照)。



以下の操作により、リレーションシップを作成することができます:

1. 表示されている中から、1つの主キー/外部キーリレーションシップを選択してください。
2. ダイアログ下部にあるペインにてプラスボタンをクリックして、リレーションシップを作成します。
3. 2つのテーブルにあるドロップダウンリストから、目的のカラムを選択します。

リレーションシップを選択して、マイナスボタンをクリックすることで、リレーションシップを削除することができます。

データベース構造ならびに接続に関するメモ

同一性制約により定義されたスキーマ構造は、生成されるデータベースにも反映されます。XML スキーマから DB 構造を作成ダイアログにより作成されるデータベースの種類と制約、そして接続方法を以下のテーブルに記します。

	直接	ODBC を使用	ADO を使用

(C) 2015-2021 Altova GmbH

MS Access (2000 と2003)	OK *	OK	OK
MS SQL Server	OK *	OK	OK
Oracle	OK *	OK	OK
MySQL	-	OK *	OK ⌖
Sybase	-	OK *	OK
IBM DB2	-	OK *	OK

* *推奨される接続方法*
⌖ *MySQL: ODBC をベースにした ADO 接続を作成する場合、User またはシステム DSN の使用が推奨されます*
\- *未サポート*

XMLSpy では、階層構造ならびにフラット構造のスキーマ[1339] のマッピングが行われます。両者のフォーマットは、XMLSpy により自動的に認識されます。
フラット構造から SQL へのマッピングは2つの点で異なります:

- SQL Server DB、Oracle DB、または Sybase DB:
  上にあるデータベースに対してフラット構造で生成されたスキーマは、スキーマカタログ名が抽出され、生成された SQL スクリプトにて DB 名として使用されます。つまり、生成される SQL スクリプトを実行するには、ターゲット DB の名前をスキーマカタログ名と同じものにする必要があります。
- Access (2000 ならびに 2003)、MySQL、または DB2 DB:
  上にあるデータベースに対してフラット構造で生成されたスキーマは、SQL スクリプトが生成される時に、スキーマカタログ名を無視します。つまり、生成される SQL はログインしたターゲットデータベースに対して実行されます。

データ型の変換

データベースのデータ型から XML スキーマのデータ型への変換に関する情報については、付録[2081]を参照ください。

28.11.7 テキストファイルへエクスポート

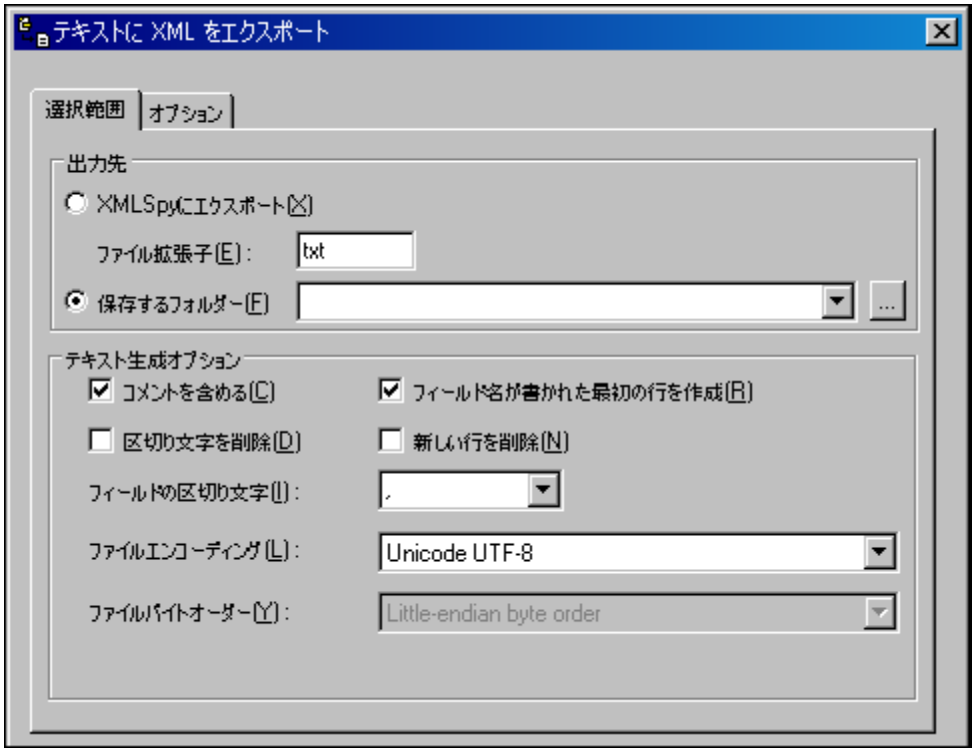
「**変換 | テキストファイルへエクスポート**」コマンドにより、XML データをテキスト形式でエクスポートすることで、データベースやレガシーなシステムと、データの受け渡しを行うことができます。ダイアログは2つのパートから構成されます(それぞれ別々にスクリーンショットを以下に示します)。

- 上部にある2つのタブ: (i) 選択範囲ならびに (ii) エクスポートオプション
- 下部にあるプレビューウィンドウ

ダイアログにて目的のオプションを選択した後、**エクスポート**ボタンをクリックしてテキストファイルへのエクスポートを行います。

選択範囲

選択範囲タブでは、エクスポートするファイルの出力先と、テキスト生成に関するオプションを指定することができます(以下のスクリーンショットを参照)。



出力先 : ファイルをフォルダーへ直接エクスポートします。ファイルの拡張子を指定することができます。ファイル名は、エクスポートする XML ファイル内にある要素の名前により決定されます。他にも、新規ファイルとして XMLSpy へエクスポートすることもできます。この場合ファイルは GUI にて表示され、後から保存することができます。.

コメントを含める : このオプションを有効にすると、データの取得に使用される SQL クエリや、データベーステーブル内にあるヘッダーカラムのリストといった、XML ファイル内にあるコメントがエクスポートされます。

フィールド名が書かれた最初の行を作成 : 有効になっていると、エクスポートされたテーブルにデータベースカラム名が含まれます。

区切り文字を削除 : エクスポートされたデータのテキスト値内にある区切り文字を削除します。コンボボックスから削除する区切り文字を指定してください。例えばこのオプションが有効になっており、アポストロフィーが指定されている場合、Ba'ker という XML 値をエクスポートすると、実際にエクスポートされる値は Baker となります。

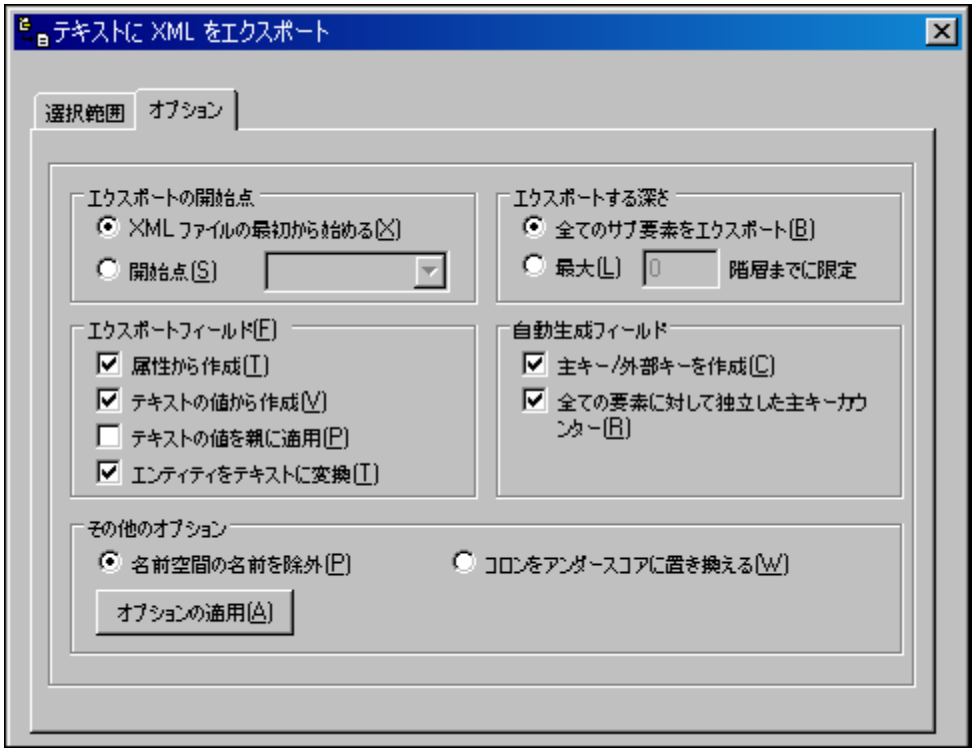
新しい行を削除 : エクスポートされるデータから新しい行を削除します。

文字列引用句 : このコンボボックスにて指定された引用句が、データの各文字列において使用されます。

バイトオーダー : 16ビットまたは32ビットユニコード(UCS-2、UTF-16、またはUCS-4)ファイルをエクスポートする場合、リトルエンディアンとビッグエンディアンからバイトオーダーを選択することができます。

エクスポートオプション

以下に記されるその他のエクスポートオプションは、エクスポートタブにて指定することができます(以下のスクリーンショットを参照)。



エクスポートの開始点: : XML ドキュメント全体を*エクスポート*するか、現在選択されている要素から始まるデータ構造の*み*を*エクスポート*するか選択することができます。*エクスポート*されるサブレベルの深さについては、*エクスポートする深さ*オプションにて指定することができます。

*エクスポートする深さ*開始点から見て、*エクスポート*されるサブレベルの数を指定します。

エクスポートフィールド: XML データによって、要素や属性、または要素内の*コンテンツ*だけを*エクスポート*することができます。プレビューウィンドウでは、個々の要素に対して*エクスポート*するかを選択することもできます。

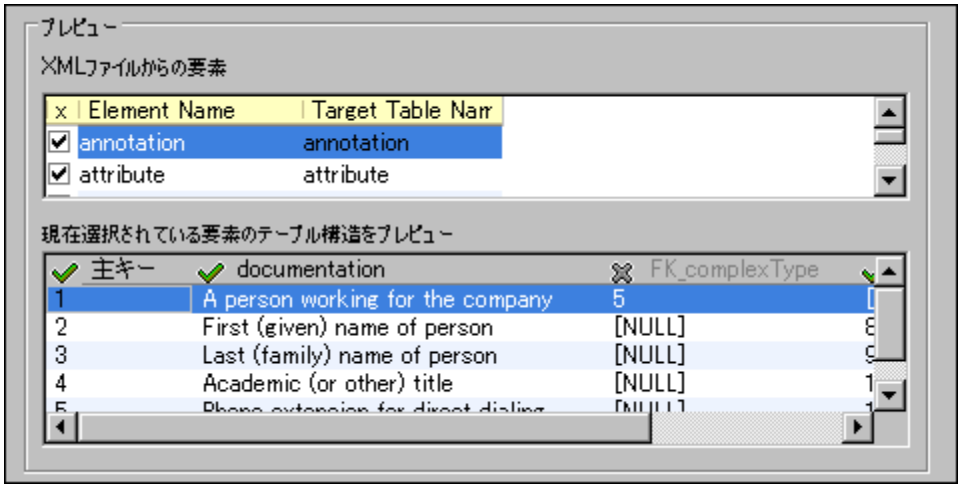
自動生成フィールド : XMLSpy では選択された各要素の型に対して1つの出力ファイルやテーブルが生成されます。主キーや外部キーのペアを自動的に作成することで、データをリレーショナルに*リンク*したり、各要素に対して主キーを定義することができます。

名前空間の名前を削除: *コロンをアンダースコアに置き換える*ラジオボタンと択一の*オプション*になります。要素や属性の名前空間プレフィックスを削除するか、名前空間プレフィックスのコロンを*アンダースコア*に置き換えるか選択することができます。

オプションの適用: *オプション*が設定されると、このボタンを*クリック*して、*オプション*を適用します。プレビューペイン内のプレビューは新しい*オプション*と共に更新されます。

プレビューウィンドウ

プレビューウィンドウは選択範囲ならびに*エクスポートオプション*タブ以下に表示されます。



XML ファイルからの要素ペインには、エクスポートされるノードの名前と、生成されるファイルにおける名前が表示されます。エクスポートされるノードを選択/選択解除することができます。要素が選択されていると、その構造が下部にあるペインにて表示されます。このペインにて、カラムの左をクリックすると、そのカラムをエクスポートするかどうか選択することができます。上のスクリーンショットでは、最後の列（FK_Office）がオフに切り替えられています。

28.11.8 データベースへエクスポート

「**変換 | データベースへエクスポート**」コマンドにより、XML データをデータベースへエクスポートします。このコマンドをクリックすることにより表示される接続ウィザードにより、更新を行うデータベースへの接続を行うことができます。接続が確立されると、データベースに XML データをエクスポートダイアログが表示されます。ダイアログは2つのパートから構成されます（それぞれ別々にスクリーンショットを以下に示します）。

- 上部にある2つのタブ: (i) 選択範囲ならびに (ii) エクスポートオプション
- 下部にあるプレビューウィンドウ

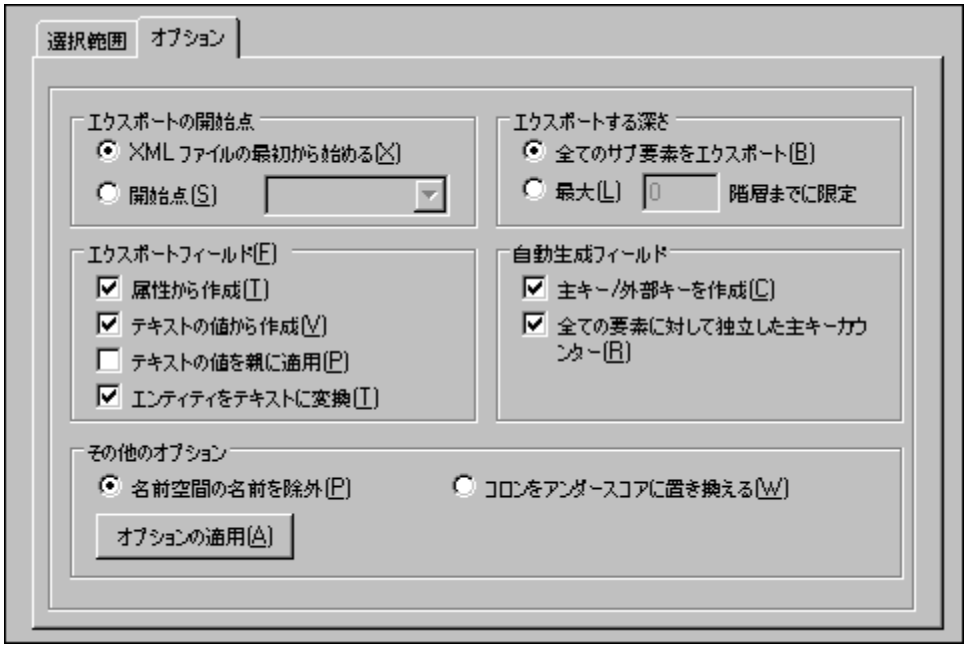
ダイアログにて目的のオプションを選択すると、**エクスポート**ボタンをクリックしてテキストファイルへのエクスポートを行います。

選択範囲

選択範囲タブでは、出力先のデータベースとテーブル生成オプションを選択することができます。出力先フィールドではデータベースへの接続を選択します。データを新たなテーブルとして作成するのか、既存のテーブルを更新するのか、または既存テーブルの更新を試み、それができない場合に新たなテーブルを作成するか選択することができます。エラーの回数による停止アクションセットしたり、SQL スクリプトのログを取るオプションが用意されています。

エクスポートオプション

以下に記されるエクスポートオプションは、エクスポートタブにて指定することができます（以下のスクリーンショットを参照）。



エクスポートの開始点: : XMLドキュメント全体をエクスポートするか、現在選択されている要素から始まるデータ構造のみをエクスポートするか選択することができます。エクスポートされるサブレベルの深さについては、*エクスポートする深さ*オプションにて指定することができます。

*エクスポートする深さ*開始点から見て、エクスポートされるサブレベルの数を指定します。

エクスポートフィールド: XMLデータによって、要素や属性、または要素内のコンテンツだけをエクスポートすることができます。プレビューウィンドウでは、個々の要素に対してエクスポートするかを選択することもできます。

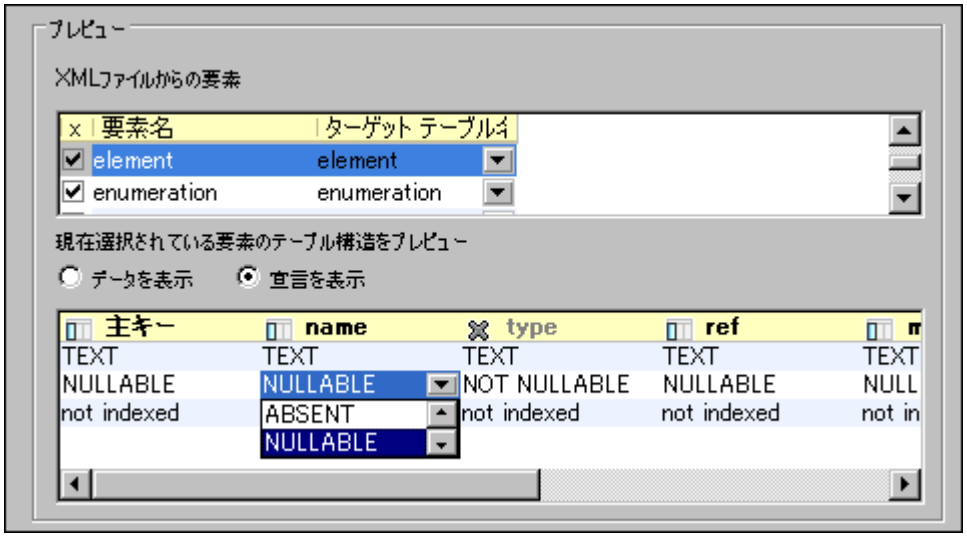
自動生成フィールド: XMLSpyでは選択された各要素の型に対して1つの出力ファイルやテーブルが生成されます。主キーや外部キーのペアを自動的に作成することで、データをリレーショナルにリンクしたり、各要素に対して主キーを定義することができます。

名前空間の名前を削除: コロンをアンダースコアに置き換えるラジオボタンと択一のオプションになります。要素や属性の名前空間プレフィックスを削除するか、名前空間プレフィックスのコロンをアンダースコアに置き換えるか選択することができます。

オプションの適用: オプションが設定されると、このボタンをクリックして、オプションを適用します。プレビューペイン内のプレビューは新しいオプションと共に更新されます。

プレビューウィンドウ

プレビューウィンドウでは選択範囲ならびにエクスポートオプションタブ以下に表示されます(以下のスクリーンショットを参照)。



XML ファイルからの要素ペインには、エクスポートされるノードの名前と、生成されるファイルにおける名前が表示されます。エクスポートされるノードを選択/選択解除することができます。要素が選択されていると、その構造が下部にあるペインにて表示されます。このペインにて、カラムの左をクリックすると、そのカラムをエクスポートするかどうか選択することができます。上のスクリーンショットでは、左から数えて3つめのカラムが無効になっており、エクスポートの対象から外されます。このプレビューは以下のプレビューの間を切り替えることができます: (i) 生成された構成内のデータ (*データの表示*); または (ii) 生成された構成内の各列の定義 (*定義の表示*)。上のスクリーンショットは、列の定義を表示しています。

2番目のペインでは、列名の左をクリックすると、列を以下の4つの設定でサイクルします: (i) テーブル構成内に含む (ii) 一意の制約 (iii) 主キーの制約 (iv) テーブル構成空除外 上のスクリーンショットでは、`Location` 列には一意の制約があり、`Fax` 列は、テーブル構成から除外されます。他の全ての列は、テーブル構成内に含まれています。

要素のテーブル構成フィールドの定義を表示する場合 (*定義の表示*)、定義は定義を選択して、定義のコンボボックスからオプションを選択することにより編集することができます (*上のスクリーンショット参照*)。

28.11.9 XML インスタンス を JSON に変換する (JSON から XML インスタンスに変換する)

アクティブなドキュメントがXMLドキュメントの場合、このコマンドは、JSONドキュメントを生成します。アクティブなドキュメントがJSONドキュメントの場合、コマンドは、XMLドキュメントを生成します。生成されたドキュメントは、新規のウインドウで開かれ、希望する場所で保存することができます。 変換オプションについては下で説明されています。に関する詳しい情報は、XMLSpy 内でのJSON とJSON編集へのサポートに関しては、以下のセクションを参照してください: JSON とJSON スキーマ 587。

変換のサンプル

ソースXMLドキュメントのサンプルは以下のとおりです。JSONドキュメントは、「**XML インスタンスを JSON に変換する**」(**JSON から XML インスタンスに変換する**) コマンドにより生成されます。

XMLドキュメント

```
<?xml version="1.0" encoding="UTF-8"?>
<Person first="Jim" last="James">
  <Address>
```

```
    <street>4 New Street</street>
    <city>New York</city>
    <state>NY</state>
    <code>10123</code>
  </Address>
  <Tel type="home">
      123 123-1234
  </Tel>
  <Tel type="office">
      123 987-9876
  </Tel>
</Person>
```

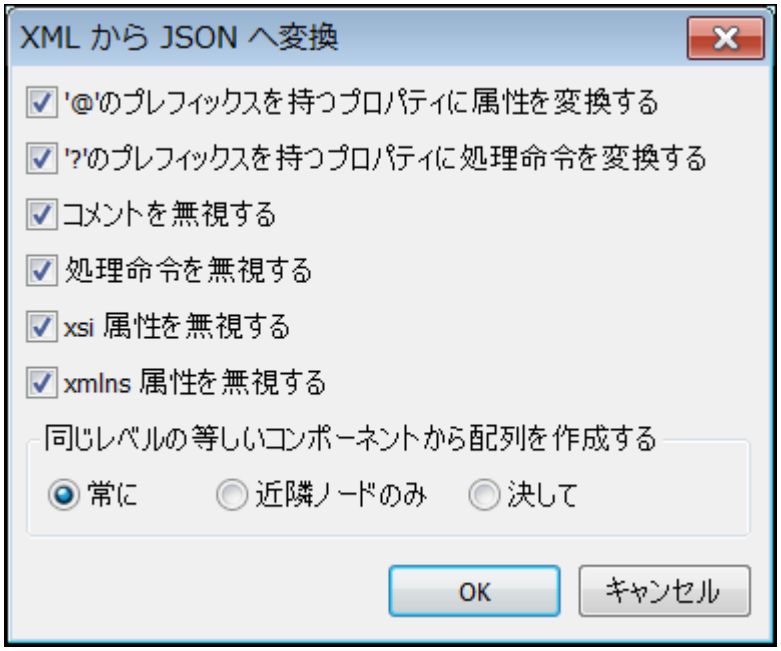
JSON ドキュメント

```
{
       "XML": {
              "version": 1.0,
              "encoding": "UTF-8"
       },
       "Person": {
              "first": "Jim",
              "last": "James",
              "Address": {
                     "street": "4 New Street",
                     "city": "New York",
                     "state": "NY",
                     "code": 10123
              },
              "Tel": [ { "type": "home",
                     "Text": "\r      123 123-1234\r  "}, { "type": "office",
                     "Text": "\r      123 987-9876\r  "} ]
       }
}
```

JSON ドキュメントを XML に変換するには、JSON ドキュメントをアクティブにし、「XML インスタンスを JSON に変換する」（JSON から XML インスタンスに変換する）コマンドをクリックします。

XML から JSON への変換オプション

「XML インスタンスを JSON に変換する」（JSON から XML インスタンスに変換する）コマンドをクリックすると、XML インスタンスドキュメントを JSON インスタンスドキュメントに変換する XML から JSON への変換 ダイアログ（*下のスクリーンショット*）が表示されます。変換オプションを設定し、「OK」をクリックします。JSON インスタンスドキュメントが XML インスタンスから生成され、生成された JSON ドキュメントは、新しいウィンドウでひらかれます。



最初の2つのオプションは、同じレベルでの要素の競合を回避するためにプレフィックスがJSON プロパティ名 に追加されるかを定義します。下の2つのリストが上記を説明しています。XML *属性* `somenode` は、JSON プロパティ `@somenode` に変換されています。このように、(XML 要素 `somenode` から作成された) JSON プロパティ `somenode` との競合が回避されます。

XML インスタンス

```
<root somenode="value">
    <somenode>content</somenode>
</root>
```

JSON instance

```
{
    "root": {
        "@somenode": "value",
        "somenode": "content"
    }
}
```

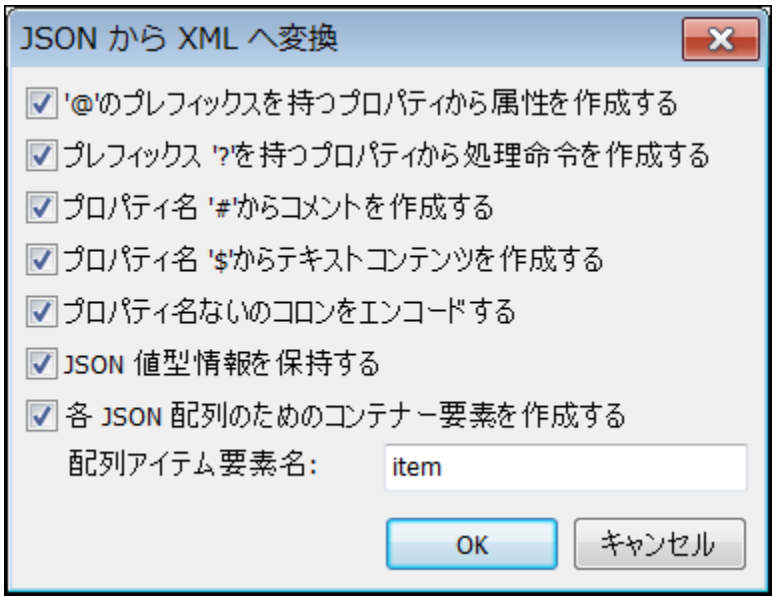
次のオプションにより、特定のXML ノードの型が変換されるかを指定することができます。XML コメントが含まれている場合は、`"#"` という名前が与えられます。(複合型コンテンツを持つ要素内で通常発生する) テキストノードは `"$"` という名前を与えられます。XML ノードが名前空間プレフィックスを持つ場合、対応するJSON の名前は、この名前空間プレフィックスを使用して作成されます。同じ名前を持つ要素が、同じレベルに存在する場合、等価 コンポーネントと考えられます。同様に、同じレベルに存在するノード、処理命令、とテキスト() は、等価 コンポーネントと考えられます。等価 コンポーネントが同じレベルに存在すると、配列を作成するか否かを選択することができます。すべての等価 コンポーネント、近隣する等価コンポーネントから配列を作成するか、または、配列を作成しないオプションを選択することができます。

XML スキーマ型情報の使用 オプションによりソースノードのXML スキーマ型をベースに行うことができます。例えば、ノードがスキーマ内で型 `xs:string` として定義されている場合、ターゲット JSON オブジェクトプロパティはJSON 型 `string` になり、引用符で囲まれます。これはとても役に立ちます。例えば、ソースノード内で文字列として保管されている数値のJSON への変換は型に従い行われます。.

(C) 2015-2021 Altova GmbH

JSON から XML への変換オプション

「XML インスタンスを JSON に変換する」（JSON から XML インスタンスに変換する）コマンドをクリックすると、JSON インスタンスドキュメントを XML インスタンスドキュメントに変換する「JSON から XML に変換」ダイアログが表示されます（下のスクリーンショット）。変換オプションを設定し、「OK」をクリックします。XML インスタンスドキュメントが JSON インスタンスから生成され、生成された XML ドキュメントは、新しいウィンドウで開かれます。



以下の点に注意してください:

- JSON オブジェクトプロパティは、XML 要素に変換されます。ダイアログ内の最初のオプションによりプロパティの型が作成されるかを選択することができます。
- *プロパティ名内のエンコードコロン:* 選択されると、JSON の名前内のコロンはエンコードされ、コロンとして作成されません。選択されていない場合、コロンはそのままで保持されます。
- *JSON 値の情報の保持:* 選択されると、プロパティの JSON 型情報は、対応する要素の属性-値ペアとして作成されます。
- *JSON 配列のためにコンテナー要素を作成する:* XML ドキュメント内のコンテナー要素が JSON 配列オブジェクトの名前として与えられます。JSON 配列のアイテムは、このコンテナー内の XML 要素として作成されます。*配列アイテム要素名* テキストボックス内で指定される名前がそれぞれに与えられます。

28.11.10 XML スキーマ を JSON スキーマに変換する（JSON スキーマから XML スキーマに変換する）

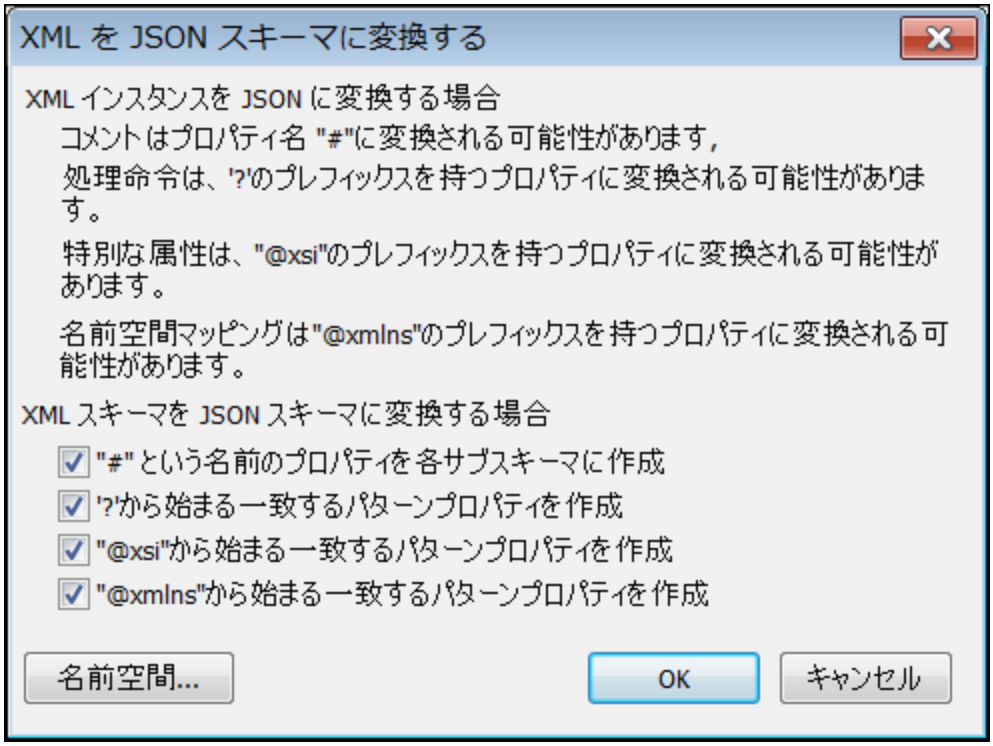
アクティブなドキュメントが XML スキーマの場合、このコマンドは、JSON スキーマドキュメントを XML スキーマから生成します。アクティブなドキュメントが JSON スキーマの場合、コマンドは、XML スキーマを JSON スキーマから生成します。生成されたドキュメントは、新規のウィンドウで開かれ、希望する場所に保存することができます。変換オプションについては下で説明されています。XMLSpy 内の JSON と JSON 編集のためのサポートに関する詳しい情報は、以下のセクションを参照してください: JSON と JSON スキーマ 587。

XML スキーマを JSON スキーマに変換するオプション

「XML スキーマを JSON スキーマに変換する」（JSON スキーマから XML スキーマに変換する）コマンドをクリックすると、XML スキーマドキュメントを JSON スキーマに変換する、「XML スキーマを JSON スキーマに変換する」ダイアログが表示されます（下のスクリーンショッ

ト)。必要とするJSON スキーマバージョン[631] と変更オプションを選択します。「OK」をクリックします。JSON スキーマは、XML スキーマから生成され、生成されたドキュメントは、新しいウィンドウでひらかれます。

一般的な変換の戦略方法は以下のとおりです: (i)(文字列番号などの) XML スキーマ単純型はJSON スキーマ単純型にマップされています(ii) XML スキーマ複雑型はJSON オブジェクトにマップされています。

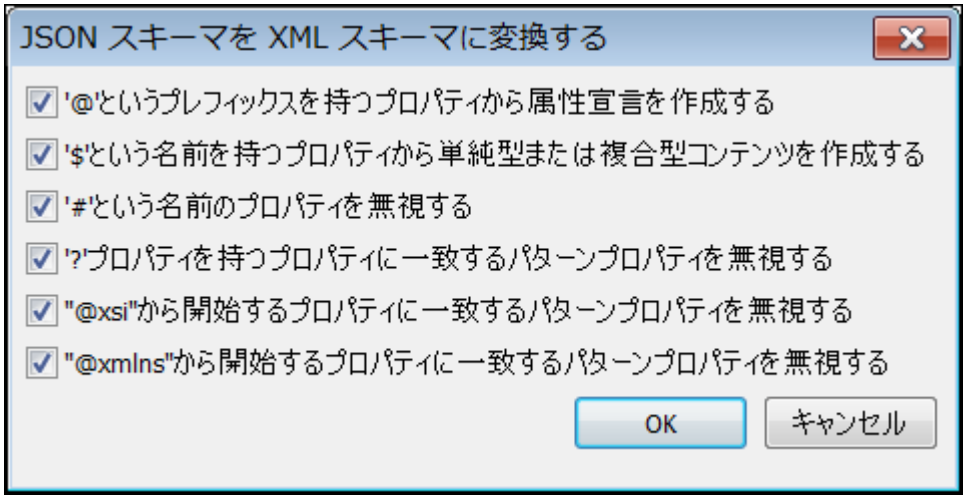


ダイアログの上の部分では、特定のXML スキーマ コンポーネントがどのように変換されるかに関する情報が提供されます。ダイアログの下の部分は、次のオプションを提供します:

- *各サブスキーマ内に"#"という名前のプロパティを作成する*: 選択されると、JSON スキーマ定義内にこの名前を持つプロパティが作成されます。
- *'?'、"@xsi"、"@xmlns"のプレフィックスを持つプロパティに一致するパターンプロパティを作成する*: それぞれのプレフィックスのために、これらのプレフィックスを持つプロパティに一致するパターンプロパティを指定します。パターンプロパティに関する詳しい情報は、以下のセクションを参照してください: JSON オブジェクトとプロパティ[640]。
- *maxOccurs > 1 を使用してパーティクルのための配列を常に作成する*: XML スキーマ内では、パーティクルは複合型コンテンツモデルの要素です。複数回発生する場合、パーティクルはJSON スキーマ内の配列として定義されます。それ以外の場合、JSON オブジェクトのプロパティとして定義されます。
- *ピュアオブジェクトを単純型コンテンツを持つ複合型のために作成する*: 単純型コンテンツを持つXML スキーマの複合型は属性とテキストコンテンツであり子要素ではありません。If the *ピュアオブジェクトの作成* オプションが選択されると、複合型はJSON オブジェクトに変換されます。型の属性はプロパティ名が@プレフィックスを持つJSON オブジェクトのプロパティに変換されます 。型のテキストコンテンツのために$ という名前のプロパティが生成されます。*ピュアオブジェクトの作成* オプションが選択されていない場合、複合型は他のオブジェクトを含むオブジェクトおよび文字列と番号などのJSON 単純型に変換されます。

JSON スキーマをXML スキーマに変換するオプション

「XML スキーマをJSON スキーマに変換する」(JSON スキーマからXML スキーマに変換する) コマンドをクリックすると、JSON スキーマドキュメントをXML スキーマに変換する、「JSON スキーマからXML スキーマに変換する」ダイアログが表示されます(*下のスクリーンショット*)。変換オプションを設定し、「OK」をクリックします。XML スキーマがJSON スキーマから生成され、生成されたドキュメントは、新しいウィンドウでひらかれます。

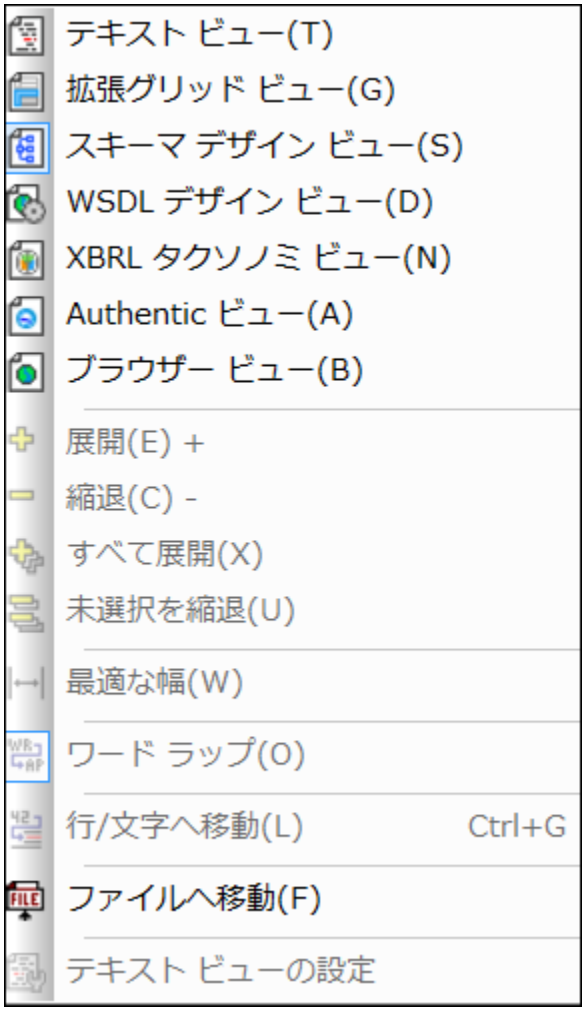


次のオプションから選択することができます:

- `'@'` と `'$'` から開始するJSON プロパティ名が作成されるかどうか。それぞれが属性ノードとテキストノードとして作成されます。
- `'#'` という名前のプロパティがXML コメントノードとして作成されるか。
- `'?'` プレフィックスを持つプロパティに一致するパターンプロパティで `"@xsi"` および/または `"@xmlns"` が無視されるか。無視されない場合、`'?'` プレフィックスを持つプロパティでは、`"@xsi"` と `"@xmlns"` がそれぞれ処理命令 `xsi:`プレフィックス属性 と `xmlns:`プレフィックス属性に変換されます。

28.12 表示メニュー

表示 メニューでは（*下のスクリーンショット*）では、アクティブな メインウィンドウ[109] の表示設定を行い、ドキュメントの表示方法を変更することができます。



このセクションでは、「**表示**」メニューから使用することのできるコマンドを説明します。

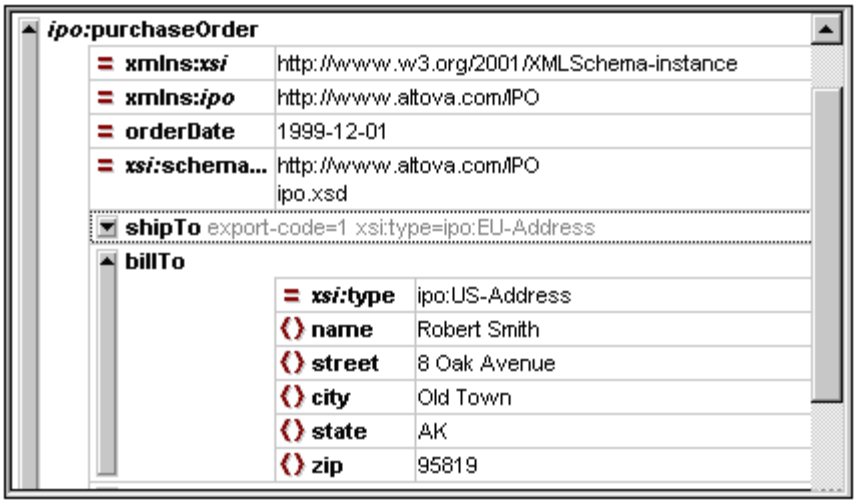
28.12.1 テキストビュー

このコマンドにより、現在使用されているドキュメントのビューを テキストビュー[130] へ切り替えます。テキストビューではテキスト形式でドキュメントの編集を行うことができ、数々のテキスト編集機能がサポートされます。テキストビューに関する詳細は、このドキュメントの テキストビュー[130] セクションを参照ください。

メモ: オプションダイアログ(「ツール| オプション[1454]」)にある様々なタブを使用することで、テキストビューの設定を変更することができます。

28.12.2 拡張グリッドビュー

このコマンドにより、現在のドキュメント表示がグリッドビュー[149] に切り替わります。それまで使用されていたビューがテキストビュー[130]の場合、ドキュメントの整形式が自動的にチェックされます。



28.12.3 スキーマデザインビュー

このコマンドにより、現在のドキュメント表示がスキーマデザインビューに切り替わります。ビューの切り替えは、現在選択されているドキュメントがXML スキーマドキュメントの場合だけ行うことができます。このビューに関する詳細は、このドキュメンテーションのスキーマビュー[159] セクションを参照ください。

28.12.4 WSDL デザインビュー

このコマンドにより、現在のドキュメント表示をWSDL デザインビューへ切り替えます。ビューの切り替えは、現在選択されているドキュメントが(.wsdl ファイル拡張子を持つ) WSDL ドキュメントの場合だけ行うことができます。このビューに関する詳細は、このドキュメンテーションのWSDL ビュー[231] セクションを参照ください。

28.12.5 XBRL タクソノミビュー

このコマンドにより、現在のドキュメント表示をXBRL タクソノミビューへ切り替えます。ビューの切り替えは、現在選択されているドキュメントが(.xsd ファイル拡張子を持つ) XBRL タクソノミドキュメントの場合だけ行うことができます。このビューに関する詳細は、このドキュメンテーションのXBRL ビュー[243] セクションを参照ください。

28.12.6 Authentic View

このコマンドは、現在のドキュメントをAuthentic View[540] に切り替えます。

Authentic View を使用するとAltova のStyleVision アプリケーションにより作成されたStyleVision Power Stylesheet テンプレートをベースにXML ドキュメントを編集することができます。これらのテンプレート (StyleVision スタイルシート または SPS ファイル) は、XML ドキュメントをグラフィカルな フォーマットで表示し、XML ドキュメントを(マークアップを使用下テキストフォーマット編集するよりも)簡単に編集することができます。

XML ドキュメントがSPS ファイル(Authentic | StyleVision スタイルシートの割り当て[1296]) と関連付けられている場合、XML ドキュメントをAuthentic View で閲覧することができます。Authentic View 内でSPS ファイルを新規の空のテンプレートとして以下の方法で開くことができます:

- 「ファイル | 新規作成」コマンドを選択して、StyleVision スタイルシートの選択 ボタンをクリックします。
- 「Authentic | 新規ドキュメント」コマンドを選択して、SPS ファイルを参照します。

詳細に関しては、Authentic View[526] とStyleVision のドキュメントを参照してください。

28.12.7 ブラウザービュー

このコマンドにより、現在のドキュメントビューをブラウザービュー[256] へ切り替えます。このビューではXML が利用可能なブラウザーを使用して、CSS やXSL スタイルシートにより得られた情報によるXML ドキュメントのレンダリングを行います。

オプションダイアログのファイルタブ[1455] にて保存時に検証を行うよう設定している場合、ブラウザビューへの切り替えが行われる際に、ドキュメントの検証が行われます。オプションダイアログは、「ツール | オプション」メニューコマンドにより開くことができます。詳細に関しては、このドキュメントのブラウザービュー[256] のセクションを参照してください。

28.12.8 展開

このコマンド（*ショートカット:* 数字パッド **'+'**）はグリッドビューにて使用することができます。このコマンドにより、選択されている要素が1段階展開されます。展開が行われると、その要素と子要素が全て選択された状態になります。この機能により、"+" キーを繰り返し押下することで、大きな要素の展開を行うことができます。

28.12.9 縮退

このコマンド（*ショートカット:* 数字パッド **'-'**）はグリッドビューにて使用することができます。このコマンドにより、選択されている要素が1段階縮退されます。
各要素の左に配置されているグレーのバーをクリックすることでも、要素の展開や縮退を行うことができます。

28.12.10 全て展開

このコマンド（*ショートカット:* * または数字パッド上で x）はグリッドビューにて使用することができます。テキストビューの縮退マージンがアクティブな場合テキストビューにて使用することもできます。選択された要素のすべての子孫ノードを展開します。

28.12.11 未選択を縮退

このコマンド（ショートカット: **Ctrl** + 数字パッド **'-'**）はグリッドビューにて使用することができます。選択されている要素の周りにある要素が縮退され、その要素とその子孫だけにフォーカスすることができます。目的の要素を選択して、このコマンドを選択することで、その他全ての（選択されていない）要素が縮退されます。

28.12.12 最適な幅

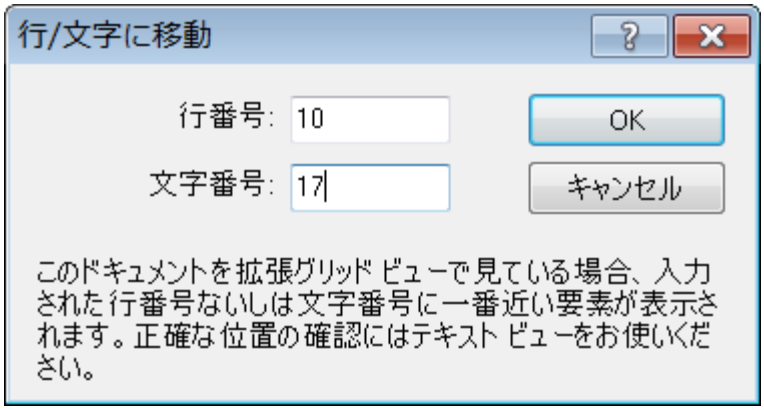
このコマンドにより、セルに含まれている最も長い文字列が1行に収まるように全カラムの幅が調整されます。オプションダイアログ(「ツール | オプション」)のビューセクションでは、最適幅の最大サイズを指定することができます。最適幅は、表示されているカラムのセルにより計算されているということに留意してください。この機能により、個々の要素が展開または縮退された時に、最適な表示を行うことができるようになります。

28.12.13 ワードラップ

このコマンドにより、「テキストビュー」におけるワードラップ(右端折り返し)を有効化/無効化することができます。ワードラップがオンに切り替えられると、テキストはウィンドウの端で折り返されます。

28.12.14 行/文字へ移動

このコマンド(*ショートカット*: **Ctrl+g**) テキストビューとグリッドビュー利用することができます。移動先の文字と数字を入力することのできるダイアログ(*下のスクリーンショット*)が表示されます。テキストビューでは、入力された位置にカーソルが移動します。グリッドビューでは、特定の行番号および文字の位置に最も近いノードがハイライトされます。



このコマンドは、テキストビューならびにグリッドビューの両方で使用することができますが、グリッドビューでは、指定された文字位置に最も近い要素をハイライトすることで、大まかな位置が示されます。

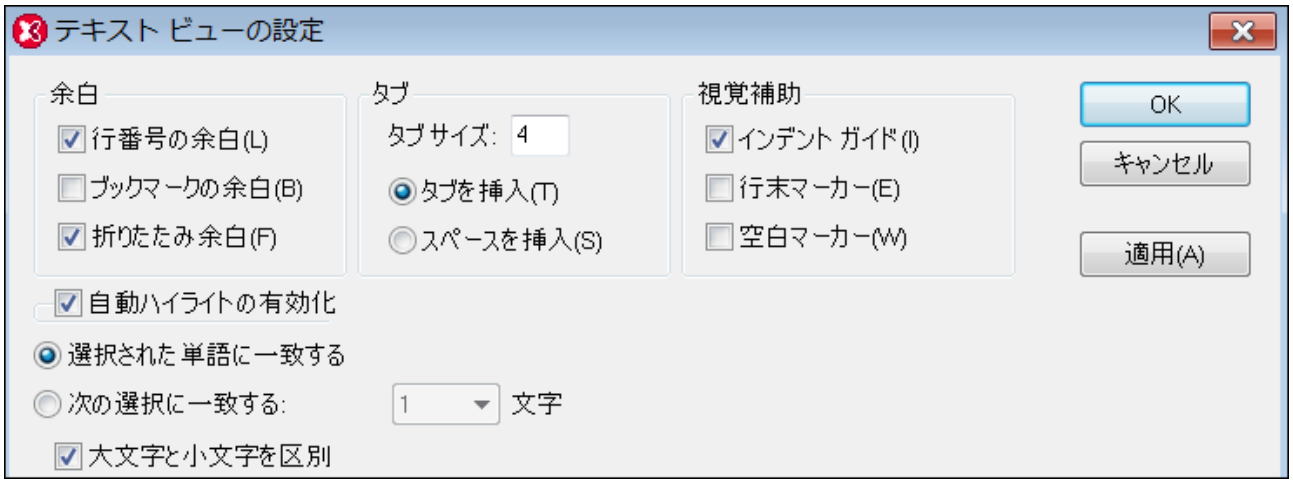
(C) 2015-2021 Altova GmbH

28.12.15 ファイルへ移動

このコマンドはテキストビューとグリッドビュー内で有効化することができます。（テキストビュー内で）カーソルがファイルを参照するテキスト内、または、（グリッドビュー内で）ファイルを参照するテキストを含むノード内にカーソルが置かれると、参照されるドキュメントが開かれます。現在編集中のファイル内から、参照されているドキュメントが開かれます。

28.12.16 テキストビュー設定

テキストビュー設定コマンドにより、テキストビュー設定ダイアログが表示され、テキストビューの設定を行うことができます（以下のスクリーンショットを参照）。テキストツールバーに配置されているショートカットアイコン を使用することでも、このコマンドを使用することができます。



余白

余白ペインでは、行番号、ブックマーク、そして折りたたみ余白を有効/無効にすることができます。これらはそれぞれ異なる種類の余白で、行番号、ブックマーク、ソースの折りたたみ（展開や縮退を行うためのアイコン）がそれぞれ表示されます。これら設定により、テキストビューにて余白が表示されるかが決定されます。ブックマークコマンドは編集メニュー内に配置されています。ノードの展開や縮退を行うには、折りたたみ余白が有効になっていなければなりません。

タブ

タブペインでは、タブの大きさを指定することができます。タブを挿入するかスペースを挿入するか決定するラジオボタンにより、インデントを使った整形がオプションダイアログのビュータブ（1466 ツール | オプション 1466）1466内で有効になっている際にタブを使用するのか、スペースによるインデントを行うかが決定されます。

視覚補助

視覚補助ペインには、インデントガイド（テキストのインデントを表示するための縦方向に走る破線）、改行マーカー、そして（下のスクリーンショットでピンク色で表示されている）空白マーカーを表示するための設定が含まれます。タブは矢印にて、空白スペースは点線で示されています。空白スペースマーカーは、オプションダイアログのテキストビューオプション[1470]内でカスタム化することができます。

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" EOL
    xmlns:ts="http://www.xmlspy.com/schemas/textstate" EOL
    elementFormDefault="qualified" attributeFormDefault="unqualified"> EOL
  <xs:import namespace="http://www.xmlspy.com/schemas/textstate" schemaLocation="TextState.xsd"/> EO
  <xs:element name="expense-report"> EOL
    <xs:annotation> EOL
      <xs:documentation>expense-report is the root element.</xs:documentation> EOL
    </xs:annotation> EOL
```

ハイライトの有効化

ハイライトが有効化されていると、テキストビュー内の選択の発生する箇所全てがハイライトされます。このペイン内のオプションにより選択項目が構成されます。選択は、大文字と小文字を区別しつつ用語、または、文字数に定義することができます。文字の選択の場合は、一致する文字の最小数を指定することができます（例えば、2つまたは複数の文字）。テキストビュー内では、選択に一致する文字のシーケンスの発生すべてがハイライトされます。用語検索の場合、引用符無しの要素名、属性名、属性値、と要素タグの角ブラケットが区切られている文字と考えられます。

キーマップ

キーマップには、XMLSpy のショートカットと関連するコマンドのリストが表示されます。

(C) 2015-2021 Altova GmbH

28.13 ブラウザーメニュー

「ブラウザー」メニューのコマンドは、ブラウザービュー[256] で有効になります。「戻る」と「進む」コマンドはしかしながら、前回使用されたコマンドに移動することのできるスキーマビューでも有効化されることができます。

28.13.1 戻る

戻るコマンド (*ショートカット:* Alt + 左矢印) は、ブラウザービューとスキーマビューで有効化されます。

ブラウザービューにて、「戻る」コマンドにより、その前に表示されたページが表示されます。バックスペースキーを押下することでも同様の操作を行えます。戻るコマンドは、XML ドキュメントのリンクをクリックした後に、再度 XML ドキュメントへ戻るために使用することができます。

スキーマビューでは、戻るコマンドにより、その前に閲覧していたコンポーネントが表示されます。Alt + 左矢印がショートカットキーになります。戻るコマンドにより、最大500回前の閲覧箇所まで戻ることができます。

28.13.2 進む

進むコマンド (*ショートカット:* Alt + 右矢印) は、ブラウザービュー内で有効化されます。「進む」コマンドは、「戻る」コマンドを使用した後に有効になります。このコマンドにより、(i) ブラウザビューにて既に閲覧したページや、(ii) スキーマビューにて既に閲覧したスキーマコンポーネントを閲覧することができます。

28.13.3 中止

中止コマンドにより、ブラウザービューで行われているドキュメントの読み取りを中止します。スピードが遅いインターネット回線などにより外部ファイルや画像などがダウンロードされており、その処理を停止したい場合にこのコマンドを使用することができます。

28.13.4 最新の状態に更新

最新の状態に更新 (F5) コマンドにより、ブラウザービューのドキュメントや、CSS、XSL スタイルシート、DTD といった関連するドキュメントが再ロードされます。

28.13.5 フォント

フォントコマンドにより、XML ドキュメント内にあるテキストのレンダリングに使用されるデフォルトのフォントサイズを選択することができます。殆どのブラウザーにおける文字のサイズコマンドに似た動作をします。

28.13.6 別のウィンドウ

別のウィンドウで開くコマンドにより、ブラウザービューが別のウィンドウにて開かれ、他のビューと隣合わせで表示することができるようになります。ブラウ別のウィンドウでは、ブラウザービューはドキュメントの編集ビューと左右に表示することができます。

ブラウザービューが別のウィンドウで表示されていると、編集ビューにて **F5** を押下することで、対応するブラウザービューの内容も更新されます。ウィンドウを元のインターフェースに統合するには、アクティブなウィンドウの右上にある最大化ボタンをクリックしてください。

(C) 2015-2021 Altova GmbH

28.14 WSDL メニュー

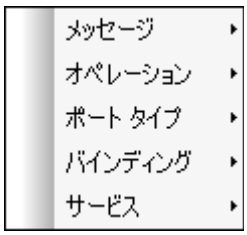
WSDL ドキュメントの作成や編集を視覚的に行うことができる WSDL ビュー[231] にて WSDL ドキュメントを閲覧している場合、**WSDL** メニューを使用することができるようになります。

WSDL ビューに関する記述は、WSDL ビュー[231] セクションを参照ください。WSDL を始めるにあたっては、WSDL チュートリアル[684] を参照ください。

関連項目：WSDL ドキュメントの処理方法については、WSDL ビュー[231] ならびに WSDL チュートリアル[684] のセクションを参照ください。

28.14.1 WSDL 1.1 コンポーネント

WSDL 1.1 コンポーネント メニューアイテムへマウスオーバーすることで、以下に示されるサブメニューが表示され、WSDL 1.1 コンポーネントの編集を行うために必要な各種コマンドを選択することができます。



WSDL 1.1 メニューの各アイテムには、それぞれ独自のサブメニューが収められており、そのコンポーネントに関するコマンドを選択することができます。これらサブメニューにある各コマンドについては、このセクション以下にあるサブセクションを参照ください。

28.14.1.1 メッセージ

メッセージの挿入

WSDL ドキュメントへ新たなメッセージを追加します。概要入力ヘルパーにてメッセージアイテムが開かれ、新たに作成されたメッセージがハイライトされます。

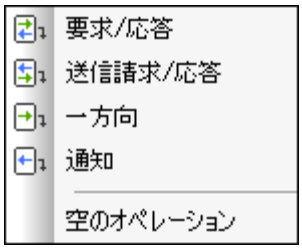
メッセージの削除
入力または出力要素から選択されたメッセージが削除されます。

メッセージパート（パラメーター）の追加
メッセージパート（パラメーター）を選択されたメッセージへ追加します。

メッセージパート（パラメーター）の削除
選択されたメッセージからメッセージパート（パラメーター）を削除します。

28.14.1.2 オペレーション

オペレーションの追加
新たなオペレーションを、オペレーションウィンドウへ追加します。追加するオペレーションの種類は、**オペレーションの追加**メニューコマンド以下のサブメニューから選択することができます。



オペレーションの削除
選択されているオペレーションをオペレーションウィンドウから削除します。

input 要素の追加
選択されているオペレーションへinput 要素を追加します。

output 要素の追加
選択されているオペレーションへoutput 要素を追加します。

fault 要素の追加
選択されているオペレーションへfault 要素を追加します。

(C) 2015-2021 Altova GmbH

input/output/fault 要素の削除
選択されている input、output、または fault 要素を削除します。

intput/output/fault 要素に新しいメッセージを追加
新たな(デフォルト)メッセージを、現在選択されている **input** または **output** メッセージに追加します。

28.14.1.3 ポートタイプ

PortType の挿入
ポートタイプウィンドウへ新たなポートタイプを追加します。

PortType の削除
選択されているポートタイプをポートタイプウィンドウから削除します。

28.14.1.4 バインディング

バインディングの挿入
新たなバインディングをバインディングウィンドウへ追加します。

バインディングの削除
選択されたバインディングをバインディングウィンドウから削除します。

子の追加
新たな拡張要素を input または output メッセージに追加することができます。メニューアイテムが有効になっていない場合、その場所へは追加できないことを表しています。拡張アイテムに関する詳細は、W 3C のW SDL 仕様を参照ください。

以下の拡張アイテムが利用可能です:

- soap:body
- soap:header
- soap:headerfault
- soap:fault
- mime:content
- mime:multipartrelated
- mime:part
- mime:mimeXml
- http:urlencoded
- http:urlreplacement

拡張要素を削除
選択された拡張アイテムを削除します。

28.14.1.5 サービス

サービスの挿入
新たなサービスをサービスウィンドウへ追加します。

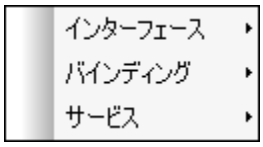
サービスの削除
選択されたサービスをサービスウィンドウから削除します。

ポートの挿入
サービスウィンドウにて選択されたサービスへ、新たなポートを追加します。

ポートの削除
選択されたポートを現在選択されているサービスから削除します。

28.14.2 WSDL 2.0 コンポーネント

WSDL 2.0 コンポーネント メニューアイテムへマウスオーバーすることで、以下に示されるサブメニューが表示され、WSDL 2.0 コンポーネントの編集を行うために必要な各種コマンドを選択することができます。



WSDL 2.0 メニューの各アイテムには、それぞれ独自のサブメニューが収められており、そのコンポーネントに関するコマンドを選択することができます。これらサブメニューにある各コマンドについては、このセクション以下にあるサブセクションを参照ください。

(C) 2015-2021 Altova GmbH

28.14.2.1 インターフェイス

以下のコマンドをインターフェイスメニューにて使用することができます(以下のスクリーンショットを参照)。

新たなインターフェイスの追加

メインウィンドウのインターフェイスカラムに、新たなインターフェイスボックスを追加します。インターフェイスボックスに表示されるデフォルトの名前がハイライトされ、そのまま新たな名前を入力することができるようになります。

インターフェイスの削除

選択されたインターフェイスを削除します。

fault 要素の追加

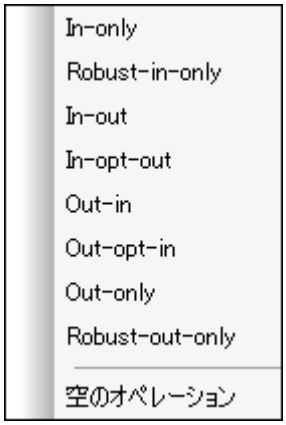
新たな `fault` 新たな fault 要素を選択されたインターフェイスへ追加します。インターフェイスボックスに表示される `fault` のデフォルト名がハイライトされ、そのまま新たな名前を入力することができるようになります。

fault 要素の削除

`fault` を削除します。

新規のオペレーションの追加

新たな `operation` 要素を選択されたインターフェイスへ追加します。追加するオペレーションの種類は、以下のスクリーンショットに示されるサブメニューから選択することができます。



インターフェイスボックスに表示される operation のデフォルト名がハイライトされ、そのまま新たな名前を入力することができるようになります。

オペレーションの削除

選択されたオペレーションを削除します。

28.14.2.2 バインディング

以下のコマンドを **バインディング** メニューにて使用することができます（以下のスクリーンショットを参照）。

新たなバインディングの追加

メインウィンドウのバインディングカラムへ、新たなバインディングボックスを追加します。バインディングボックスに表示されるデフォルトの名前がハイライトされ、そのまま新たな名前を入力することができるようになります。

バインディングの削除

選択されたバインディングを削除します。

(C) 2015-2021 Altova GmbH

fault 要素の追加

新たな `fault` 要素を選択されたバインディングへ追加します。バインディングカラムに表示される `fault` 要素には、インターフェースで宣言されている `fault` を参照する `ref` 属性が含まれます。バインディングにて新たに作成された `fault` では、参照を行うインターフェースの `fault` を、新たに作成された `fault` のコンボボックスから選択することができます。

fault の削除

選択された `fault` を削除します。

新規のオペレーションの追加

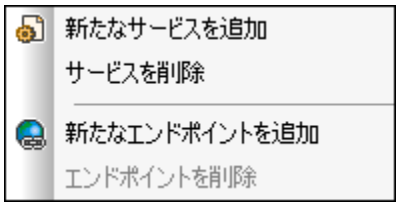
新たな operation 要素を選択されたバインディングへ追加します。バインディングカラムに表示される `operation` 要素には、インターフェースで宣言されている `operation` を参照する `ref` 要素が含まれます。バインディングにて新たに作成された `operation` では、参照を行うインターフェースの `operation` を、新たに作成された `operation` のコンボボックスから選択することができます。

オペレーションの削除

選択されたオペレーションを削除します。

28.14.2.3 サービス

以下のコマンドを **サービス** メニューにて使用することができます（以下のスクリーンショットを参照）。



新たなサービスの追加

メインウィンドウのバインディングカラムへ、新たなサービスボックスを追加します。サービスボックスに表示されるデフォルトの名前がハイライトされ、そのまま新たな名前を入力することができるようになります。インターフェースへの参照は、*Interface* プロパティのコンボボックスから選択することができます。

サービスの削除

選択されたサービスを削除します。

新たなエンドポイントの追加

新たな `endpoint` 要素を選択されたサービスへ追加します。エンドポイントに表示されるデフォルトの名前がハイライトされ、そのまま新たな名前を入力することができるようになります。インターフェースへの参照は、*Binding* プロパティのコンボボックスから選択することができます。エンドポイントのアドレスを、*Address* プロパティに入力する必要があります。property

エンドポイントの削除

選択されたエンドポイントを削除します。

28.14.3 型、ならびに図の保存

型 メニューアイテムには、次のコマンドが搭載されたサブメニューを使用することができます。これらについては下で説明されています。

- **新規のスキーマ**
- **埋め込まれたスキーマ**
- **スキーマの抽出**
- **スキーマビュー内のスキーマの編集**

ダイアグラムの保存 コマンドは、デザインダイアグラムを PNG ファイルとして保存します。

型 | 新規のスキーマ

このオプションは、W SDL ファイルにスキーマ要素が含まれていない時に有効になります。
「ファイル | 新規作成」メニューオプションを選択すると、スキーマ要素がスケルトン W SDL ファイルに追加されることに留意してください。従って（そのままの状態では）このメニューアイテムを選択することはできません（以下を参照）。

```
<types>
            <xs:schema/>
</types>
```

型 | 埋め込まれたスキーマ

コマンドは、埋め込むスキーマファイルを参照することのできるファイルを開くダイアログをポップアップします。ダイアログ内で「OK」をクリックすると、スキーマは、`types` 要素内でインラインスキーマとして作成されます。選択されたスキーマが既にインポートされている場合、既にインポートされたスキーマを埋め込むかをプロンプトされます。インポートされたスキーマの埋め込みを選択すると、`types` 要素内のインラインスキーマに変換されます。

型 | スキーマの抽出

このコマンドを選択すると、（`types` 要素内でインラインとして定義されている）埋め込まれたスキーマのそれぞれが、スキーマビュー内の一時ファイルとして開かれ、各ファイルのために名前を付けて保存 ダイアログがポップアップします。スキーマファイルを保存するには、スキーマが W SDL ファイルから抽出され、指定される場所に保存され、W SDL ファイルにインポートされます。埋め込まれたスキーマではなく、外部のインポートされたスキーマとなります。

スキーマビューでスキーマを編集

特定のスキーマに対する参照が W SDL ファイルに含まれない場合、スケルトンスキーマファイルが開かれます。「ファイル | 新規作成」メニューオプションを使用した場合に、この処理が行われます。特定のスキーマに対する参照が存在する場合、スキーマビューにてスキーマが開かれます。

28.14.4 ドキュメンテーションの生成

「WSDL | ドキュメンテーションの生成...」コマンドにより、現在編集しているWSDL ファイルの詳細なドキュメンテーションを生成することができます。ドキュメンテーションの出力は、HTML、RTF、またはPDF 形式で行うことができます。このコマンドにより生成されたドキュメンテーションは、Altova から許可を得ることなく、自由に改変、使用することができます。ドキュメンテーションの生成は、(ドキュメンテーションの生成コマンドを選択すると表示される)WSDL ドキュメンテーションダイアログにて選択されたコンポーネントに対して行われます。関連するコンポーネントは画面上の出力にてハイパーリンク化され、コンポーネント間のナビゲーションを簡単に行うことができるようになります。WSDL ドキュメンテーションは、インポートされたWSDL ファイルならびにXML スキーマファイル対しても生成することができます。ドキュメンテーションの生成に関する様々なオプションについては、ドキュメンテーション オプション[1375].のセクションを参照ください。

メモ: MS Word 形式でドキュメンテーションの生成を行うには、(バージョン 2000 以降の)MS Word がインストールされていなければなりません。

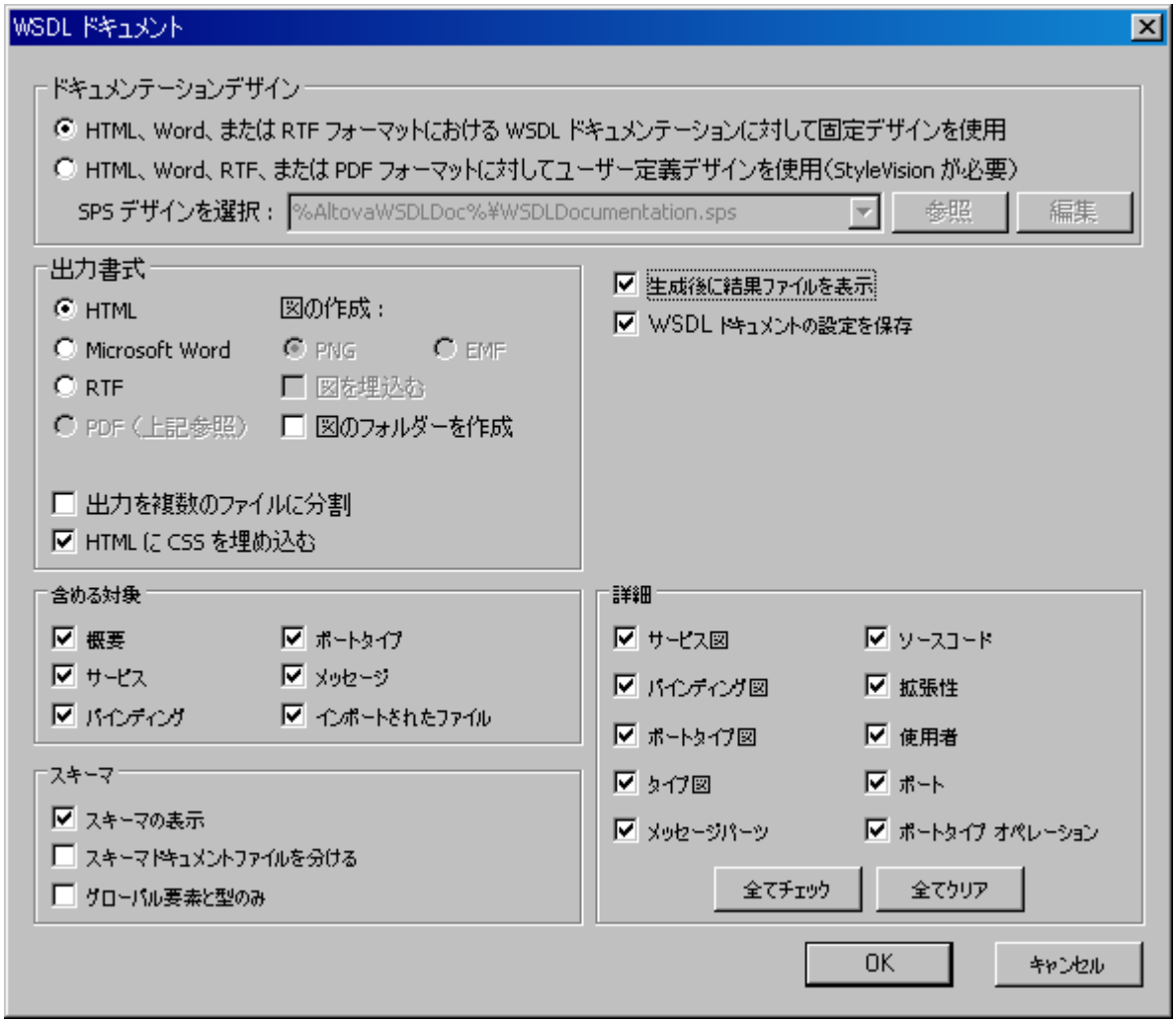
ドキュメンテーションの生成には、XMLSpy の固定フォーマット、またはStyleVision SPS のデザインを使用することができます。StyleVision SPS を使用することで、生成されたドキュメンテーションのデザインをカスタマイズすることができます。ドキュメンテーションのカスタマイズ方法については、ユーザーにより定義されたデザイン[1377]のセクションを参照ください。

メモ: SPS による WSDL ドキュメンテーションの生成には、お使いのコンピューターに StyleVision をインストールする必要があります。

28.14.4.1 ドキュメンテーションのオプション

「WSDL | ドキュメンテーションの生成」コマンドにより、WSDL ドキュメンテーションダイアログが表示され、ドキュメンテーションに関するオプションを選択することができます(以下のスクリーンショットを参照)。

ダイアログのドキュメンテーションデザインペインでは、ドキュメンテーションの生成にXMLSpy の固定デザインを使用するのか、StyleVision SPS により作成されたカスタマイズデザインを使用するのか選択することができます。使用するオプションを選択してください。PDF 出力は、StyleVision SPS により生成されたドキュメンテーションでしか利用することはできず、固定デザインを使ったドキュメンテーションの生成では利用できないということに注意してください。ユーザー定義デザインの使用方法については、ユーザー定義デザイン[1377]のセクションを参照ください。



W SDL ダイアログにあるその他のオプションに関する説明を以下に記述します。アクティブなドキュメントがW SDL 1.1 かW SDL 2.0 で、ダイアログ内にある*含める対象* ならびに*詳細* ペインの表示内容が変化します。上に示されるスクリーンショットは、W SDL 1.1 ドキュメントに対するドキュメンテーションダイアログを示しています。

- 目的のフォーマットを出力形式ペインにて選択します。HTML、Microsoft W ord、RTF、またはPDF から選択を行うことができます (PDF 出力フォーマットは、StyleVision SPS を使ったドキュメンテーションの生成でのみ選択可能になります)。OK をクリックすると、出力ファイルが保存される場所と名前を指定するよう促されます。
- Microsoft W ord ドキュメントは、固定デザインを使用した際には.doc 拡張子を持つファイルが生成され、StyleVision SPS を使用した生成では、 .docx ファイル拡張子が使用されます。
- ドキュメンテーションの生成は、単一のファイルに対して行える他、複数のファイルに分割して行うこともできます。複数のファイルが生成された場合、各ファイルが個々のコンポーネントに対応します。出力に含まれるコンポーネントは、含める対象ペインにあるチェックボックスにより指定することができます。固定デザインの場合、複数ドキュメント間を結ぶリンクが自動的に作成されます。
- HTML 出力の場合、CSS スタイル定義を、独立したCSS ファイルに収めるか、HTML ファイル(の<head> 要素)内部に収めるか指定することができます。独立したCSS ファイルが作成される場合、HTML ファイルと同じ名前で、 .css 拡張子を持つファイルが作成されます。必要に応じて *HTML に CSS を埋め込む* オプションにチェックを入れてください。
- Microsoft W ord、RTF、PDF オプションでは、*図を埋め込む* オプションが有効になります。このオプションが有効になっている場合、PNG 形式でダイアグラムが結果ファイル内部に埋め込まれます。オプションが無効になっている場合、ダイアグラムはPNG ファイルとして作成され、オブジェクトリンクにより結果ファイルにて表示されます。

(C) 2015-2021 Altova GmbH

- 出力ファイルがHTML の場合、全てのダイアグラムはドキュメントから独立したPNG ファイルとして作成されます。*図のフォルダーを作成* チェックボックスにチェックを入れることで、HTML ファイルと同じ場所にフォルダーが作成され、その中にPNG ファイルが保存されます。このフォルダーはHTMLファイル名_diagrams という名前になります。*図のフォルダーを作成* チェックボックスにチェックが入っていない場合、HTML ファイルと同じ場所にPNG ファイルが保存されます。
- 含める対象ペインでは、ドキュメンテーションに含めるアイテムを選択することができます。概要オプションを選択することで、全てのコンポーネントがコンポーネントの種類ごとにファイルの上部に表示されます。インポートされたファイル（WSDL 1.1） またはインポート/インクルードされたファイル（WSDL 2.0）オプションにチェックが入れられた場合、インポートされたファイル（WSDL 2.0 の場合は更にインクルードされたファイル）がスキーマのドキュメンテーションに加えられます。
- スキーマペインでは、ファイル内部にあるスキーマもドキュメンテーションに含まれるか選択することができます。スキーマを含めるよう選択した場合、(i) スキーマのドキュメンテーションが独立したファイルとして生成されるのか、またはメインドキュメンテーションファイル内部に加えるのか、(ii) スキーマ全体をドキュメンテーションの生成対象とするのか、またはグローバル要素、単純型、そして複合型だけを含めるのか選択することができます。
- 詳細ペインでは各コンポーネントに含めることのできるアイテムの詳細が表示されます。ドキュメンテーションに含めるアイテムを選択してください。**全てチェック**ならびに**全てクリア**ボタンにより、ペインにある全てのオプションを全て選択、または選択解除することができます。
- 生成後にファイルを表示 オプションは、全ての出力オプションに対して有効になります。このオプションが有効になっていると、生成されたファイルが（HTML 出力の場合は）ブラウザービュー、（MS W ord 出力の場合は）MS W ord、（RTF ならびにPDF 出力の場合は） .rtf ならびに.pdf ファイルのデフォルトアプリケーションにて表示されます。

パラメーター値

StyleVision SPS に複数のパラメーター定義が含まれる場合、**OK** をクリックした後にSPS にて定義されている全てのパラメーターがダイアログにて表示されます。このダイアログでパラメーター値を入力することで、SPS に割り当てられたデフォルトのパラメーター値をオーバーライドすることができます。

28.14.4.2 ユーザーにより定義されたデザイン

標準で使用されるXMLSpy の固定デザインの代わりに、カスタマイズされたデザインをXBRL タクソノミドキュメンテーションの生成に使用することができます。カスタマイズされたデザインは、出力ドキュメントのデザインテンプレートとなるStyleVision SPS により作成されます。

SPS の作成

StyleVision Power Stylesheet（またはSPS） は、Altova のStyleVision 製品を使用して作成されます。W SDL ドキュメントを生成するためのSPS は、W SDL ドキュメントの構造を指定するXML スキーマをベースにする必要があります。W SDL 1.1 とW SDL 2.0 のための2 つのスキーマは、は、XMLSpy パッケージ内にあります。マイドキュメントフォルダー[33]のフォルダー内で `WSDLDocumentation.xsd` と`WSDL20Documentation.xsd` を見つけることができます。

- `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2021\Documentation\WSDL.`
- `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2021\Documentation\WSDL20.`

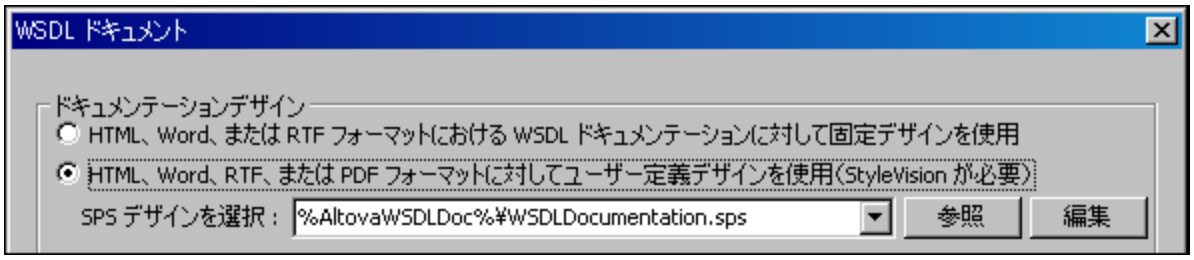
StyleVision 内でSPS デザインを作成するには、スキーマからのノードは、デザインテンプレートに置かれ、スタイルとプロパティが割り当てられます。リンク、テーブルとイメージなどの追加のコンポーネントもSPS デザインに追加することができます。このように、出力ドキュメント全体をSPS 内でデザインすることができます。StyleVision 内でSPS デザインを作成する方法は、StyleVision ユーザーマニュアル内で詳しく説明されています。

W SDL ドキュメントの生成のためにSPS を使用する利点は、SPS デザインの管理が完全におこなえる点です。W SDL ドキュメントのPDF 出力は、ユーザー定義 SPS が使用されている場合のみ、使用することができます。固定されたXMLSpy デザインが使用されている場合、PDF 出力を使用することはできません。

WSDL ドキュメントのために使用する SPS を指定する

SPS が作成された後、WSDL ドキュメントを作成するために使用することができます。WSDL ドキュメントの生成のために使用する SPS は、(**WSDL | ドキュメンテーションの生成** コマンドによりアクセスすることのできる) WSDL ドキュメンテーション ダイアログ内で選択することができます。このダイアログの**ドキュメント デザイン** ペインで、*ユーザーにより定義されたデザイン* ラジオ ボタンを選択します (*下のスクリーンショット*)。**参照** ボタンをクリックして、SPS を参照します。ダイアログの「**OK**」ボタンをクリックして、保存ダイアログ内で、出力 ファイルの名前とフォルダーを選択します。

メモ: SPS ファイルは、以下をベースにしたスキーマを正確に検索できる必要があります: `WSDLDocumentation.xsd` または `WSDL20Documentation.xsd` (*上を参照*).



(WSDL 1.1 と WSDL 2.0 のために つずつの) 2 つの編集することのできる SPS デザインは、XMLSpy により作成することができます。これらのデザインは、それぞれ、マイドキュメント フォルダー[33]: `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2011\Documentation\` の `WSDL` と `WSDL20` サブフォルダー内にあります。デザインには以下の名前が付けられています:

- `WSDL\WSDLDocumentation.sps`
- `WSDL20\WSDL20Documentation.sps`

SPS ファイルは、最近参照したその他の SPS ファイルとともに、ユーザー定義デザイン オプションのコンボボックスに表示されます。(*上のスクリーンショット参照*).

ドキュメンテーション デザイン ペインにある **編集** ボタンをクリックすると、StyleVision が起動され、選択された SPS が StyleVision ウインドウにて開かれます。StyleVision にて生成されるドキュメントのプレビューを行うには、作業用 XML ファイルが必要になります。このため、nanonull.xml という名前のサンプル XML ファイルが製品には同封されており、以下のフォルダーに配置されます マイドキュメント フォルダー[33]:

```
C:\Documents and Settings\<username>\My
Documents\Altova\XMLSpy2021\Documentation\WSDL(20)\SampleData
```

メモ: SPS を使った XBRL タクソノミドキュメンテーションの生成を行うには、お使いのコンピューターに StyleVision をインストールする必要があります。

28.14.5 WSDL ドキュメントの再解析

ドキュメントの再解析を行います。この機能は、例えば WSDL ドキュメントに割り当てられたスキーマが変更されたというような場合に必要になります。WSDL ドキュメントの再解析を行うことで、修正されたスキーマ ドキュメントから得られた情報により、ドキュメントが更新されます。

28.14.6 WSDL 2.0 に変換

「WSDL 2.0 に変換」コマンドは、WSDL ビューにて WSDL 1.1 ドキュメントがアクティブになっている時に有効になります。アクティブな WSDL 1.1 ドキュメントから WSDL 2.0 ドキュメントが生成されます。このコマンドをクリックすると、ファイル保存ダイアログが表示されます。XMLSpy により生成される WSDL 2.0 ファイルを保存する場所とファイル名を指定してください。

ファイルの保存ダイアログで「OK」をクリックすると、WSDL 2.0 ドキュメントが生成され、指定された場所に保存された後、WSDL ビューで新たに開かれます。生成されたファイルは、通常の WSDL 2.0 ドキュメントと同様に、必要に応じて編集することができます。

28.14.7 MapForce で WSDL プログラムコードを生成

MapForce で WSDL プログラムコードを生成コマンドにより、Altova の MapForce が（インストールされていれば）起動します。MapForce ではスキーマを他の DTD、XML スキーマ、またはデータベースへマッピングし XML の生成を行ったり、WSDL ファイルからプログラムコードを生成することができます。

28.15 SOAP メニュー

XMLSpy ではSOAP バージョン1.1 ならびに1.2、そしてW SDL バージョン1.1 ならびに2.0 がサポートされます。

SOAP: に関する[695] セクションでは、Altova により提供されている**nanonull.com timeservice** サーバーを使いながら、SOAP デバッガーを使用する方法について記述されています。SOAP デバッガーのテストを行うにはこのサービスを使用してください。以下のページに示されるAirportW eather ウェブサービスは、常に利用できるとは限りません。

SOAP 機能の使用:

- クライアントアプリケーションを実装することなく、ウェブサービスをテスト
- サードパーティーのウェブサービスを簡単にテスト

追加情報

これらの仕様に関する詳細については、以下を参照ください:

SOAP: http://www.w3.org/TR/SOAP/
WSDL: http://www.w3.org/TR/wsdl

28.15.1 新規 SOAP リクエストを作成

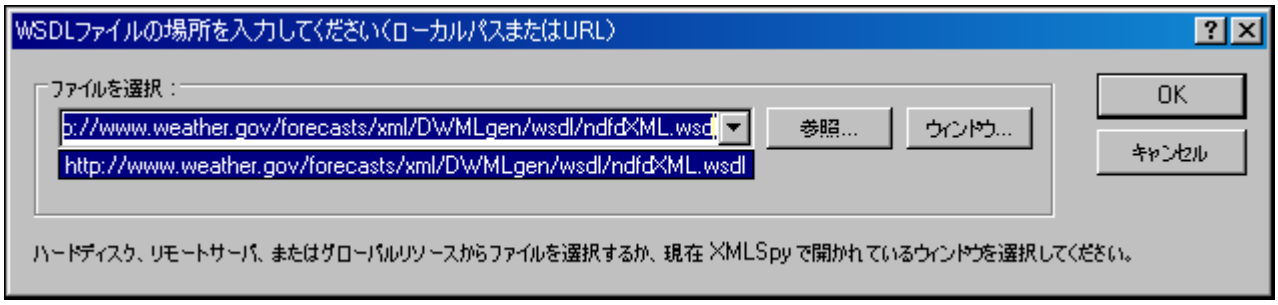
このコマンドにより、新たな SOAP リクエストドキュメントが作成されます。以下記されるステップが実行されます:

1. W SDL ファイルの場所を入力して、サーバーへ接続。[1381].
2. サーバーにより、オペレーションのリストが表示される。目的のSOAP オペレーションを選択します。[1381]
3. サーバーはSOAP リクエストフォームをXML 書式でレスポンドします。SOAP リクエストフォームを定義[1381].

以下では、US National Digital Forecast Database (NDFD) SOAP サービス(http://www.nws.noaa.gov/xml/)に対するSOAP リクエストを作成することで、上記のステップを説明します。

SOAP サーバーへ接続

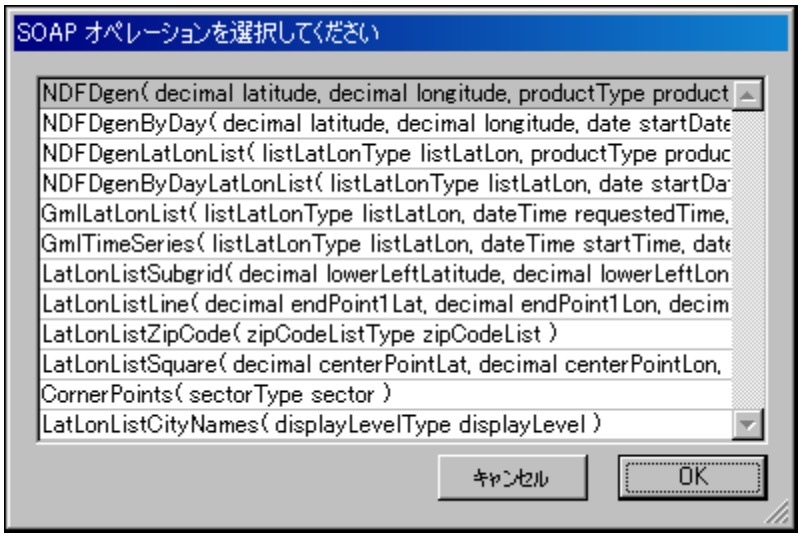
W SDL ファイルにより接続が行われます。この例では、W SDL ファイルのURI：
http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl. 接続を行うには、新規 SOAP リクエストを作成コマンドをクリックして、表示されるダイアログにて URI を入力します（以下のスクリーンショットを参照）。



OK をクリックして、選択を確認します。

目的のSOAP オペレーションを選択

サーバーによりオペレーションのリストが返され、ダイアログに表示されます（以下のスクリーンショットを参照）。



オペレーションを選択して、OK をクリックします。ここではNDFDgenByDay オペレーションを選択します。

SOAP リクエストを定義

サーバーからの応答はXML ファイルにより行われ、XMLSpy のテキストビューに表示されます。上で選択したオペレーションに対しては、以下にあるXML ファイルが受信されます。

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:m0="http://www.weather.gov/forecasts/xml/DWMLgen/schema/DWML.xsd">
   <SOAP-ENV:Body>
```

```
      <m:NDFDgenByDay
xmlns:m="http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl"
       SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <latitude xsi:type="xsd:decimal">0.0</latitude>
         <longitude xsi:type="xsd:decimal">0.0</longitude>
         <startDate xsi:type="xsd:date">1967-08-13</startDate>
         <numDays xsi:type="xsd:integer">0</numDays>
         <format xsi:type="m0:formatType">String</format>
      </m:NDFDgenByDay>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

必要に応じてパラメーターを入力します（以下の太字を参照）：

```
<SOAP-ENV:Body>
   <m:NDFDgenByDay
xmlns:m="http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <latitude xsi:type="xsd:decimal">45</latitude>
         <longitude xsi:type="xsd:decimal">-90</longitude>
         <startDate xsi:type="xsd:date">2009-10-13</startDate>
         <numDays xsi:type="xsd:integer">1</numDays>
         <format xsi:type="m0:formatType">24 hourly</format>
   </m:NDFDgenByDay>
</SOAP-ENV:Body>
```

これでSOAP リクエストの定義が完成しました。次のステップでは、リクエストの送信[1382]を行います。

28.15.2 サーバーにリクエストを送信

SOAP リクエストの定義の方法は、コマンド 新規のSOAP リクエストの作成[1380]内で説明されています。コマンドの選択時、XMLSpy 内のアクティブなドキュメントである必要があります。このコマンドは、SOAP リクエストをSOAP サーバーに送信します。

SOAP リクエストの送信後、SOAP サーバーからのレスポンスが受信されます。このレスポンスはXMLSpy のテキストビューに表示されるXML ドキュメントです。下に表示されるXML ドキュメントのスクリーンショットは、セクション 新規のSOAP リクエストの作成[1380]内で定義したSOAP リクエストのレスポンス内に返されたものです。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SO
http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:
http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/enc
   <SOAP-ENV:Body>
      <ns1:NDFDgenByDayResponse xmlns:ns1="http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/n
         <dwmlByDayOut xsi:type="xsd:string"><?xml version="1.0"?>
<dwml version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/
xsi:noNamespaceSchemaLocation="http://www.nws.noaa.gov/forecasts/xml/DWMLgen/schema/DWML.xsd">
  <head>
  <data>
    <location>
      <location-key>point1</location-key>
      <point latitude="45.00" longitude="-90.00"/>
    </location>
    <moreWeatherInformation applicable-location="point1">
http://forecast.weather.gov/MapClick.php?textField1=45.00&amp;amp;textField2=-90.00
    </moreWeatherInformation>
    <time-layout time-coordinate="local" summarization="24hourly">
    <time-layout time-coordinate="local" summarization="12hourly">
    <parameters applicable-location="point1">
      <temperature type="maximum" units="Fahrenheit" time-layout="k-p24h-n1-1">
        <name>Daily Maximum Temperature</name>
        <value>45</value>
      </temperature>
      <probability-of-precipitation type="12 hour" units="percent" time-layout="k-p12h-n2-2">
        <name>12 Hourly Probability of Precipitation</name>
        <value>14</value>
        <value xsi:nil="true"/>
      </probability-of-precipitation>
    </parameters>
  </data>
</dwml>
</dwmlByDayOut>
      </ns1:NDFDgenByDayResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP リクエストを再利用して保存する

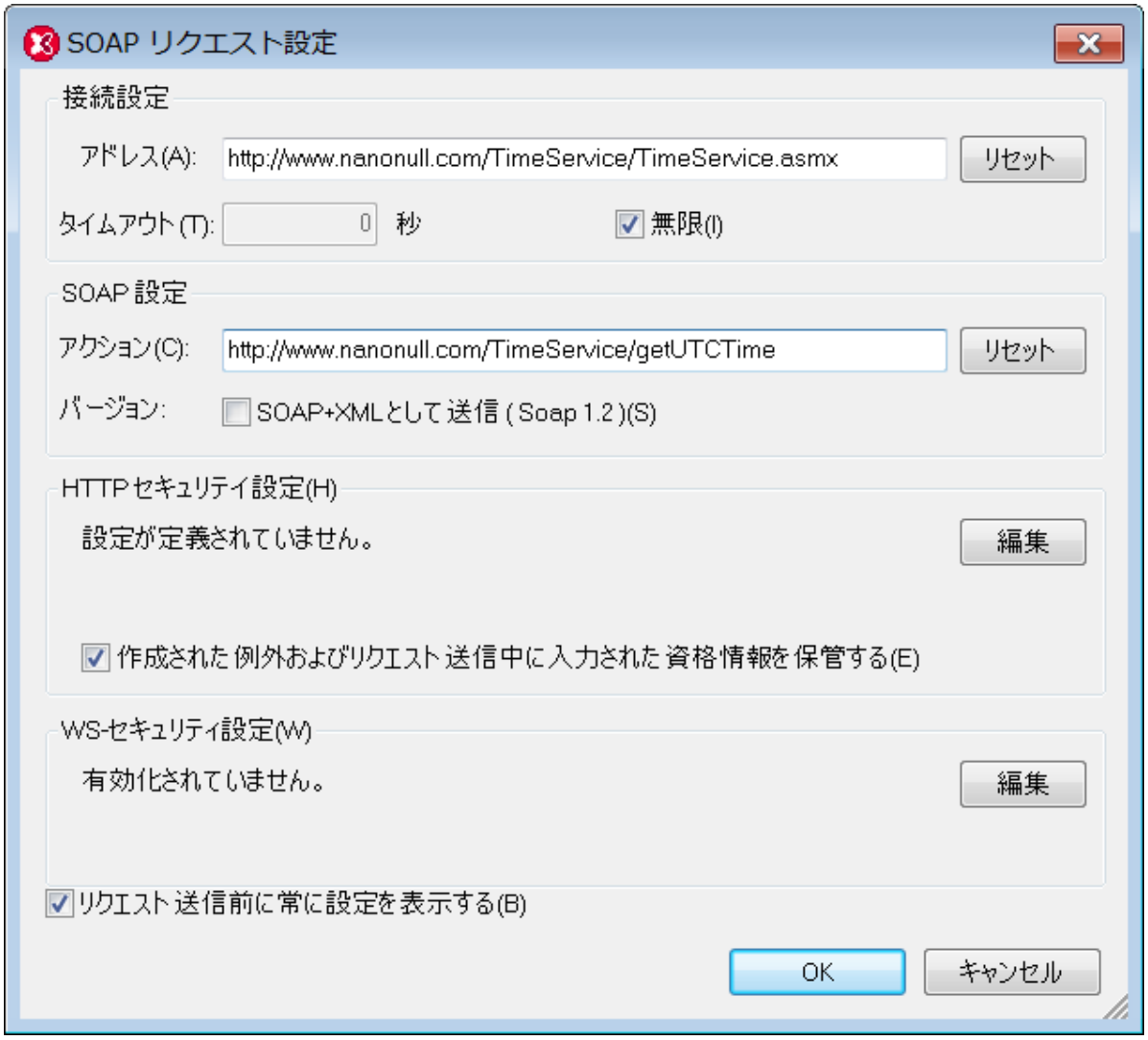
XMLSpy によりSOAP リクエストを保存して、後で再利用することができます。以下の手順を行ってください：

1. SOAP リクエスト XML ドキュメントを保存します（**「ファイル | 名前を付けて保存」**）。
2. SOAP リクエスト ファイルを閉じます。
3. SOAP リクエスト XML ドキュメントをもう一度開き、メニューオプション**「SOAP | リクエストをサーバーに送信する」**を開きます。（すべてのXML ファイルをSOAP リクエスト ドキュメントとして使用することができます）

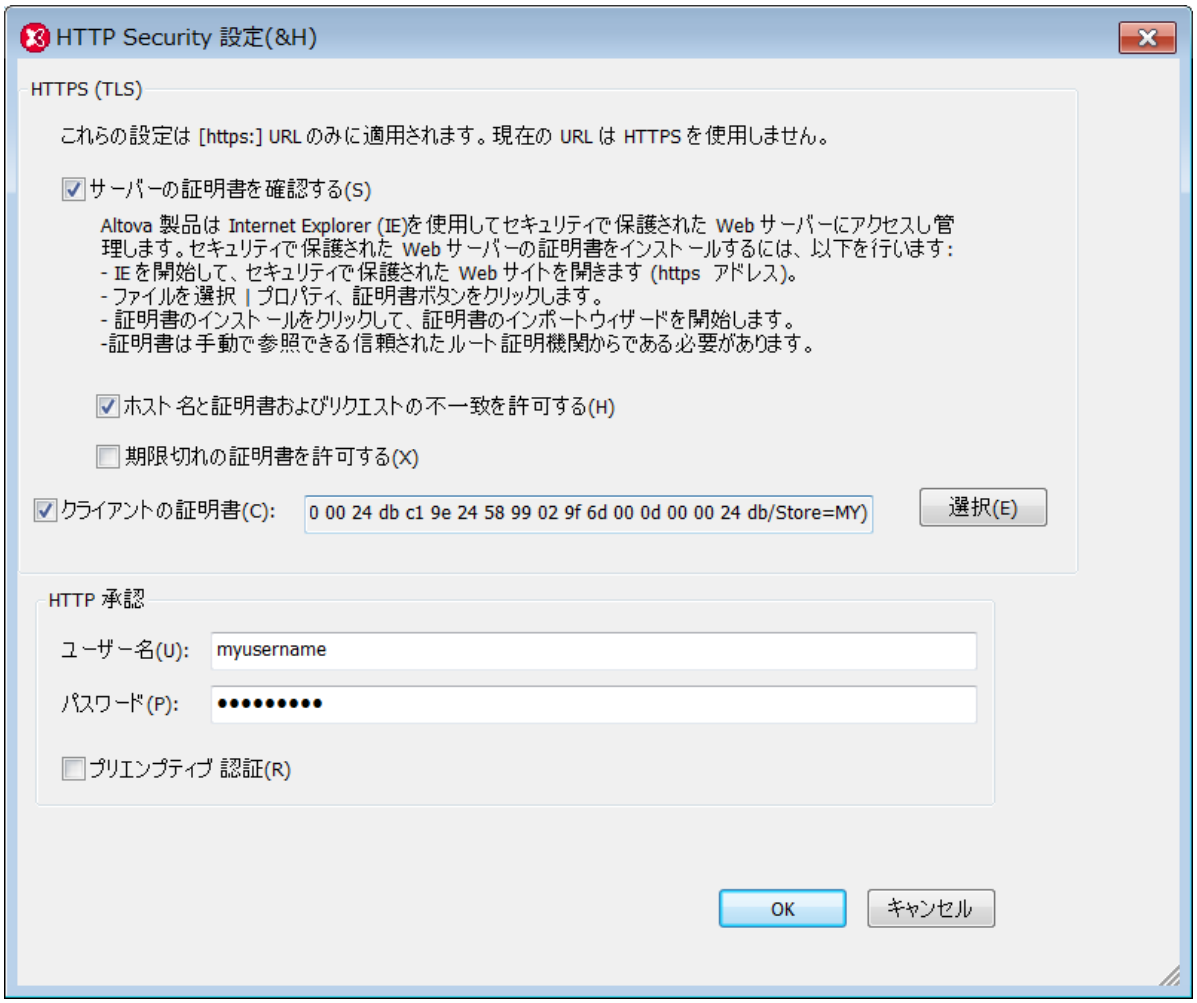
28.15.3 SOAP リクエスト設定

このコマンドは、SOAP リクエスト の異なる設定を指定することのできる、SOAP リクエスト設定 1380 ダイアログを表示します。（*下のスクリーンショット*）。これらの設定は下で説明されています。

1. SOAP リクエストドキュメントをアクティブにします。
2. メニューオプション**Soap | SOAP リクエスト設定**を選択します。これにより、SOAP リクエスト設定 ダイアログボックスが開かれます（*下のスクリーンショット*）。

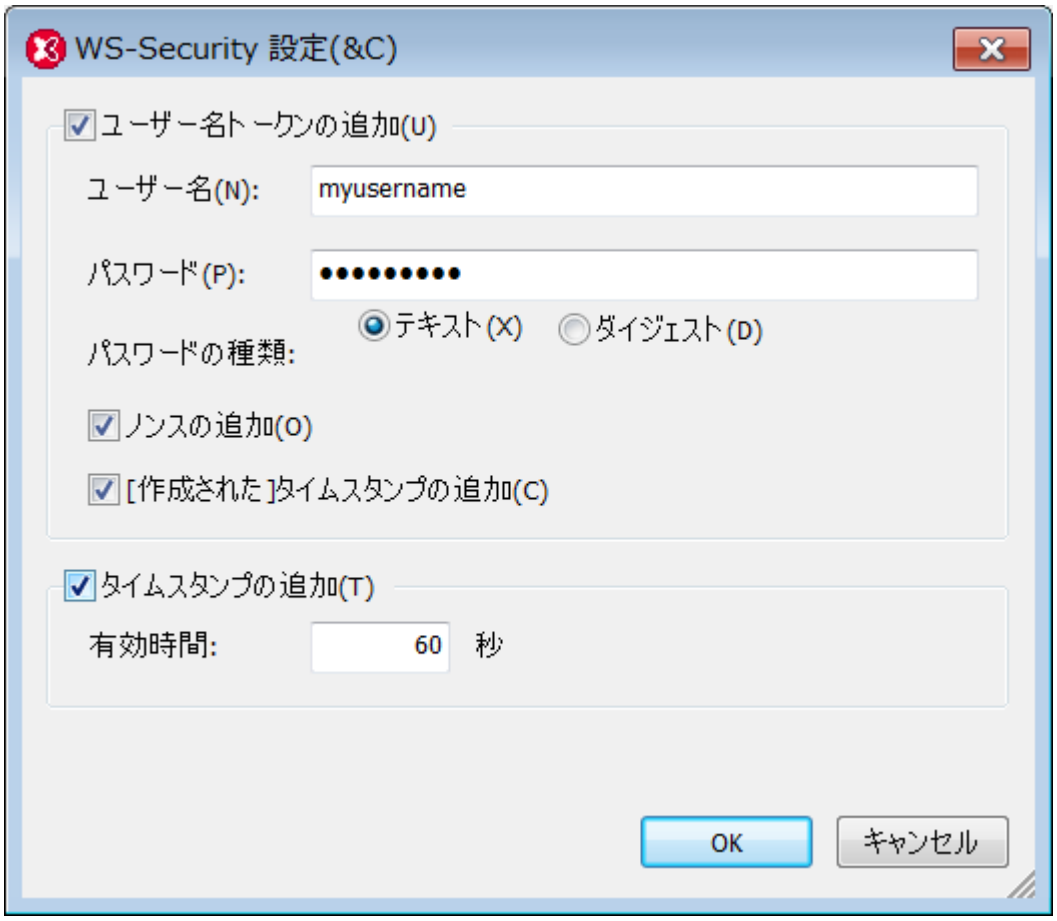


3. *アドレス* フィールドに、希望する接続 エンドポイントに入力します。SOAP リクエストがXMLSpy 内のW SDL ファイルから作成されている場合、*アドレス* フィールドの値は、W SDL 内で選択されたエンドポイントの場所になります。「**リセット**」をクリックして、このエンドポイントを取得します。接続の*タイムアウト*の値は秒の単位で指定することができます。*タイムアウト*値を設定しない場合は、*永続のチェックボックス*を選択します。
4. *アクション* フィールドに実行するSOAP *アクション*を入力します。リクエストの送信をSOAP 1.2 として送信する場合、あと *SOAP+XML (SOAP 1.2)* として送信チェックボックスをチェックします。SOAP リクエストがXMLSpy 内のW SDL ファイルから作成された場合、SOAP *アクション*は、W SDL ファイル内の対応するSOAP バインドオペレーション下で受け入れられます。この場合、SOAP バージョンはW SDL により選択されます(SOAP バージョンは以下のHTTP ヘッダーコンテンツ型値に影響を与えます: `text/xml` または `application/soap+xml`。)「**リセット**」をクリックしてSOAP *アクション*をW SDL ファイルから取得します。
5. HTTP セキュリティ設定 ペインは、セキュリティ設定リストの概要を提供します。*リクエストの送信中に例外を保管する* オプションがチェックされていると、リクエストが送信されると全ての設定が保存され、次のリクエストのために再利用することができます。「**編集**」ボタンをクリックして、 HTTP セキュリティ設定 ダイアログを表示します (*下のスクリーンショット*)。サーバーの証明書のインストールの方法は、下で説明されています。



「ホスト名の不一致」（サーバー証明書内のホスト名と使用する実際のアドレス）または*「期限の切れたサーバーの証明書」*を許可する場合、ダイアログ内の*オプション*をチェックします。サーバーが *クライアント 証明書* を必要とする場合、証明書の場所を指定します。認証がサーバーにより必要とされる場合、*ユーザー名* と *パスワード* を標準の認証のために指定します。最初のサーバーへのクライアントリクエストに必要な 認証の情報が含まれている場合、この処理は、*プリエンプティブ 認証*として扱われます。サーバーにより必要とされる場合、*プリエンプティブ 認証* オプションを選択します。それ以外の場合は、*プリエンプティブ 認証* オプションを選択しないでください。

6. トランスポートレイヤー上のセキュリティに加え、（HTTP セキュリティ設定） Web サービスにより必要とされる場合、Web サービス セキュリティ設定を指定することができます。WS セキュリティ設定 ペインの「**編集**」ボタンをクリックして、WS セキュリティ設定 ダイアログを表示します（*下のスクリーンショット*）。セキュリティ情報には、ユーザー名、パスワード、パスワードの種類、自動的に生成されたノンスコード文字列、および、タイムスタンプが含まれています。セキュリティ情報の有効期間（*タイムスタンプの追加*）も指定することができます。ダイアログは、セキュリティ情報を含むXML フラグメントを作成し、リクエスト内でSOAP リクエストに埋め込みます。*下のリストを参照してください*。.



7. 完了すると、click **OK**.

信頼される証明書について

Altova 製品は、Internet Explorer (IE) を使用して、信頼できるウェブサーバーの信頼できる証明書を管理しています。IE 内にウェブサーバーの証明書をインストールすると、警告を発行せずに、または、処理を中断することなく、IE はウェブサーバーにアクセスすることができます: 安全なサーバーの証明書をインストールするには以下を行います:

- Internet Explorer 8 で、安全な ウェブサイトを開きます。
- 「ファイル | プロパティ」を選択して、証明書 ボタンをクリックします。
- 「**証明書のインストール**」をクリックして、証明書のインポート ウィザードを開始します。(「**ツール | インターネット オプション | コンテンツ | 証明書 | インポート**を使用してこのウィザードにアクセスすることができます)
- 手動で参照することのできる、信頼することのできるルート証明書機関ストアに証明書を保管します。
- ウィザード ステップを完了し、証明書とプロパティ ダイアログをそれぞれ「OK」をクリックして閉じます。Internet Explorer を開く必要があるかもしれません。

メモ: SOAP メソッドと対応する SOAP アクションへのアクセスが存在する場合のみ SOAP アクション設定を変更してください。

Web サービスセキュリティ情報

Web サービスの一部は、ユーザー認証を必要とします。(Web サービス セキュリティの層は、サーバーにより実装される HTTP セキュリティの層に追加されます)。Web サービス 認証の情報は、SOAP リクエストに下の構造を持つ XML フラグメントとして保管されます。この

XML フラグメント は、 W S セキュリティ設定 ダイアログに入力される W eb サービス 認証の情報から、SOAP リクエスト 内で自動的に生成されます。

```
<wsse:Security xmlns:wsse="…" xmlns:wsu="…" SOAP-ENV:mustUnderstand="true">
   <wsse:UsernameToken>
      <wsse:Username>usr</wsse:Username>
      <wsse:Password Type="...#PasswordText">pwd</wsse:Password>
      <wsse:Nonce EncodingType="…#Base64Binary">UqrtD963797WBRgWiJPu2w==</wsse:Nonce>
      <wsu:Created>2014-11-17T16:08:07.016Z</wsu:Created>
   </wsse:UsernameToken>
   <wsu:Timestamp>
      <wsu:Created>2014-11-17T16:08:07.016Z</wsu:Created>
      <wsu:Expires>2014-11-17T16:09:07.016Z</wsu:Expires>
   </wsu:Timestamp>
</wsse:Security>
```

28.15.4 SOAP デバッグセッション

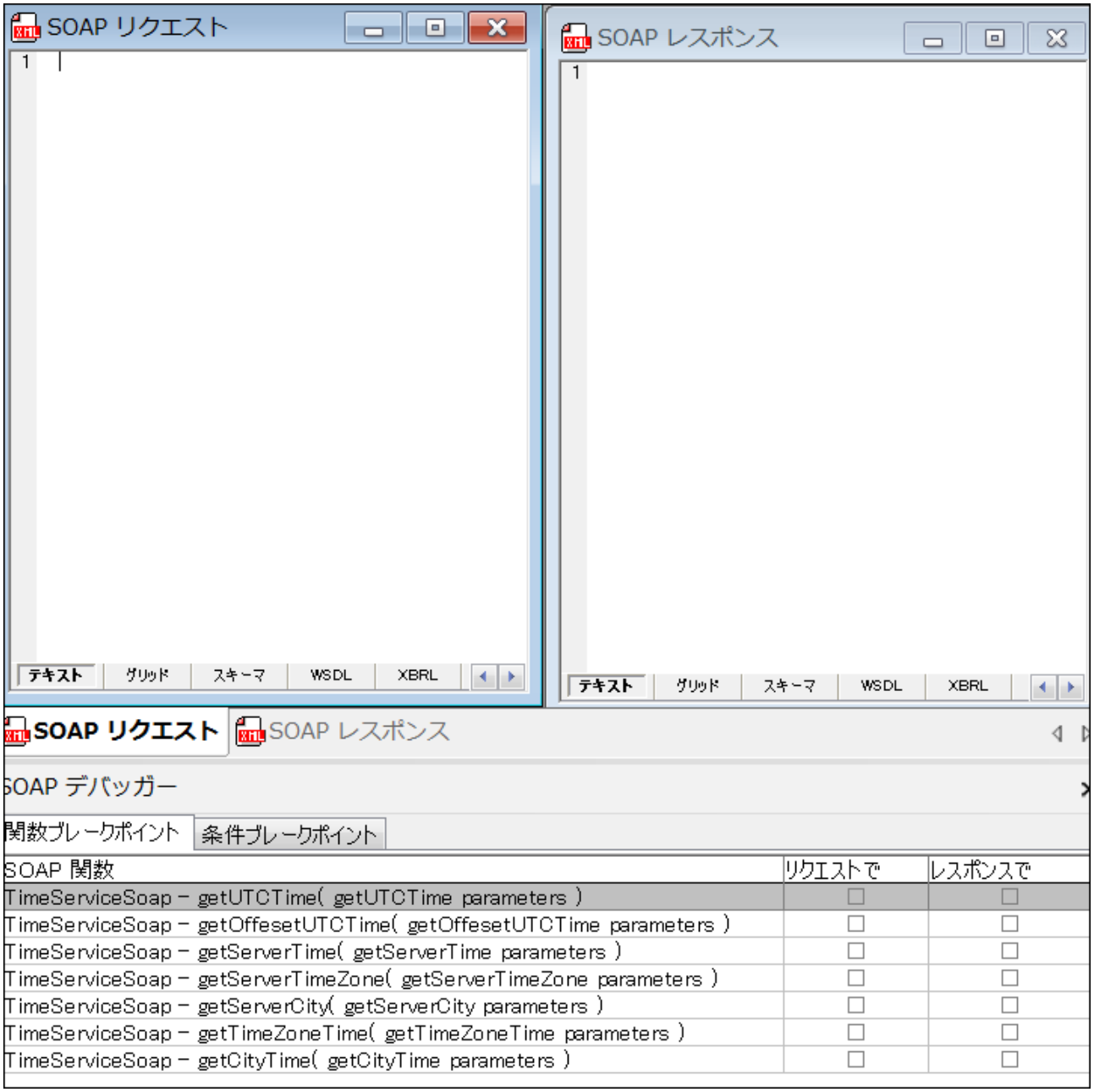
このコマンドにより SOAP デバッグセッションが開始されます。

- コマンドを選択すると、ダイアログボックスが表示されます。W SDL ファイルの位置を選択してください（通常は URL になります）。

  ファイルの選択は、グローバルリソース(アイコンをクリックして、表示されるダイアログにてグローバルリソースを選択）や、XMLSpy で開かれているファイルからも選択することができます。
- その後表示されるダイアログにて、デバッガープロキシサーバーとウェブサービスに必要なソースならびにターゲットのポートを選択します

  この操作により、SOAP デバッガープロキシサーバーが非アクティブ状態で開かれます。SOAP ツールバーアイコンをクリックすることで、SOAP デバッガーが開始され、クライアントのリクエストを待つ状態になります。

  より詳細な情報についてはSOAP 695 のセクションを参照ください。



28.15.5 実行

(C) 2015-2021 Altova GmbH

このコマンドにより、SOAP プロキシサーバーがアクティブになり、WSDL ファイルがブレークポイントに到達するまで処理されます。対応する SOAP ドキュメントが、SOAP ドキュメントウィンドウにて表示されます。

28.15.6 単一のステップ

このコマンドにより、送信する SOAP リクエストや受信したレスポンスを、シングルステップで実行することができます。各リクエストならびにレスポンスに対して SOAP デバッガーが停止します。プロキシサーバーが非アクティブな場合、プロキシサーバーも開始されます。

28.15.7 次のリクエストでブレーク

このコマンドにより、次の SOAP リクエストにてデバッガーが停止し、SOAP リクエスト ドキュメントウィンドウにてデータが表示されます。ウェブサービスへの送信が行われる前に、このウィンドウにて直接データを編集することができます。

28.15.8 次のレスポンスでブレーク

このコマンドにより、次の SOAP レスポンスにてデバッガーが停止され、SOAP Response ドキュメントウィンドウにてデータが表示されます。クライアントへの送信が行われる前に、このウィンドウにて直接データを編集することができます。

28.15.9 プロキシサーバーの停止

このコマンドによりデバッガープロキシサーバーが停止します。

28.15.10 SOAP デバッガーオプション

SOAP デバッガーオプション ダイアログにより、コンピューターの IP アドレスと、下にリストされる他のデバッガーオプションを指定することができます（*下のスクリーンショット*）。SOAP | SOAP デバッガーオプション メニューコマンドを使用してダイアログにアクセスしてください。

- *コンピューターのアドレス*: デバッガーが実行されるプロキシのサーバーのアドレス。プロキシのサーバー上のデバッガーは、ネットワーク上のマシンからリクエストを受け、Web サービスに送ります。XMLSpy 内でデバッガーは作動するため、XMLSpy がインストールされているマシンは、プロキシ サーバーとしての役割を果たします。マシンのIP アドレスは、自動的に検知され、このフィールドに入力されます。IP アドレスが自動的に検知されない場合、IP アドレス(`http` アドレスとして) をこのフィールドに入力する必要があります。使用中のコンピューターのIP アドレスを検出するには、コマンド プロンプト ウィンドウを開き、コマンド `ipconfig /all` を入力し、**Enter** を押す。
- *タイムアウト*: この値は、SOAP デバッガーがブレークポイント内に存在する時間の値を指します。デフォルトの値は5 秒です。
- *入力ヘルパーの非表示; Hide プロジェクト/情報ウィンドウ*: これらのオプションはSOAP デバッガー ウィンドウのためのスクリーンの表示スペースを増やすために役に立ちます。

(C) 2015-2021 Altova GmbH

28.16 XBRL メニュー

以下に記されるXBRL メニュー(*下のスクリーンショット*)のコマンドは、タクソノミがXBRL ビューで編集されている際に有効になります。これらのコマンドは下にリストされており、このセクションのサブセクションで更に詳しく説明されています。

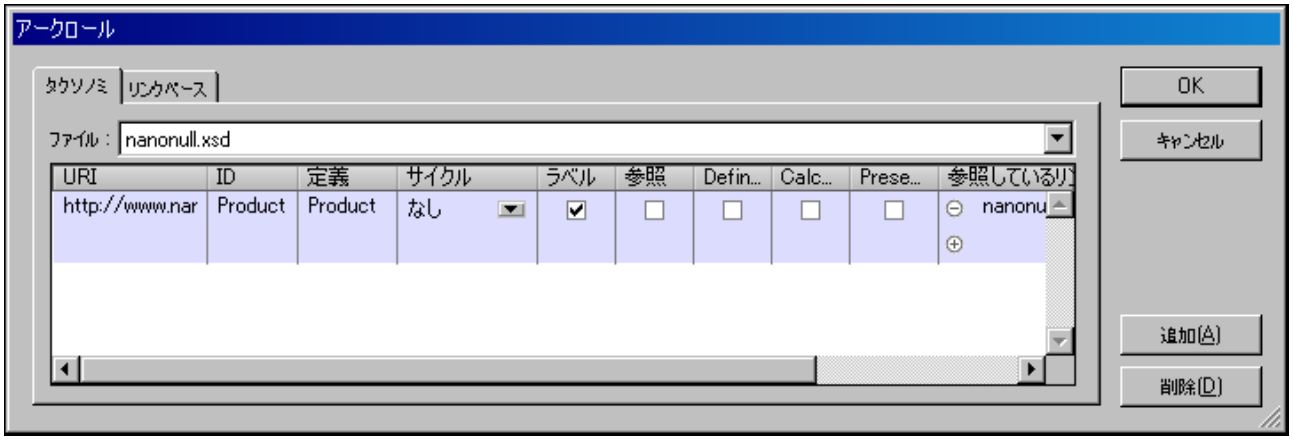
- アークロール[1391]: アークロールを定義します。
- リンクロール[1393]: リンクロールを定義します。
- 名前空間プレフィックス[1394]: タクソノミ名前空間を管理します。
- ターゲット名前空間の設定[1395]: タクソノミのターゲット名前空間を定義ならびに宣言します。
- パラメーターの値[1396]: 編集することのできるダイアログ内のフォーミュラとテーブル パラメーターを表示します。
- インポート/参照[1396]: XBRL タクソノミのインポートやリンクベースの参照を行います。
- ID によりフォーミュラのコンポーネントを検索する[1398]: ユーザーにより提供されたID によりコンポーネントを検索します。
- ドキュメンテーションの生成[1398]: XBRL タクソノミのドキュメンテーションを生成します。
- ビューの設定[1401]: XBRL ビューのデフォルト値をセットします。
- MapForce でDB/Excel/CVS からXBRL を生成[1403]: Altova MapForce を起動して、XBRL インスタンスファイルの生成を行います。
- StyleVision でHTML/PDF/W ord としてXBRL を提供[1403]: Altova StyleVision を起動して、XBRL レポートのデザインを行います。
- フォーミュラの実行[1403]: アクティブなXBRL インスタンスドキュメントと関連するDTS から、フォーミュラとアサーションを実行します。
- テーブルの生成[1406]: XBRL インスタンスからXBRL テーブルを生成します。
- 重複の検知[1408]: XBRL インスタンス内の重複ファクトを検知します。
- XULE の実行[1409]: XBRL インスタンスドキュメント上でXULE ドキュメントを実行します。
- インラインXBRL の変換[1410]: XHTML ドキュメントのインラインXBRL パートをXBRL として生成します。
- サーバー上でEDGAR を検証[1410]: RaptorXML+XBRL Server 上のEDGAR を使用してXBRL インスタンスを検証します。
- 処理オプション[1411]: XBRL インスタンスの処理のためにオプションの設定を有効化します。

XBRL に関する詳細については、XBRL[721] と編集ビュー| XBRL ビュー[243]を参照ください。

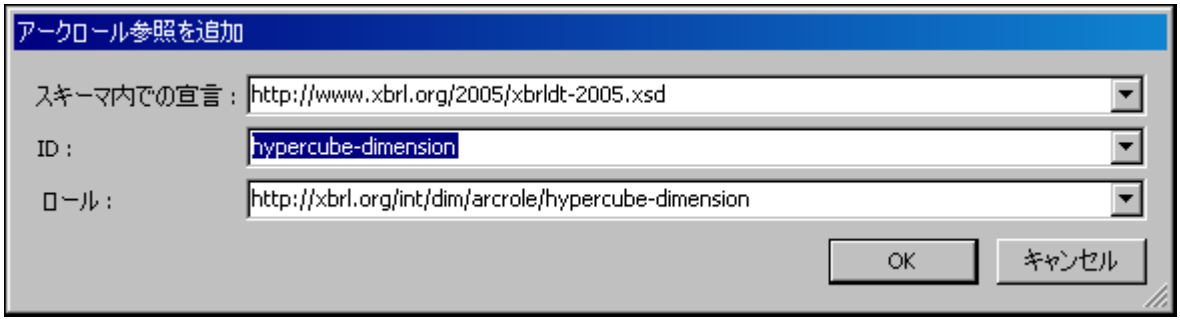
28.16.1 アークロール

アークロールコマンドにより、アークロールダイアログが表示され、タクソノミに対してアークロールを作成することができます。アークロールはコンセプト定義ファイルのappinfo 要素内に保管されます。アークロールによりアークの役割が指定されます。

アークロールダイアログの**タクソノミタブ**では、編集可能または*アークロールを含むタクソノミやリンクロール*だけがコンボボックスに表示されます。**追加**ボタンによりアークロールをタクソノミに加えることができます。アークロールのURI、ID、定義、サイクルを定義します。アークロールが利用可能となる種類のリレーションシップを定義するには、目的のリレーションシップの種類に対応するボックスをクリックしてください。アークロールへの参照を行うリンクベースは、参照しているリンクベースファイルカラムにて追加することができます。



*リンクベース*タブでは、*タクソノミ*の*アークロール*に対して別のビューを提供します。このビューでは、（計算や表示*リンクベース*などの）各*リンクベース*に従って*アークロール*の追加や閲覧を行います。*コンボボックス*から*リンクベース*を選択し、必要に応じて*アークロール*を追加または削除します。「**追加**」*ボタンをクリック*すると、*アークロール参照を追加*ダイアログが表示されます（以下の*スクリーンショット*を参照）。

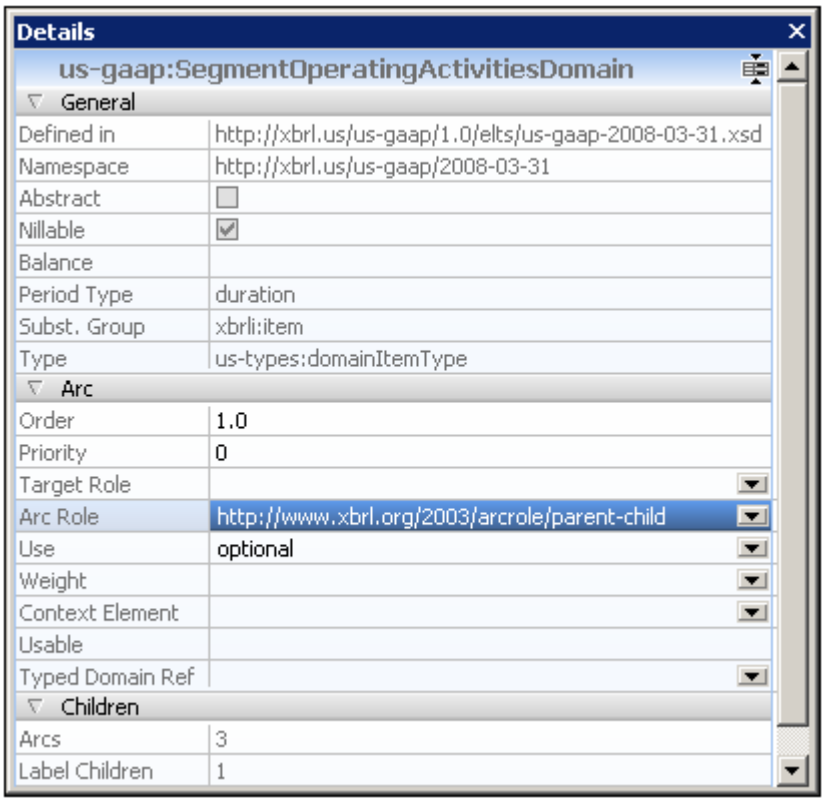


*このダイアログ*における各エントリーは*コンボボックス*となっており、利用可能なオプションから選択を行うことができます。*スキーマ内での宣言*フィールドにより、*アークロール*が定義されている*タクソノミ*を選択することができます。*ID* と*ロール*コンボボックスでは、利用可能な*アークロール*を選択することができます。*アークロール*を選択し「OK」*をクリック*すると、*リンクベース*に参照が追加されます。*タクソノミタブ*では、参照した*アークロール*により、参照している*リンクベース*が*参照している リンクベースファイル*カラムに追加されます。この*アークロール*を特定種類の*リレーションシップ*に対して使用する場合、適切な*リレーションシップ*のチェックボックスにチェックを入れる必要があります。

*タクソノミ*にて*アークロール*が作成されると、*アーク*の定義で利用可能となっている*リレーションシップ*にて*アーク*の*ロール*を指定するために使用することができます。例えば、上の*スクリーンショット*では*アークロール*は*ラベルリレーションシップ*に対して利用することができるようになっています。

*アーク*の*アークロール*は詳細入力ヘルパーで選択されます（以下の*スクリーンショット*では、*アークロール*がハイライトされています）。

(C) 2015-2021 Altova GmbH



アークの to 終端にある要素が選択されている状態で、目的のアイテムを詳細入力ヘルパー内にあるアークロールエントリーのドロップダウンリストから選択します。

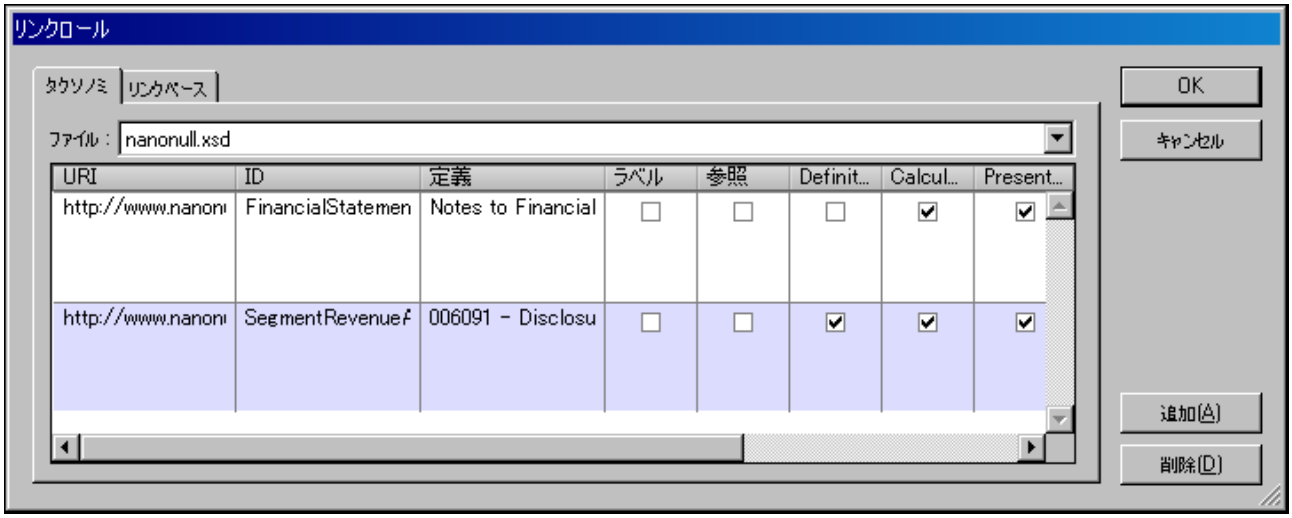
28.16.2 リンクロール

リンクロールコマンドを選択すると、リンクロールダイアログが表示され、タクソノミに対してリンクロールの作成を行うことができます（以下のスクリーンショットを参照）。リンクロールはコンセプト定義ファイルの appinfo 要素内に保管されます（以下の表示を参照）。リンクロールは definitionLink 要素だけではなく、（例えば calculationLink や presentationLink 要素など）その他種類のリレーションシップでも使用されます。

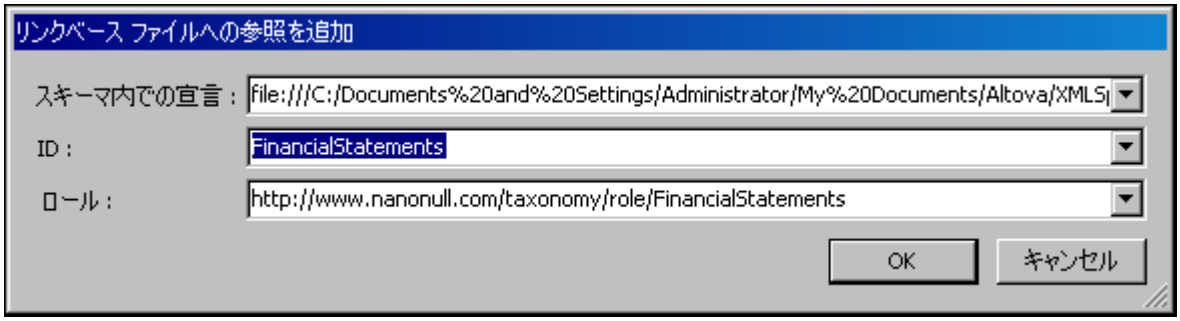
```
<xs:appinfo>
   <link:roleType id="SegmentRevenueAndOperatingIncome"
   roleURI="http://www.nanonull.com/taxonomy/role/SegmentRevenueAndOperatingIncome">
      <link:definition>006091 - Disclosure - Segment Revenue and Operating
Income</link:definition>
      <link:usedOn>link:calculationLink</link:usedOn>
      <link:usedOn>link:definitionLink</link:usedOn>
      <link:usedOn>link:presentationLink</link:usedOn>
   </link:roleType>
</xs:appinfo>
```

上の表示では、このリンクロールを使用することができるリレーションシップの種類を指定する usedOn 要素が使用されていることに注目してください。

*リンクロールダイアログのタクソノミタブ*では、**追加ボタンをクリック**することで、*タクソノミに*リンクロールを追加することができます。*リンクロール*の URI や ID を定義します(上の表示を参照ください)。*どのリレーションシップ*に対して*リンクロール*が利用可能かを指定するには、適切な*リレーションシップのチェックボックス*にチェックを入れてください。参照している*リンクベースファイル*カラムでは、各*リンクロール*に対して*リンクロール*を参照する*リンクベースファイル*を追加または削除することができます。



*リンクベースタブ*では、*タクソノミのリンクロール*に対して別のビューを提供します。このビューでは、(計算や表示*リンクベース*などの)各*リンクベース*に従って*リンクロール*の追加や閲覧を行います。*コンボボックス*から*リンクベース*を選択し、必要に応じて*リンクロール*を追加または削除します。例えば、*リンクロール*を計算*リンクベース*に追加することができます。**追加ボタンをクリック**すると、*リンクベースファイル*への参照を追加ダイアログが表示されます(以下の*スクリーンショット*を参照)。



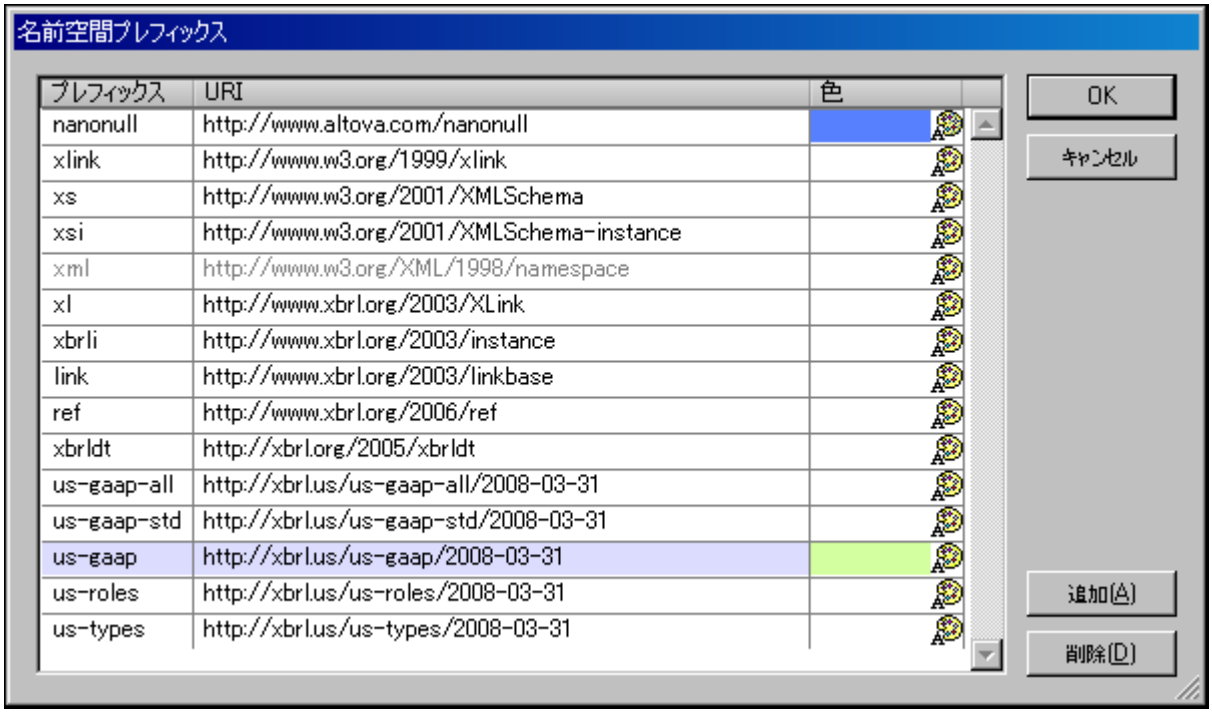
*このダイアログ*における各エントリーは*コンボボックス*となっており、利用可能な*オプション*から選択を行うことができます。*スキーマ内での宣言*フィールドにより、*リンクロール*が定義されている*タクソノミ*を選択することができます。*ID* と*ロール*コンボボックスでは、利用可能な*リンクロール*を選択することができます。*リンクロール*を選択し、**OK** を*クリック*すると、*リンクベース*に参照が追加されます。*タクソノミタブ*では、参照した*アークロール*により、参照している*リンクベース*が*参照している*リンクベースファイル*カラム*に追加されます。この*リンクロール*を特定種類の*リレーションシップ*に使用する場合、適切な*リレーションシップのチェックボックス*にチェックを入れる必要があります。

*タクソノミ*にて*リンクロール*が作成されると、リレーションシップの作成 760 時に使用することができます。

28.16.3 名前空間プレフィックス

名前空間プレフィックスコマンドにより、名前空間*プレフィックス*ダイアログが表示され、インポートされた*タクソノミ*も含めて、*タクソノミ*内にある全ての名前空間が表示されます(*下のスクリーンショットを参照*)。名前空間*プレフィックス*ダイアログでは、名前空間と*プレフィックス*の編集を行うことができ、各名前空間に対して背景色を設定することができます。名前空間に対して背景色がセットされると、メインウィンドウならびに

入力ヘルパーにて、その名前空間にある要素が、選択された背景色で表示されます。 特定の名前空間に対する色の設定は、XBRL ビューで開かれている全てのタクソノミに対して適用されることにご注意してください。

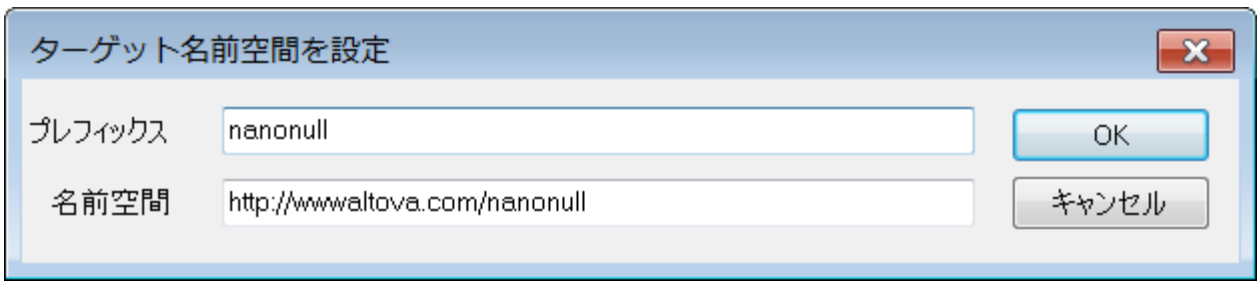


名前空間の追加や削除を行うには、それぞれ「追加」と「削除」ボタンを使用してください。カラーパレットにより、名前空間に色を割り当てることができます。 名前空間ダイアログにおける編集が完了した際には、「OK」をクリックしてください。

タクソノミのターゲット名前空間も名前空間ダイアログに表示されます。ターゲット名前空間の修正は、このダイアログではなく[ターゲット名前空間の設定](1395)コマンドにより行ってください。ターゲット名前空間に関する詳細な情報については、この[ドキュメンテーションのXBRL セクション](747)を参照ください。

28.16.4 ターゲット名前空間の設定

「ターゲット名前空間の設定」コマンドにより、タクソノミドキュメントに対してターゲット名前空間をセットすることができます。このコマンドをクリックすると、ターゲット名前空間を設定ダイアログが表示されます(以下のスクリーンショットを参照)。目的のターゲット名前空間をプレフィックスとともに入力します。「OK」をクリックして完了してください。



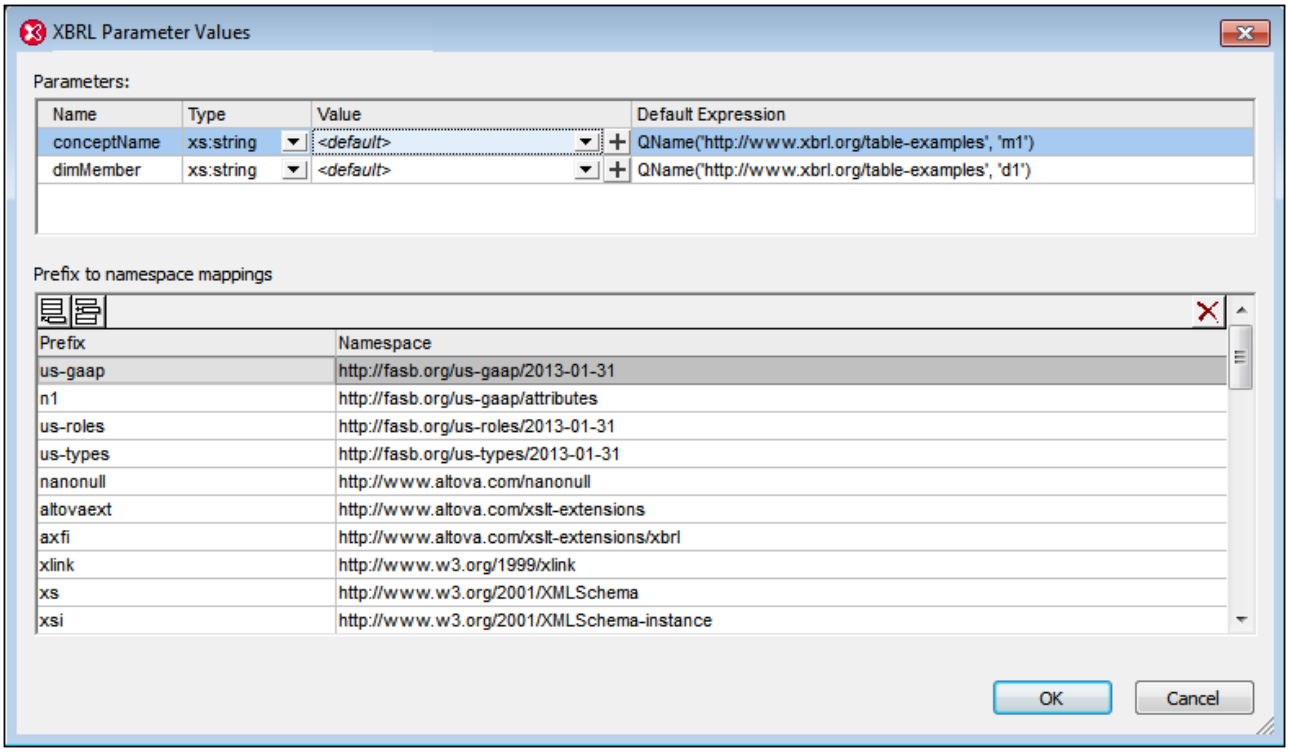
ターゲット名前空間は以下のように定義、宣言されます:

```
<xs:schema targetNamespace="http://www.altova.com/XBRL/Taxonomies"
        xmlns:ns1="http://www.altova.com/XBRL/Taxonomies" >
        ...
</xs:schema>
```

上の記述では、ターゲット名前空間は targetNamespaces 属性とともに定義され、ns1 というプレフィックスで宣言されています。

28.16.5 パラメーターの値

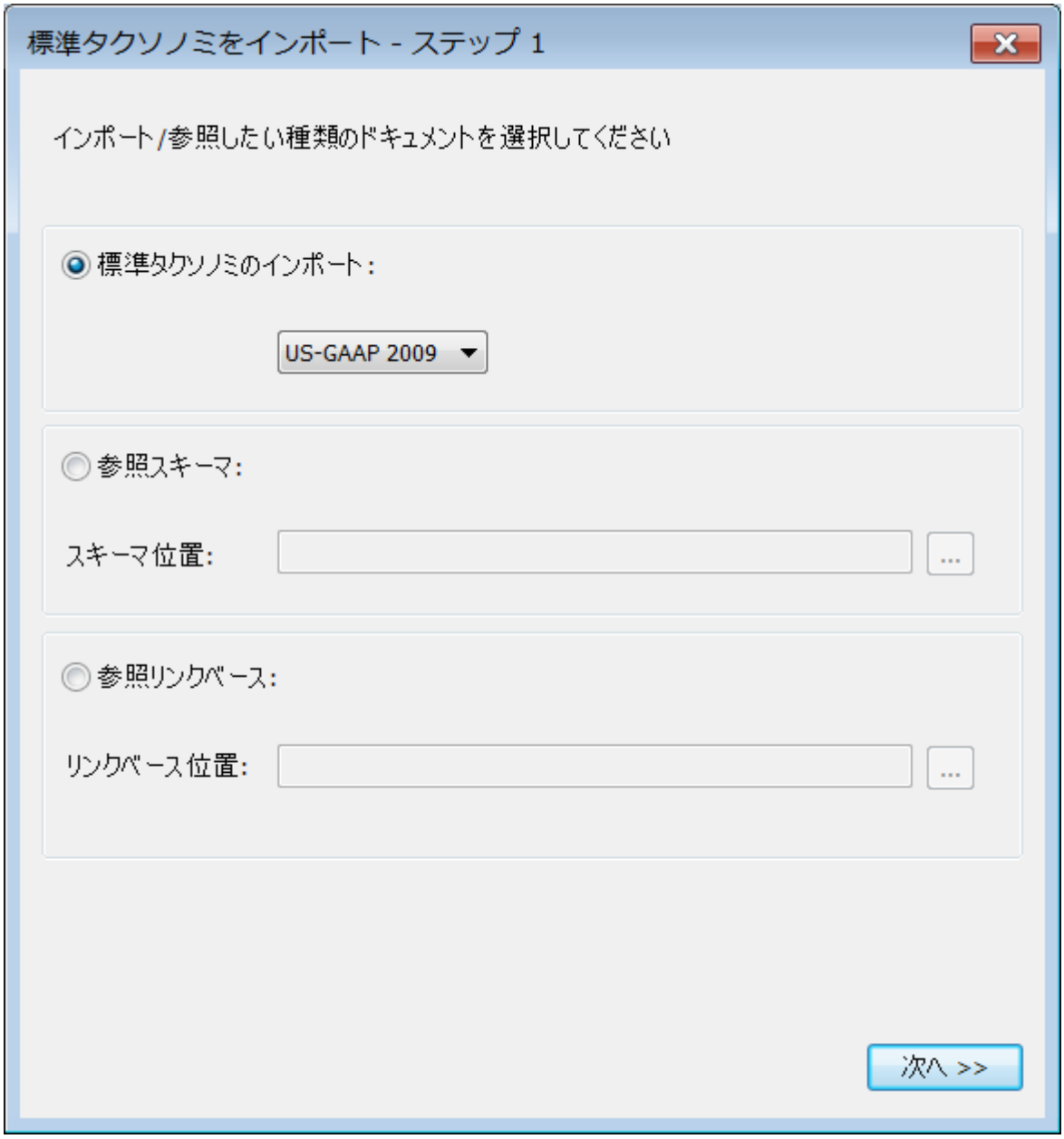
パラメーター値 コマンドは、XBRL パラメーター値 ダイアログを表示します(*下のスクリーンショット*)。フォーミュラタブ(フォーミュラ パラメーター)およびテーブルタブ(テーブル パラメーター) 内で定義されたパラメーターが表示されます。XBRL パラメーター値 ダイアログ内では、パラメーターのデータ型を編集し、デフォルトの値を上書きするダイアグラムを使用して与えられる、パラメーター値が与えられます。ダイアグラムを使用して入力するパラメーター値はデフォルトの値を上書きします。テーブル パラメーターは、複数の値を取ることができ、値列内の+ アイコンをクリックしてパラメータのためのパラメーター値を追加することができます。



このダイアログ内で割り当てられたグローバル パラメーターの値は、テーブル パラメーターのためにのみ評価されます。フォーミュラ パラメーターは表示されますが、このダイアログ内では編集することができません。

28.16.6 インポート/参照

「**インポート/参照**」コマンドを選択すると、インポートするスキーマや参照するリンクベースを指定することができます(*以下のスクリーンショットを参照*)。



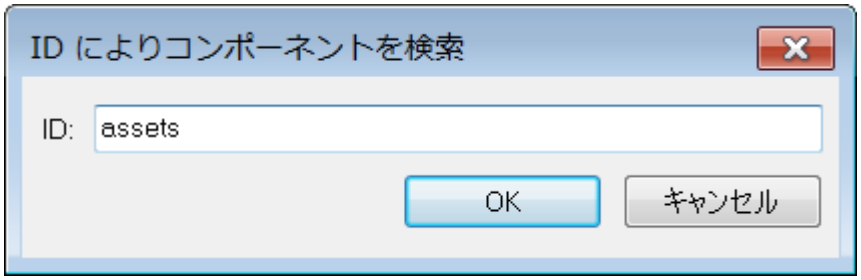
ダイアログには以下のラジオボタンオプションが用意されています。

- *標準タクソノミのインポート* :このオプションにより、US-GAAP タクソノミまたはIFRS タクソノミを素早く正確にインポートすることができます。コンボボックスのドロップダウンメニューから目的の標準タクソノミを選択して、次へをクリックしてください。選択内容に対応したダイアログが表示され、目的のタクソノミに対する情報を指定することができます。タクソノミの作成方法については、新規タクソノミの作成[741]を参照ください。
- *任意のタクソノミをインポート(参照スキーマ)* :このオプションにてタクソノミファイル(.xsdファイル)の場所を指定することで、任意のタクソノミをインポートすることができます。タクソノミ
- *参照リンクベース* :タクソノミに加えるリンクベースを指定することができます。リンクベースファイルの場所を指定して、リンクベースファイルの場所を指定して、完了をクリックしてください。タクソノミ内にリンクベースファイルへの参照が作成されます。新たに参照されたリンクベースのリレーションシップタイプは、ファイル名を右クリックして、「リンクベースの種類を設定」[750]コマンドにより指定することができます。

```
<xsl:template match="*">
   <xsl:copy>
      <xsl:copy-of select="@*[not(.='')]"/>
      <xsl:apply-templates/>
   </xsl:copy>
</xsl:template>
```

28.16.7 ID を使用してコンポーネントを検索する

大きなフォーミュラを使用するタクソノミ、または、テーブルリンクベースでは、同じ型の複数のコンポーネントを含む、フォーミュラまたはテーブルリンクベースがID 別にコンポーネントを検索する際に役に立つ可能性があります。メニューコマンド **XBRL | Id 別にコンポーネントを検索** により ID 別に検索することができます。コマンドをクリックすると検索する ID を尋ねるダイアログが表示されます（*下のスクリーンショット*）。



「OK」をクリックして検索を開始します。

28.16.8 ドキュメンテーションの生成

「XBRL | ドキュメンテーションの生成」コマンドにより、現在開かれている XBRL タクソノミの詳細なドキュメンテーションを生成することができます。ドキュメンテーションの出力は、HTML、MS W ord、RTF、または PDF により行うことができます。このコマンドにより生成されたドキュメンテーションは、Altova から許可を得ることなく、自由に改変、使用することができます。ドキュメンテーションの生成は、（ドキュメンテーションの生成コマンドを選択すると表示される）XBRL ドキュメンテーションダイアログにて選択されたコンポーネントに対して行われます。関連するコンポーネントは画面上の出力にてハイパーリンク化され、コンポーネント間のナビゲーションを簡単に行うことができるようになります。ドキュメンテーションの生成に関する様々なオプションについては、ドキュメンテーション オプション[1398] のセクションを参照ください。

メモ: MS Word 形式でドキュメンテーションの生成を行うには、（バージョン 2000 以降の）MS Word がインストールされていなければなりません。

ドキュメンテーションの生成には、XMLSpy の固定フォーマット、または StyleVision SPS のデザインを使用することができます。StyleVision の SPS を使用することで、生成されたドキュメンテーションのデザインをカスタマイズすることができます。ドキュメンテーションのカスタマイズ方法については、ユーザー定義デザイン[1400] のセクションを参照ください。

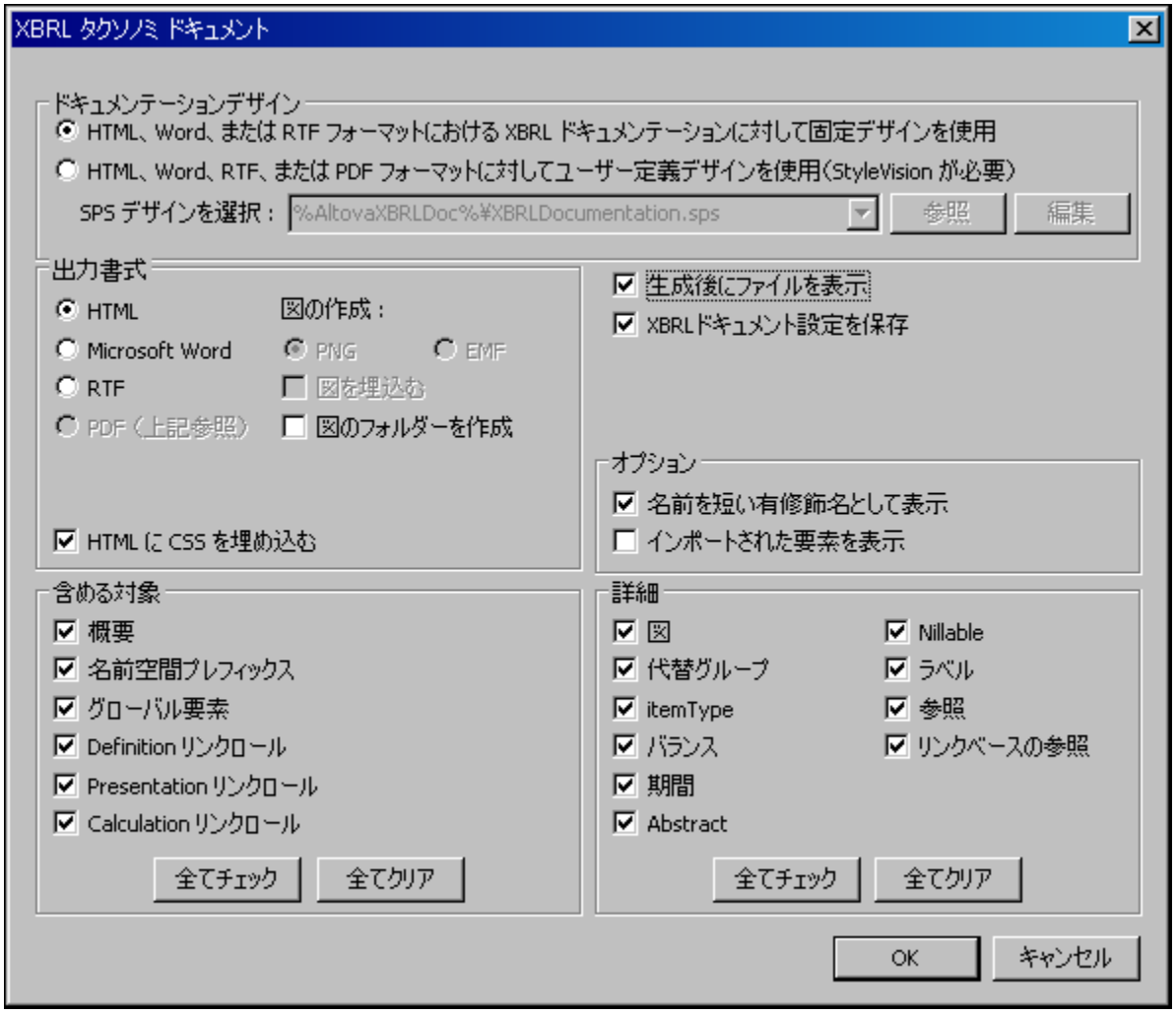
メモ: SPS によるスキーマドキュメンテーションの生成には、お使いのコンピューターに StyleVision をインストールする必要があります。

28.16.8.1 ドキュメンテーションオプション

「XBRL | ドキュメンテーションの生成」コマンドにより、XBRL タクソノミドキュメンテーションダイアログが表示され、ドキュメンテーションに関するオプションを選択することができます（以下のスクリーンショットを参照）。

ダイアログのドキュメンテーションデザインペインでは、ドキュメンテーションの生成にXMLSpy の固定デザインを使用するのか、StyleVision SPS により作成されたカスタマイズデザインを使用するのか選択することができます。希望するオプションを選択してください。PDF 出力は、StyleVision SPS により生成されたドキュメンテーションでしか利用することはできず、固定デザインを使ったドキュメンテーションの生成では利用できないということに注意してください。ユーザー定義デザインの使用方法については、ユーザー定義デザイン 1400 のセクションを参照ください。

メモ SPS によるスキーマドキュメンテーションの生成には、お使いのコンピューターにStyleVision をインストールする必要があります。



ドキュメンテーションの生成 コマンドをクリックすることで、以下の特徴を持つXBRL タクソノミドキュメンテーションダイアログが表示されます:

- 目的のフォーマットを出力形式ペインにて選択します。HTML、Microsoft W ord、RTF、またはPDF から選択を行うことができます。（PDF 出力フォーマットは、StyleVision SPS を使ったドキュメンテーションの生成でのみ選択可能になります）。「OK」をクリックすると、出力ファイル名および保存先の場所を問われます。
- Microsoft W ord ドキュメントによる生成は、固定デザインを使用した際に .docx 拡張子を持つファイルが生成され、StyleVision SPS. を使用した生成では、 .docxファイル拡張子が使用されます。MS W ord フォーマットでドキュメントを生成する場合、MS W ord（バージョン2000 またはは以降）がインストールされている必要があります。
- HTML 出力の場合、CSS スタイル定義を、独立した CSS ファイルに収めるか、HTML ファイル（の <head> 要素）内部に収めるか指定することができます。独立した CSS ファイルが作成される場合、HTML ファイル

と同じ名前で、.css 拡張子を持ったファイルが作成されます。必要に応じて *HTML に CSS を埋め込む* オプションにチェックを入れてください。

- Microsoft Word、RTF、PDF オプションでは、*図を埋め込む* オプションが有効になります。このオプションが有効になっている場合、PNG または EMF 形式でダイアグラムが結果ファイル内部に埋め込まれます。オプションが無効になっている場合、ダイアグラムは PNG または EMF ファイルとして作成され、オブジェクトリンクにより結果ファイルにて表示されます。
- 出力ファイルが HTML の場合、全てのダイアグラムはドキュメントから独立した PNG ファイルとして作成されます。*図のためにフォルダーを作成* チェックボックスにチェックを入れることで、HTML ファイルと同じ場所にフォルダーが作成され、その中に PNG ファイルが保存されます。このフォルダーは HTML ファイル名 _diagrams という名前になります。*図のためにフォルダーを作成* チェックボックスにチェックが入っていない場合、HTML ファイルと同じ場所に PNG ファイルが保存されます。
- 含める対象ペインでは、ドキュメンテーションに含めるアイテムを選択することができます。*概要* オプションはファイルの上でコンポーネント型別に整理されたすべてのコンポーネントをリストします。**全てチェック** ならびに **全てクリア** ボタンにより、ペインにある全てのオプションを全て選択、または選択解除することができます。
- 詳細ペインでは各コンポーネントに含めることのできるアイテムの詳細が表示されますドキュメンテーションに含めるアイテムを選択してください。**全てチェック** ならびに **全てクリア** ボタンにより、ペインにある全てのオプションを全て選択、または選択解除することができます。*ジェネリックなリンクロール* がインクルードペイン内でチェックされているとメッセージチェックボックスが有効化されます。*グローバル要素* または *ジェネリックリンクロール* がインクルードペインでチェックされていると他のすべてのチェックボックスは有効化されます。
- *結果ファイルの表示* オプションは、全ての出力オプションに対して有効になります。このオプションが有効になっていると、生成されたファイルが（HTML 出力の場合は）ブラウザービュー、（MS Word 出力の場合は）MS Word、（RTF ならびに PDF 出力の場合は）.rtf ならびに .pdf ファイルのデフォルトアプリケーションにて表示されます。
- オプションペインでは、(i) 要素名を短縮してプレフィックス（短い修飾名）で表示するか、（名前空間全体とともに）展開されたかたちで表示するか、(ii) インポートされた要素も表示するかを指定することができます。

パラメーター値

StyleVision SPS に複数のパラメーター定義が含まれる場合、**OK** をクリックした後に SPS にて定義されている全てのパラメーターがダイアログにて表示されます。このダイアログでパラメーター値を入力することで、SPS に割り当てられたデフォルトのパラメーター値をオーバーライドすることができます。

28.16.8.2 ユーザー定義デザイン

標準で使用される XMLSpy の固定デザインの代わりに、カスタマイズされたデザインを XBRL タクソノミドキュメンテーションの生成に使用することができます。カスタマイズされたデザインは、出力ドキュメントのデザインテンプレートとなる StyleVision SPS により作成されます。

SPS の作成

StyleVision Power Stylesheet（または SPS）は Altova の StyleVision を使って作成することができます。XBRL タクソノミドキュメンテーションの生成に使用される SPS は、XBRL タクソノミドキュメンテーションの構造を指定する XML スキーマをベースに作成する必要があります。このスキーマは、XBRLDocumentation.xsd という名前で XMLSpy パッケージに同封されており、C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2021\Documentation\XBRL のフォルダーに収められています。

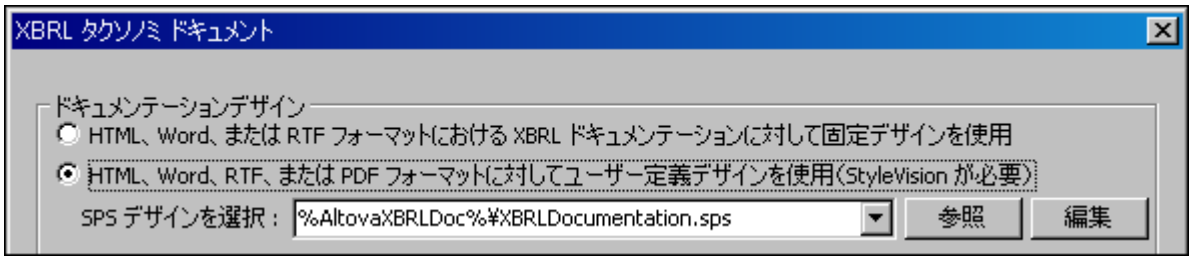
StyleVision にて SPS デザインを作成すると、XBRLDocumentation.xsd スキーマから得られたノードがデザインテンプレート、スタイル、プロパティに配置されます。リンクやテーブル、イメージといったその他のコンポーネントを SPS デザインへ追加することもでき、この場合、出力ドキュメント全体を SPS にてデザインすることになります。StyleVision における SPS デザインの作成方法については、StyleVision ユーザーマニュアルを参照ください。

SPS を使った XBRL タクソノミドキュメンテーションの生成では、デザイン全てをコントロールすることができます。PDF 出力フォーマットによる XBRL タクソノミドキュメンテーションの生成は、ユーザー定義の SPS でしか利用できず、XMLSpy の固定デザインでは PDF 出力を行えないということにも注意してください。

XBRL タクソノミドキュメンテーションの生成にSPS を使用するよう選択

SPS が作成されると、XBRL タクソノミドキュメンテーションの生成に使用することができます。XBRL タクソノミドキュメンテーションの生成に使用するSPS は、(「**XBRL | ドキュメンテーションの生成**」コマンドにより表示される)XBRL タクソノミドキュメンテーションダイアログにて選択することができます。このダイアログの**ドキュメンテーションデザイン**ペインにて、*ユーザー定義デザイン*ラジオボタンを選択してください(以下のスクリーンショットを参照)。**参照**ボタンをクリックして、目的のSPS ファイルを指定することができます。「**OK**」ボタンをクリックすると保存ダイアログが表示されるので、出力ファイルが保存されるフォルダーと、ファイル名を入力してください。

メモ: SPS ファイルでは、ベースとなっているスキーマのXBRLDocumentation.xsd を正しく指定する必要があります(上記を参照)。



XMLSpy にはXBRL タクソノミドキュメンテーションの生成に対応したSPS デザインが同封されています。このファイルは、`XBRLDocumentation.sps` という名前で `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2011\Documentation\XBRL\` フォルダー以下に配置されています。SPS ファイルは、最近参照したその他のSPS ファイルとともに、**ユーザー定義デザイン**オプションのコンボボックスに表示されます。

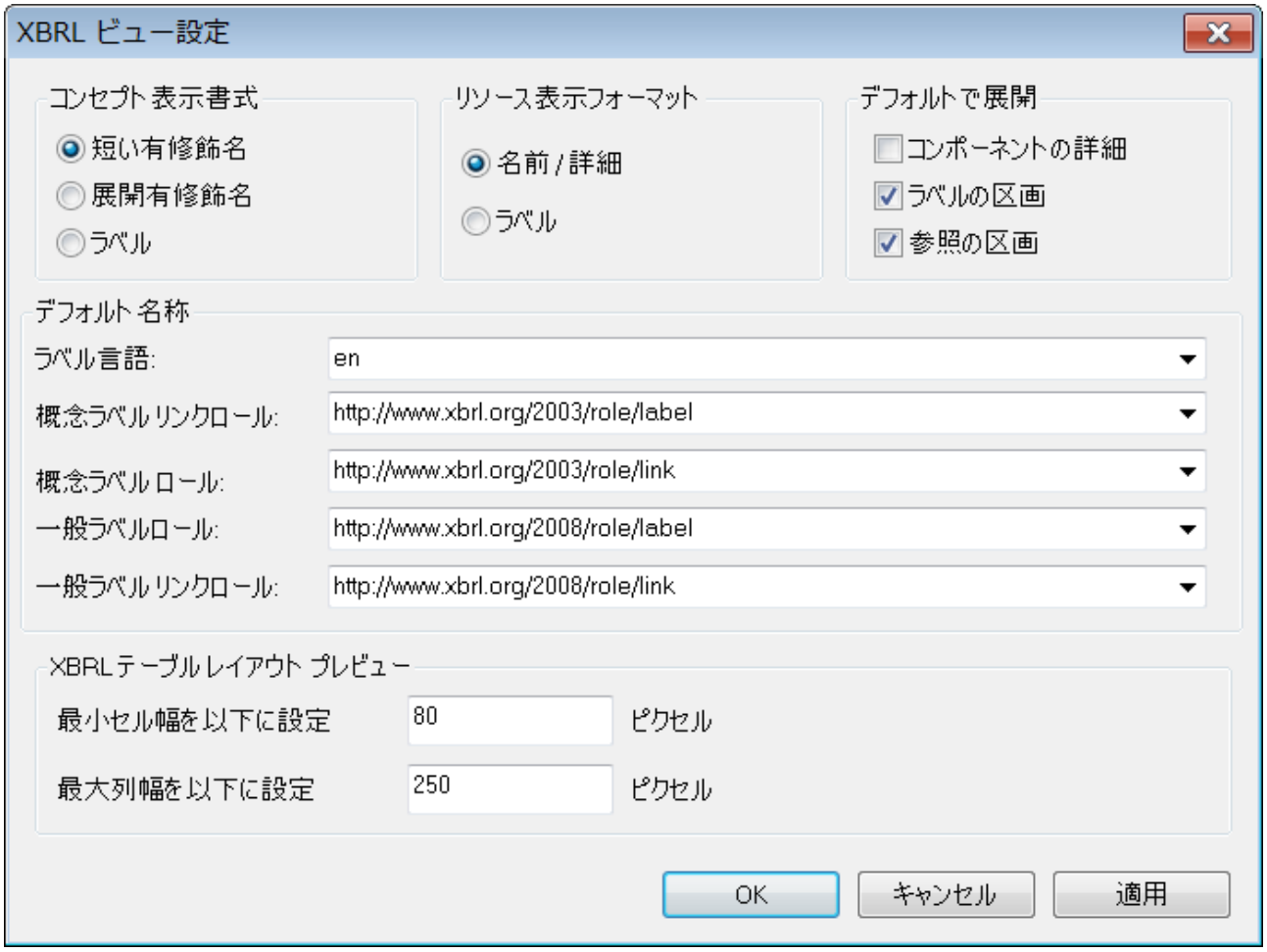
ドキュメンテーションデザインペインにある**編集**ボタンをクリックすると、StyleVisionが起動され、選択されたSPS がStyleVision ウィンドウにて開かれます。StyleVision にて生成されるドキュメントのプレビューを行うには、作業用 XML ファイルが必要になります。このため、nanonull.xml という名前のサンプルXML ファイルが製品には同封されており、以下のフォルダーに配置されます::

```
C:\Documents and Settings\<username>\My
Documents\Altova\XMLSpy2021\Documentation\XBRL\SampleData
```

メモ: SPS を使った XBRL タクソノミドキュメンテーションの生成を行うには、お使いのコンピューターに StyleVision をインストールする必要があります。

28.16.9 ビューの設定

ビューの設定コマンドにより、XBRL ビュー設定ダイアログが表示され、XBRL ビューのデフォルト設定を指定することができます(以下のスクリーンショットを参照)。



以下の設定を行うことができます:

- コンセプト名の表示 は短い、または、展開された修飾名、または、ラベルに設定することができます。これらの設定は、メインウィンドウと詳細入力ヘルパーに適用されますが、グローバル要素ヘルパーには適用されません。グローバル要素入力ヘルパー内のアイテムの表示は、入力ヘルパーのメニューバー[251]内で定義されます。
- リソース表示フォーマット: フォーミュラとテーブルタブ内では、リソースは名前またはラベルで表示されることができます。(名前が割り当てられていない場合、リソースの説明が使用されます)。
- デフォルトで展開 : メインウィンドウにおいて、要素の詳細、ラベルの区間、そして参照の区間をデフォルトで展開された状態にしておくことができます。ラベルや参照の区間がデフォルトで展開されたかたちで表示されるようになっている場合、展開されたボックスは要素の詳細が(デフォルトまたは手動で)展開されたときにしか見えないということに注意してください。これらのデフォルト設定は、表示が再描画された時に有効になります。
- デフォルトのラベル ラベルが定義されていない場合、使用するデフォルトの言語とデフォルトのラベルのロールを指定します。各プロパティのためのコンボボックス使用することのできる値のリストを表示します。
- XBRL テーブルレイアウト プレビュー: 列の幅の最小値と最大値をピクセルで設定することができます。

(C) 2015-2021 Altova GmbH

28.16.10 MapForce で DB、Excel、CSV から XBRL を生成

このコマンドにより、現在アクティブなタクソノミをベースにしたXBRL インスタンスファイルの生成を開始します。インスタンスファイルのデータは、MS Excel データシート、データベース、またはCSV ファイルから入手されます。XBRL インスタンスファイルは、お使いのコンピューターにインストールされたAltova のMapForce プログラムにより生成されます。このコマンドにより、Altova MapForce が起動され、タクソノミファイルがロードされます。インスタンスファイルの出力に必要なソースデータファイルを、視覚的なデザインにて指定します。

28.16.11 StyleVision で HTML/PDF/Word として XBRL を表示

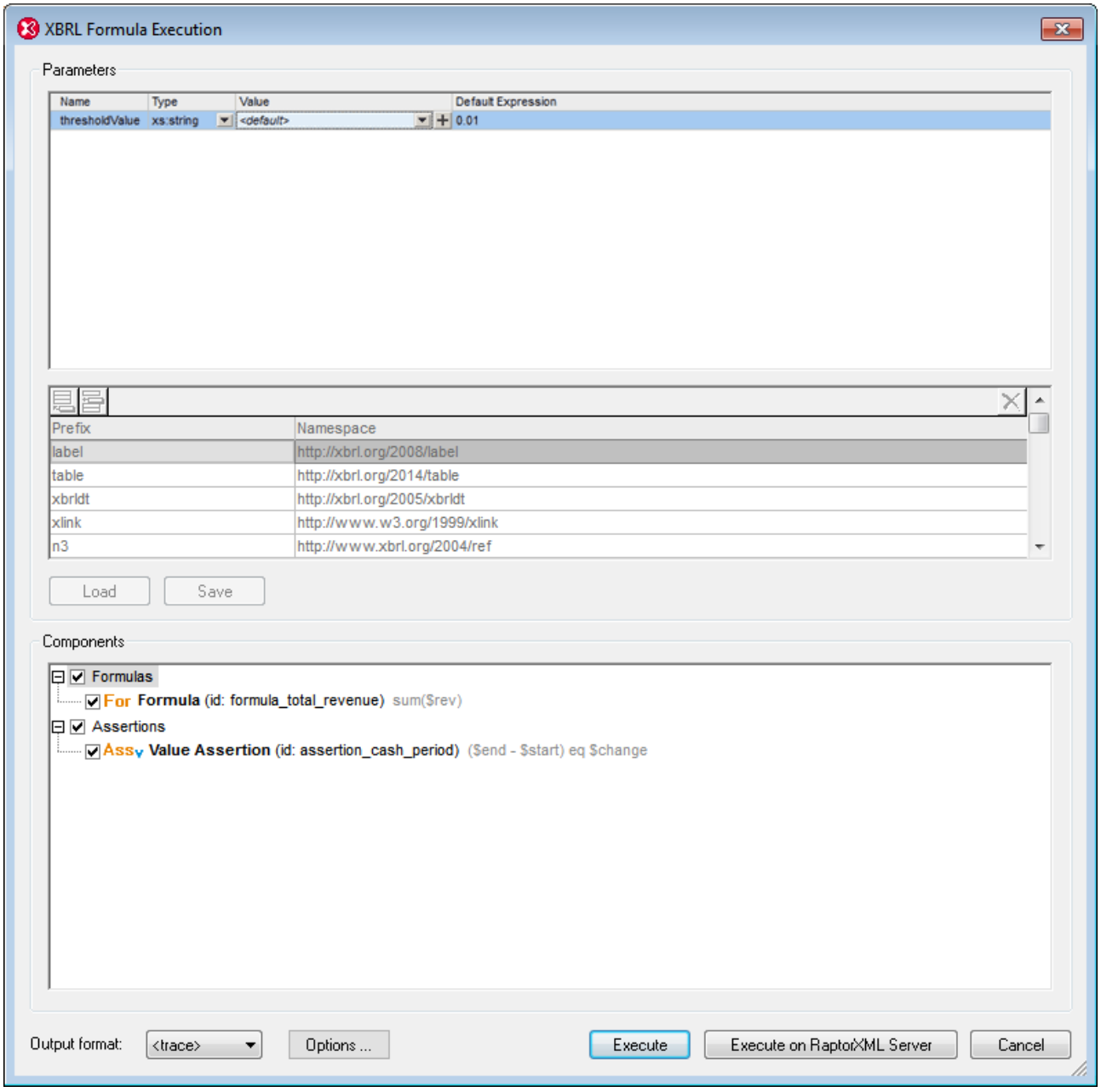
このコマンドにより、現在アクティブなタクソノミを Altova StyleVision へロードして、タクソノミをベースにしたレポートのデザインを生成することができます。このコマンドを使用するには、お使いのコンピューターにAltova StyleVision がインストールされていなければなりません。

28.16.12 (サーバーでの)フォーミュラの実行

フォーミュラの実行 コマンドは、XBRL インスタンスドキュメントがテキストビューまたはグリッドビュー内でアクティブなドキュメントの場合、有効化されます。XBRL インスタンスファイルに関連付けられたDTS 内で定義されたフォーミュラおよび/またはアサーションを実行します(Discoverable Taxonomy Set の省略であるDTS は、タクソノミのコレクションです)。フォーミュラは、XBRL インスタンスファイル内のデータを使用して検証されます。フォーミュラは、XBRL インスタンスファイル内の出力です。アサーション個別に評価され、結果はJSON またはXML ファイルの出力です。

「サーバー(ハイパフォーマンス)上でフォーミュラを実行する」コマンドは、フォーミュラを実行するために関連したRaptorXML+XBRL Server 957 を使用します。コマンド「ツール | Raptor サーバーの管理」1434 を使用してRaptorXML+XBRL サーバーをセットアップします。

DTS 内で定義されるフォーミュラまたはアサーションは存在しません。これを通知するメッセージが表示されます。DTS 内に有効なフォーミュラまたはアサーションが存在する場合、XBRL 式実行 ダイアログ(*下のスクリーンショット*) がポップアップします。



パラメーター
パラメーターがDTS 内で定義されている場合、パラメーター ペイン内に各パラメーターが表示され、その値を入力することができます。パラメーター名は、読み取り専用です。必須のパラメーターは感嘆符と共に表示され、パラメータに値が割り当てられるまで、「OK」ボタンは無効化されています。任意のパラメーターにはデフォルトの値が存在します。必要な型が指定されている場合、型が表示されます。複数の値を必要とするパラメーターは、新規の値をクリックして追加することのできる + アイコンと共に表示されています。値の無い任意のパラメーターは実行のためにエンジンにパスされることご注意してください。デフォルトの値は、読み取り専用で、ユーザーが値を入力しない場合、実行されます。

名前空間マッピング
このテーブルは、パラメーターと型のQNames 内で使用されているプレフィックスを定義します。パラメーターの評価内での使用のための追加の名前空間は、ここで定義される場合があります。

(C) 2015-2021 Altova GmbH

パラメーターの保存とロード
名前空間マッピングを含むパラメーターの設定は、「保存」ボタンをクリックすることにより、JSON または XML フォーマットに保存されます。ファイルのフォーマットは、ファイルに与えられているファイル拡張子により決定されます。値を持たない任意のパラメーターは、保存されません。保存されると、「ロード」ボタンを使用して、パラメーターファイルをダイアログにロードすることができます。

XBRL 処理命令
「オプション」ボタンにより（重複するファクトを自動的に無視する）重複の回避をオンに切り替え、テーブル生成オプションを設定することのできる XBRL 処理オプションダイアログ[1411]が開かれます。

コンポーネント

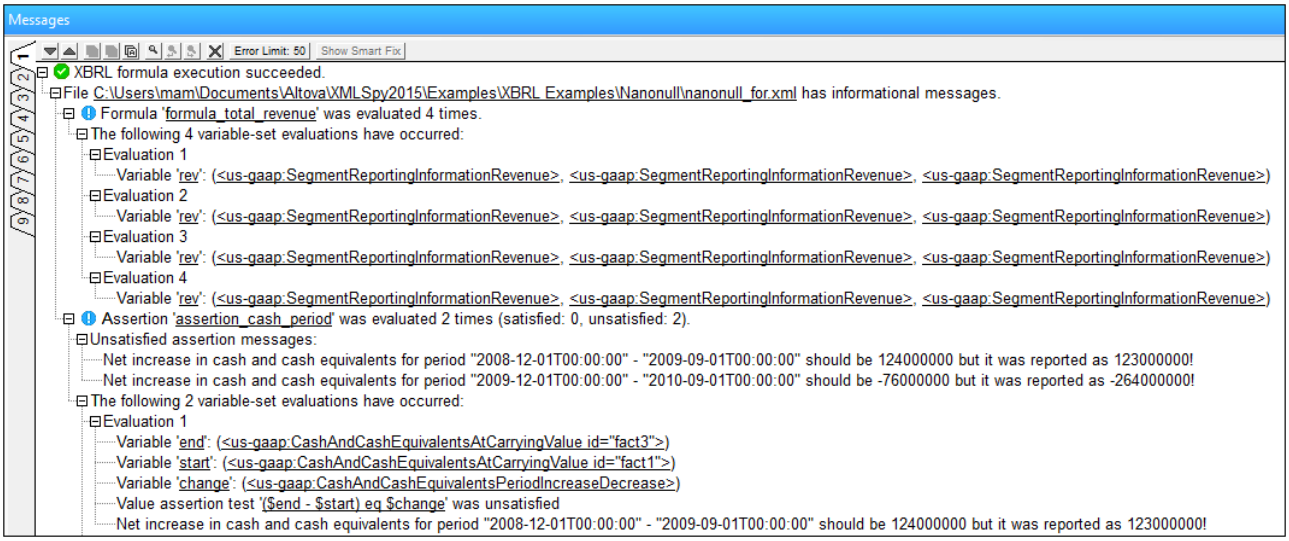
このペインには、実行される フォーミュラ および/または アサーション コンポーネントの選択を許可するツリービューが含まれています。各アイテムはアイコンとその説明を表示します。使用できる場合はID と式も表示されます。実行のためのコンポーネントを選択する場合は、チェックボックスをチェックします。アサーション 実行の出力は、JSON または XMLです。*出力フォーマット* コンボボックスから出力フォーマットを選択します。XBRL フォーミュラ 実行の書式は常に、XML です。

実行

XMLSpy の内部エンジン または、Altova の RaptorXML Server[957] を使用して実行がおこなわれるかを選択します。実行エラーの場合、エラーメッセージが出力ウィンドウに表示されます。それ以外の場合は、成功のメッセージが表示されます。出力ファイル、`asssertions-ouput-file.xml/json` および/または `formula-output-file.xml`, が新規ドキュメントウィンドウで開かれますが、ディスクには保存されません。ディスク上の希望する場所にファイルを明示的に保存する必要があります。

トレース

出力フォーマット コンボボックス内で `<trace>` を選択すると、（ダイアログの左下）すべての「変数セットの評価」ための追加デバッグ情報がフォーミュラの実行中に収集され、メッセージウィンドウに表示されます（*下のスクリーンショットを参照*）。トレースは、その評価内の実際の変数の割り当てこの時点の各フォーミュラ/アサーションのための個別の変数のセットの評価をリストします。変数 *レファレンスインスタンスファクト* の場合、値をクリックすると、インスタンス内の対応するファクト 要素に移動することができます。フォーミュラ/アサーション または変数名をクリックすると、フォーミュラ リンクベース ファイル内の対応する定義に移動することができます。検証アサーションの場合、評価ステップに生成されたアサーションメッセージも表示されます。



メモ: トレースの実行は、メモリおよび計算スピードを大幅に必要とする場合があります。大きな XBRL インスタンスを使用する場合、アサーションは、制限無く評価され、各評価は40 個の変数間での値を保管することができます。ですから、この機能 は、縮小さ

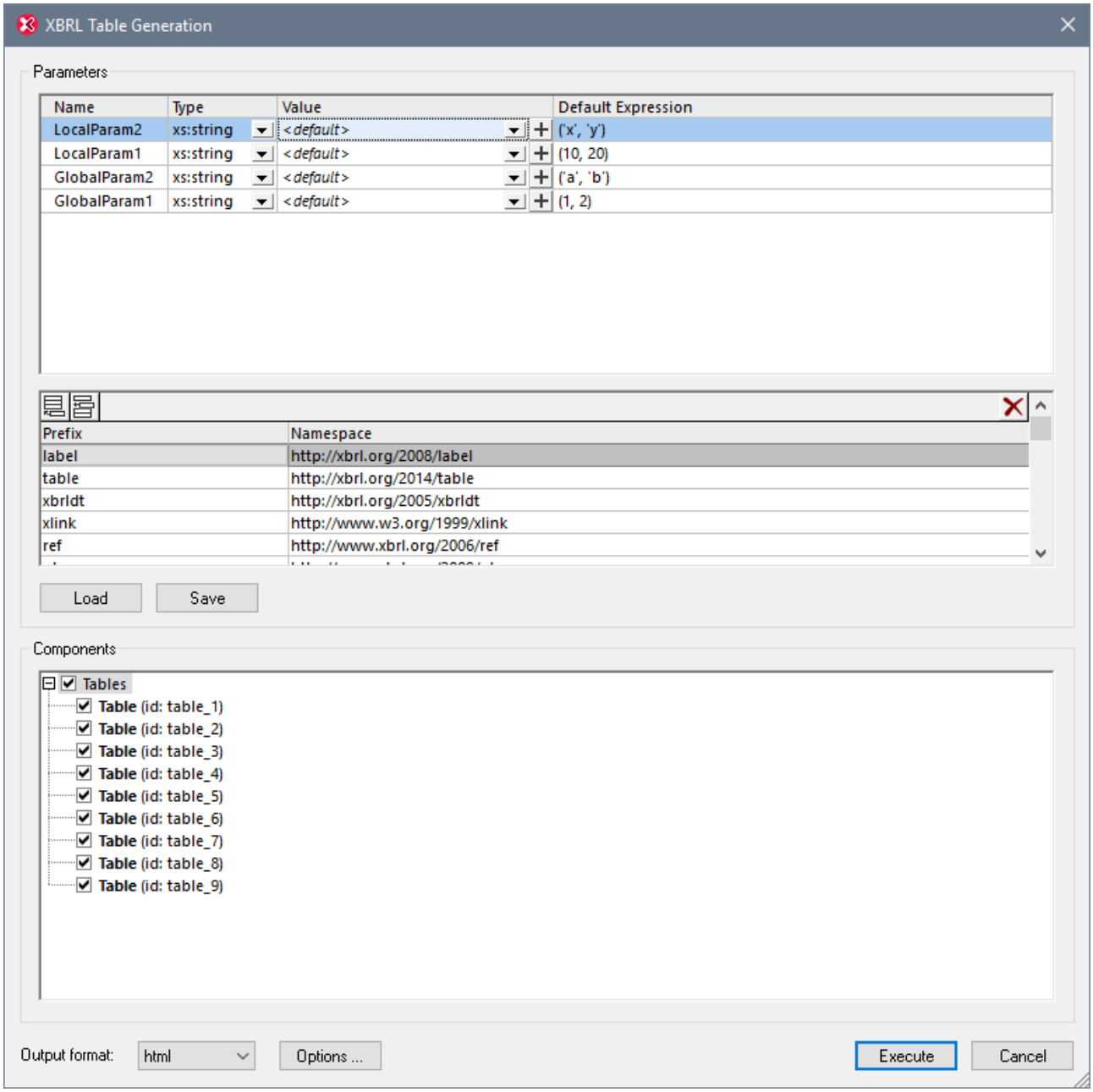
れたサンプルをデバッグする場合に使用するべきです。それ以外の場合、実行の速度は遅く、XMLSpy がメモリ不足になる可能性があります。このため、各トレースは、ハードコードされ、評価は1000個までに制限されています。

28.16.13 (サーバーでの)テーブルの生成

テーブルの生成 コマンドは以下の場合有効化されます(i) XBRL *インスタンス*ドキュメントは、テキストビューまたは グリッドビュー内のアクティブなドキュメントです 、または(ii) XBRL *タクソノミ*は、XBRL ビュー、テキストビュー、グリッドビュー、または スキーマビュー内のアクティブなドキュメントです。*アクティブ*なドキュメントに関連したDTS 内で定義されているXBRL テーブルを含むXML またはHTML ドキュメントを生成します。、(*タクソノミのコレクション*であるDTS (Discoverable Taxonomy Set)) XBRL *インスタンスファイル*の場合、テーブルは、XBRL *インスタンスファイル*内のデータを使用して生成されます。

「**サーバー(ハイパフォーマンス)上でフォーミュラを実行する**」コマンドは、テーブルを生成するために関連したRaptorXML+XBRL Server 957 を使用します。コマンド 「**ツール| Raptor サーバーの管理**」1434 を使用してRaptorXML+XBRL サーバーをセットアップします。

DTS 内でテーブルが定義されていない場合、これを通知するメッセージが表示されます。有効なテーブル定義がDTS 内に、存在する場合、XBRL テーブル生成 ダイアログ(*下のスクリーンショット*) がポップアップします。



パラメーター

パラメーターがDTS 内で定義されている場合、パラメーターペイン内に各パラメーターが表示され、その値を入力することができます。パラメーター名は、読み取り専用です。必須のパラメーターは感嘆符と共に表示され、パラメータに値が割り当てられるまで、「OK」ボタンは無効化されています。任意のパラメーターにはデフォルトの値が存在します。必要な型が指定されている場合、型が表示されます。複数の値を必要とするパラメーターは、新規の値をクリックして追加することのできる + アイコンと共に表示されています。値の無い任意のパラメーターは実行のためにエンジンにパスされることに注意してください。デフォルトの値は、読み取り専用で、ユーザーが値を入力しない場合、実行されます。

名前空間マッピング

このテーブルは、パラメーターと型のQNames 内で使用されているプレフィックスを定義します。パラメーターの評価内での使用のための追加の名前空間は、ここで定義される場合があります。

パラメーターの保存とロード
名前空間マッピングを含むパラメーターの設定は、「**保存**」ボタンをクリックすることにより、JSON または XML フォーマットに保存されます。ファイルのフォーマットは、ファイルに与えられているファイル拡張子により決定されます。値を持たない任意のパラメーターは、保存されません。保存されると、「**ロード**」ボタンを使用して、パラメーターファイルをダイアログにロードすることができます。

XBRL 処理命令
「**オプション**」ボタンにより（重複するファクトを自動的に無視する）重複の回避をオンに切り替え、テーブル生成オプションを設定することのできる XBRL 処理オプションダイアログ[1411] が開かれます。

コンポーネント

このペインには、実行されるテーブルにコンポーネントの選択を許可するツリービューが含まれています。各アイテムは、アイコンとその詳細、また、存在する場合は ID を表示します。実行のためのテーブルにコンポーネントを選択するには、チェックボックスをチェックします。XML または HTML フォーマットで出力が生成されます。出力フォーマットを*出力フォーマット* コンボボックスから選択します。「**オプション**」ボタンをクリックして、XBRL テーブル生成オプションと重複の無視を指定する XBRL 処理オプションダイアログ[1411] を表示します。優先されるラベルオプションを使用できる場合、これらが使用されます。それ以外の場合、XBRL 処理オプションダイアログ[1411] 内で指定されるデフォルトが使用されます。

実行

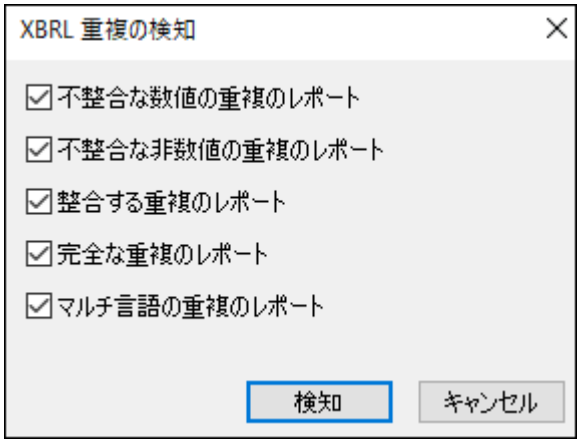
実行エラーの場合、エラーメッセージが出力ウィンドウに表示されます。それ以外の場合は、成功のメッセージが表示されます。出力ファイル `table-ouput-file.xml/html` が新規ドキュメントウィンドウで開かれますが、ディスクには保存されません。ディスク上の希望する場所にファイルを明示的に保存する必要があります。

28.16.14 （サーバー上での）重複の検知

重複の検知 と **サーバー上で重複の検知（ハイパフォーマンス）** コマンドはテキストビューまたはグリッドビュー内で XBRL インスタンスドキュメントがアクティブなドキュメントである場合有効化されます。これらのコマンドは、重複ファクトのためのインスタンスドキュメントをチェックし、メッセージウィンドウ内に重複をレポートします。 **サーバー上で重複の検知（ハイパフォーマンス）** コマンドは、関連した RaptorXML+XBRL Server[957] を使用して、重複をチェックします。コマンド、「**ツール | Raptor サーバーの管理**」[1434] を使用して、RaptorXML+XBRL Server をセットアップします。どちらかのコマンドをクリックすると、XBRL 重複検知 ダイアログ（*下のスクリーンショップ*）が表示されます。このダイアログ内で、検知する重複の型を選択することができます。

XBRL とインライン XBRL 1.0 内での重複ファクトの扱い 仕様内で複数の異なる型について詳しく説明されています。ここでは手短に説明されています:

- *完全な重複* は名前、コンテキスト、値が同じ重複です。
- *整合する重複* は四捨五入のために指定されている少数点までの同じ値を持つ重複数値ファクトです。
- *マルチ言語の重複* は複数の言語内で同じファクトを繰り返します。
- *不整合な重複* は XBRL 2.1 仕様 内で設定された重複ファクトのための条件を満たします。これらは完全な重複ではなく、整合した重複、または、マルチ言語の重複です（例えば、異なる数値を持つ数値の重複）。



28.16.15 XULE の実行

XULE の実行 コマンドはXBRL *インスタンス*ドキュメント上でXULE ルールを実行します。コマンドは以下のケース内で有効化されます:

- XULE ドキュメントが*アクティブ*なドキュメントの場合。XULE ドキュメントには `.xule` のファイル拡張子が伴います。この場合、XULE ドキュメントが処理されるXBRL *インスタンス*を選択するようにプロンプトされます。
- (通常 `.xbrl` または `.xml` ファイル拡張子を持つ) XBRL *インスタンス*ドキュメントが*アクティブ*なドキュメントの場合。この場合、使用するXULE ドキュメントを選択するようにプロンプトされます。

メモ XULE ドキュメントとXBRL *インスタンス*ドキュメントがXMLSpy プロジェクト 950 の一部の場合、XMLSpy プロジェクトのプロパティ 1194 内のターゲット XBRL *インスタンス*ファイルを指定することができます。XULE ファイルを右クリックし、「XULE の実行」コマンドを選択すると、XULE 実行*のための*ターゲットとして指定されているXBRL ドキュメント上で実行されます。

例外オプション

XULE 出力

XULE 実行は以下に送信されます: (i) *メッセージウィンドウ*、または、(ii) 新規のXMLSpy ウィンドウ内で表示され、メモリ内で一時的に保管される新規のドキュメント。このドキュメントは「ファイル | 名前を付けて保存」 1138 コマンドを使用して保管することができます 。出力がウィンドウ、または新規のドキュメントに移動するかを指定するにはXBRL XULE オプションタブ 1488 (「ツール | オプション | XBRL | XULE」) 内で希望するオプションを選択します。*下のスクリーンショットを参照してください*。

重複ファクト
重複ファクトは同じファクトへの複数の参照を指しています。XBRL XULE オプションタブ[1488] 内の *複製ファクトを無視* オプションをチェック数ごとにより結果内での重複するファクトを回避することができます(「**ツール | オプション** | XBRL | XULE」)。*上のスクリーンショットを参照してください*。この場合、重複ファクトは一度のみ報告されます。

28.16.16 インライン XBRL の変換

インライン XBRL の変換 コマンドは、テキストビュー、グリッドビュー、または、ブラウザービュー内で、インライン XBRL を含む XHTML ドキュメントがアクティブなドキュメントの場合、有効化されます。このコマンドはアクティブな XHTML ドキュメントからインライン XBRL データを抽出し、抽出されたデータを含む 1 つ、または、複数の XBRL ドキュメントを生成します。生成された XBRL ドキュメントは新しいウィンドウで開かれ、ファイルに保存されます。コマンドが正確に作動するために、インライン XBRL ドキュメントにより参照される全てのリソースは、処理のために使用できる必要があります。

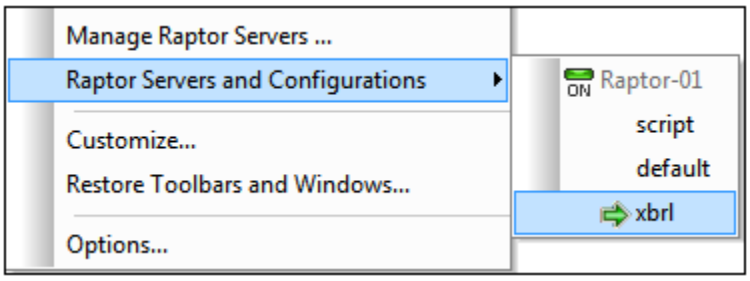
メモ 重複を無視するには、(**XBRL | 処理オプション** メニューコマンドによりアクセスすることのできる) XBRL 処理オプション ダイアログ[1411] 内の *重複するファクトの無視* オプションを選択してください。

28.16.17 サーバー上で EDGAR を検証

サーバー上で EDGAR を検証 (ハイパフォーマンス) コマンドはアクティブな XBRL インスタンスドキュメントを 現在アクティブな RaptorXML+XBRL Server[1438] とアクティブな構成[1438] を使用して検証します。EDGAR を介して検証する場合、Raptor は内部 EDGAR スクリプトを使用して XBRL インスタンスを検証します。[973] コマンドはすぐに検証を実行し、メッセージウィンドウに結果を表示します。

メモ 実際のパフォーマンスは RaptorXML+XBRL Server により検証のために使用される PC プロセッサーコアにより異なります: 使用されるコア数が多いほうが、早く処理を行うことができます。

複数のサーバーで複数の構成を定義した場合、サーバーとそのアクティブな構成として 1 つの構成を選択することができます。アクティブな構成は、以降の検証に使用されます。カーソルを「**ツール | Raptor Servers および 構成**」コマンドにポイントして、追加されたすべてのサーバーとその詳細が含まれるサブメニューから、を含む表示されるサブメニューが表示されます (*下のスクリーンショット参照*)。アクティブな構成と設定するサーバー構成を選択します。したのスクリーンショットでは、`Raptor-01` という名前のサーバーの(緑色の矢印で示されている) `xbrl` 構成がアクティブな構成として選択されています。

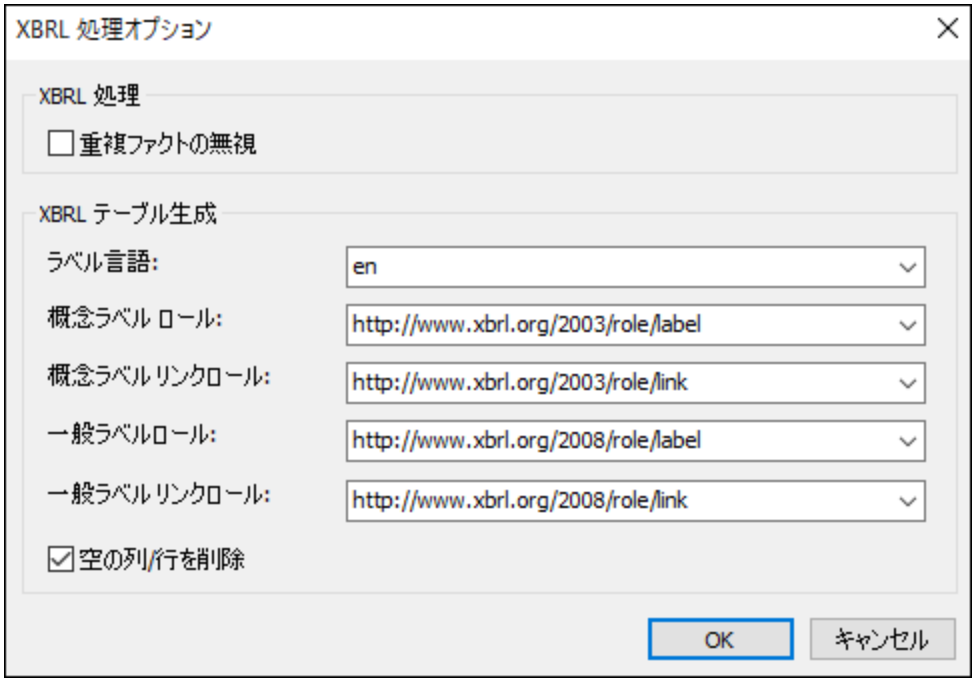


詳細に関しては RaptorXML+XBRL Server の使用方法[957] を説明するセクションを参照してください。

28.16.18 処理オプション

XBRL 処理オプション コマンドは、XBRL 処理オプション ダイアログ (*下のスクリーンショット*) を表示します。ここでは以下を指定することができます:

- 重複ファクトは以下のために無視されます: (i) XBRL フォーミュラの実行、(ii) XBRL テーブル生成、(iii) インライン XBRL 変換
- XBRL テーブル生成のためのラベル設定
- XBRL テーブル生成中に空の行と列が削除されます。



メモ XBRL 処理オプション ダイアログには、XBRL フォーミュラ実行ダイアログ[1403] と XBRL テーブル生成ダイアログ[1406] からアクセスすることができます。

重複の解除に関するメモ

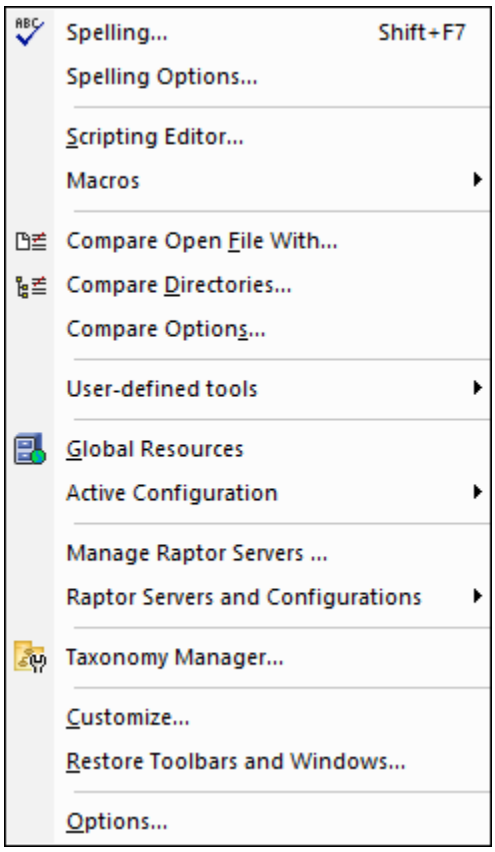
以下の重複の型のみが無視されます: (i) 完全な重複 (ii) 一貫性のある重複。(無視されていない) 選択されている(重複内の)ファクトが最も正確なものです。例えば、最も高い数値の精度を持つ数値のファクトの重複が選択されます。

重複の詳細に関下は、XBRL とインライン XBRL 1.0 内での重複ファクトの扱いの仕様を参照してください。

28.17 ツールメニュー

ツールメニューを使用して以下を行うことができます:

- XML ドキュメントのスペル[1414] チェックをチェックします。
- XMLSpy のスクリプト環境[1505] にアクセスします。フォーム、マクロ、イベントハンドラーを作成、管理、および保管することができます。
- 現在割り当てられているマクロをビュー[1423]します。
- 差異をチェックするために2つのファイルを比較する。
- 差異をチェックするために2つのフォルダーを比較する。
- 外部アプリケーションを使用するカスタマイズされたコマンドにアクセスします。これらのコマンドは、カスタマイズ ダイアログのツールタブ内で作成することができます[1442]。
- グローバルリソースの定義[1433]
- XMLSpy 内のグローバルリソースのためにアクティブな構成を変更します[1434]。
- XML と XBRL の検証と RaptorXML 検証オプションを構成するために[1436] RaptorXML Servers を追加します[1435]。
- アクティブな構成として Raptor Server 構成を選択します[1438]
- XBRL タクソノミパッケージを専用のアプリケーション XBRL タクソノミ マネジャー[1438] を使用して管理する。
- XMLSpy の自身のバージョンをカスタマイズします[1438]: 自身のツールバー、キーボードショートカット、メニューとマクロを定義します。
- グローバルな XMLSpy 設定[1454]を定義します。



28.17.1 スペリング

内蔵の言語辞書を搭載したXMLSpy のスペルチェッカーは、テキストビュー、グリッドビュー、およびAuthentic View. で使用することができます。その他のビューで編集したドキュメントに対してスペルチェックを行いたい場合は、テキストビューまたはグリッドビューへ切り替えた後にスペルチェックを起動してください。例えば、XML スキーマドキュメントをスキーマビューで編集している場合、テキストビューまたはグリッドビューへの切り替えを行い、スペルチェックを行うことができます。

メモ: Altova ソフトウェアに内蔵されている辞書は、Altova の好みにより選択されたものではありません。MPL、LGPL、またはBSD ライセンスといった商用ソフトウェアにおける再配布が認められているライセンス下で利用可能かどうかに依存しております。GPL ライセンスなどの、より強力なライセンス下で配布されています。これらの辞書はhttps://www.altova.com/ja/dictionaries に配置されているインストーラーからご利用いただくことができます。ライセンスと有用性を基に使用する辞書を選択してください。

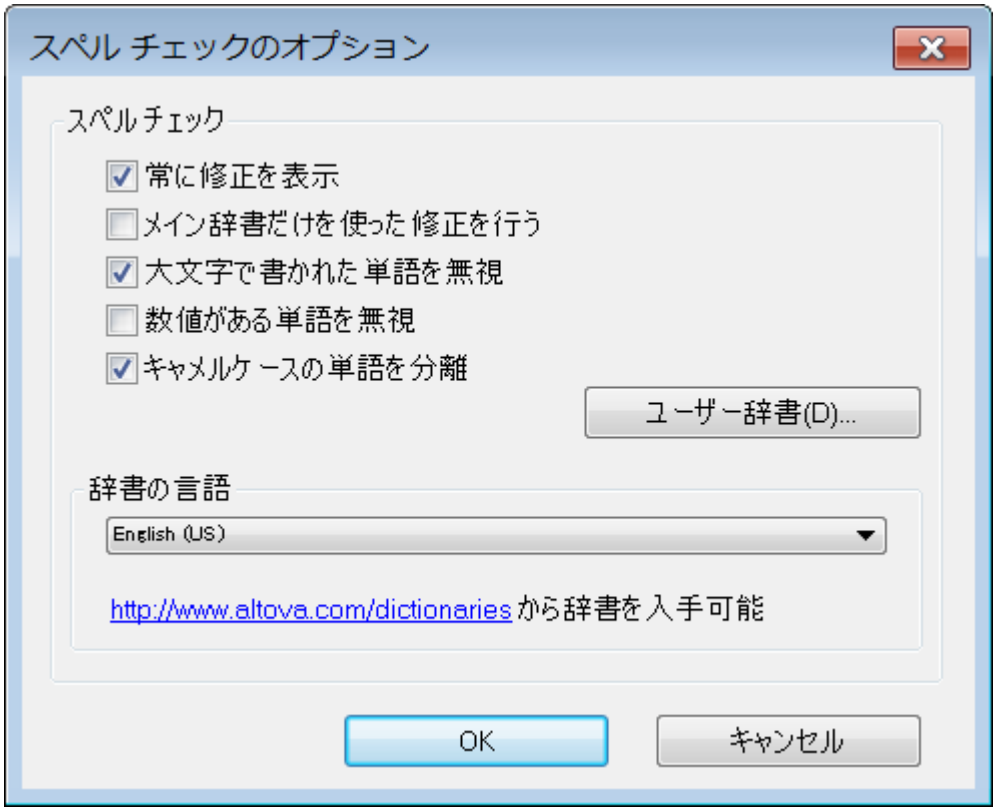
このセクションではスペルチェッカーの使用方法について記述され、以下の3つのサブセクションから構成されます:

- スペルチェッカーの言語を選択 1414
- チェックを行う範囲の定義 1415
- スペルチェッカーを起動する 1416

スペルチェッカーの言語を選択

以下の操作により、スペルチェックを行う言語をセットすることができます:

1. 「ツール| スペルチェックのオプション」メニューコマンドをクリックします。
2. スペルチェックのオプションダイアログにて、辞書の言語コンボボックスのリストからインストールされた辞書を選択します(以下のスクリーンショットを参照)。



3. **OK** をクリックして完了します。

選択された言語の辞書がスペルチェックに使用されます。スペルチェックを行う言語の辞書がまだインストールされていない場合、新たな辞書をダウンロードすることができます。ダウンロードの方法については、スペルチェッカーに辞書を追加[1421]のセクションを追加を参照ください。

チェックを行う範囲の定義

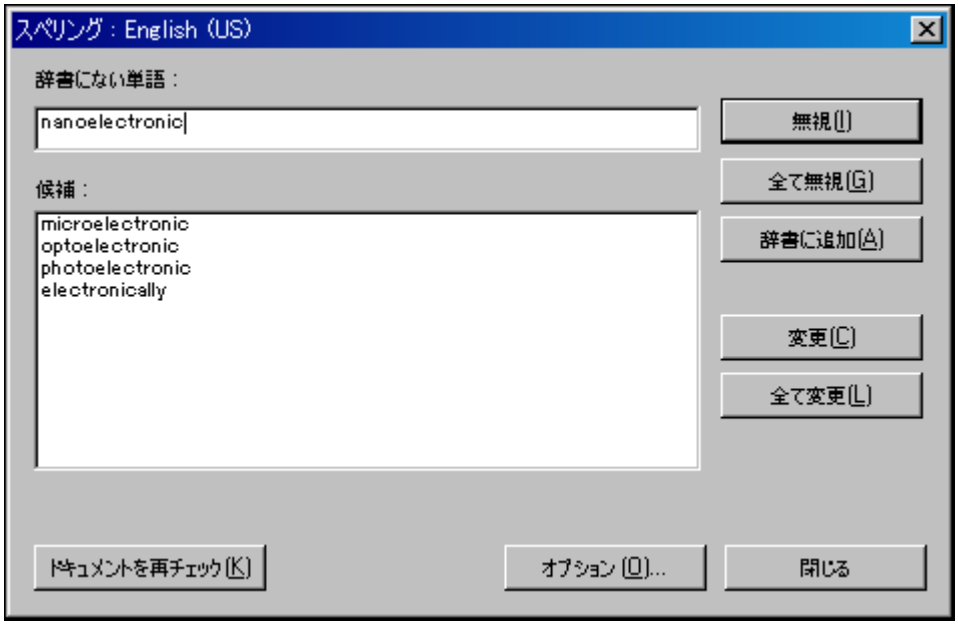
テキストまたはグリッドビューでスペルチェックが実行される場合、チェックを開始する直前にチェックする範囲を定義することができます。コマンド [**ツール | スペリングオプション**] を選択して表示されるXML スペルチェックのオプションのダイアログ[1418] 内で目的の範囲を定義します(以下のスクリーンショットを参照)。

チェックを行う要素や属性を選択できるほか、CDATA セクションやコメントをチェックするか指定することができます。[スペルチェックのオプション](1418) コマンドの記述も参照ください。

スペルチェッカーを実行する

「ツール | スペルチェック」コマンド（Shift + F7）により、現在アクティブになっているXMLドキュメント内における、[定義づけされた範囲](1419)に対してチェックが自動的に始まります。未知の単語が発見されると、*スペリング*ダイアログが表示されます（以下のスクリーンショットを参照）。未知の単語が無い場合、スペルチェックがドキュメントの最後まで行われます。

(C) 2015-2021 Altova GmbH



スペリングダイアログで使用することのできるオプションを以下に記します:

辞書にない単語
このテキストボックスには、選択された言語辞書やユーザー辞書にて発見できなかった単語が表示されます。以下のオプションが利用可能です:

- テキストボックスに表示されている単語をキーボードを使って編集するか、候補 リストから選択することができます。その後変更ボタンをクリックすることで、XML ドキュメント内にある単語が、編集された単語に置き換えられます（候補リストにある単語をダブルクリックすることで、XML ドキュメントの内容が直接置き換わります）。単語が辞書にない単語 テキストボックスに表示されると、XML ドキュメント内の対応する箇所がハイライトされ、ドキュメント内の単語を直接編集することもできます。全て変更をクリックすることで、置換処理がXML ドキュメント全体に対して行われます。
- 変更を行わず、現在ハイライトされている箇所のみ、またはドキュメント全体においてスペルチェッカーの警告を無視することができます。
- 単語をユーザー辞書に追加することで、それ以降のチェックにおいてその単語が正しいものであると認識させることができます。

候補
このリストボックスには、未知の単語に近い（言語辞書ならびにユーザー辞書から得られた）単語が表示されます。リストにある単語をダブルクリックすることで、その単語がドキュメント内へ挿入され、スペルチェック処理が続けられます。

一度無視する
このコマンドにより、現在選択されている未知の単語を無視して、ドキュメントのチェックを続けることができます。同じ単語がドキュメント内で再度発見された場合、その単語が再度表示されます。

全て無視
このコマンドにより、現在の未知単語をドキュメント全体において無視することができます。

辞書に追加
このコマンドにより、未知の単語をユーザー辞書へ追加します。ユーザー辞書へは スペルチェックのオプション[1421] ダイアログにてアクセス、編集を行うことができます。

変更
このコマンドにより、XML ドキュメント内で現在ハイライトされている単語が、(変更された) *辞書にない単語* テキストボックス内の単語に置き換えられます。

全て変更
このコマンドにより、XML ドキュメント内で現在ハイライトされている単語が、ドキュメント全体で(変更された) *辞書にない単語* テキストボックス内の単語に置き換えられます。

ドキュメントを再チェック
ドキュメントを再チェックボタンにより、ドキュメントの先頭から再度チェックを行います。.

オプション
現在使用されているビューにより以下のダイアログボックスが表示されます:

- 現在のビューが Authentic View の場合、スペルチェックのオプション[1418] ダイアログボックスが表示されます。
- 現在のビューがテキストビューまたはグリッドビューの場合 XML スペルチェックのオプション[1418] ダイアログボックスが表示されます。

ダイアログボックスに関する詳細はスペルチェックのオプション[1418]セクションを参照ください。

閉じる
スペリングダイアログボックスが閉じられます。

28.17.2 スペルチェックのオプション

「**ツール | スペルチェックのオプション**」コマンドにより スペルチェックのオプション[1419]が表示されます現在アクティブになっているビューに従い、「**ツール | スペルチェックのオプション**」コマンドにて、スペルチェックのオプション[1419] ダイアログ(スキーマビュー、WSDL ビュー、XBRL ビュー、Authentic ビュー、ブラウザービュー)、または XML スペルチェックのオプション[1418] ダイアログ(テキストビュー、グリッドビュー)が表示されます。コンテキストダイアログにある「**スペリングオプション**」ボタンにて、スペルチェックのオプションダイアログを表示することができます。

これら2つのダイアログでは、以下に記される様々な設定を行うことができます:

- スペルチェックのコンテキスト[1418]
- スペルチェックのオプション[1419]
- スペルチェッカーに辞書を追加[1421]
- ユーザー辞書と作業[1421]

XML スペルチェックのオプションのダイアログ

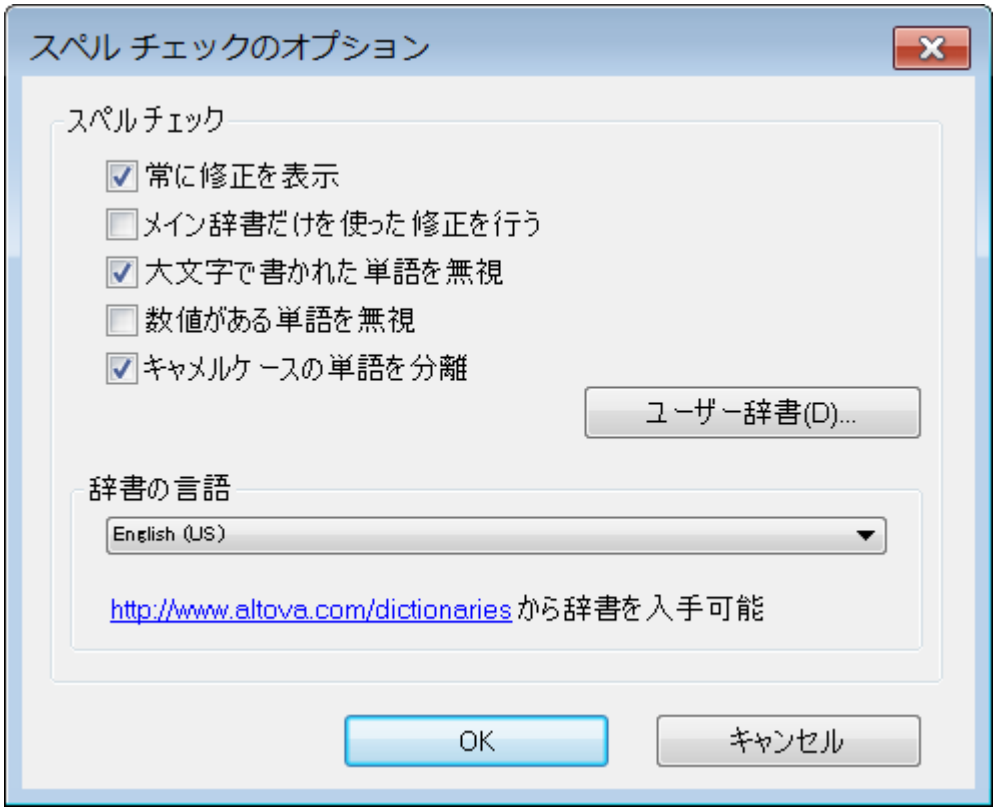
テキストビューまたはグリッドビューで「**スペルチェックのオプション**」コマンドをクリックすると、スペリングコンテキストダイアログが表示され、スペルチェックを行う範囲を指定することができます(以下のスクリーンショットを参照)。チェックを行う要素や属性を選択できるほか、CDATA セクションやコメントをチェックするか指定することができます。

スペルチェックを行う対象、またはスペルチェックを行う対象から除外する要素や属性を指定することができます。スペルチェックを全ての要素や属性に対して行うこともできます。

ダイアログ下部にある「**詳細スペリングオプション**」ボタンをクリックすることで、スペルチェックのオプションダイアログが表示されます。

スペルチェックのオプション

スペルチェックのオプションダイアログではグローバルなスペルチェッカーオプションを設定することができます。



常に修正を表示:
このオプションを選択することで、ユーザー辞書とそれ以外の辞書から得られた単語の候補が候補リスト ボックスに表示されます。このオプションを無効にすると、候補が表示されなくなります。

メイン辞書だけを使った修正を行う:
このオプションを選択することで、言語辞書（メイン辞書）だけが使用されます。提案にユーザー辞書は使用されません。「**ユーザー辞書**」ボタンも無効化され、ユーザー辞書の編集を行うことができなくなります。

大文字で書かれた単語を無視:
このオプションを選択することで、すべて大文字で記述された単語がチェックの対象から外されます。

数値を伴う単語を無視:
このオプションを選択することで、数値を含んだすべての単語が無視されます。

キャメルケースの単語を分離:
単語内に大文字の文字が含まれる単語をキャメルケースの単語といいます。例えば"CamelCase" という単語では、"Case" の"C" が大文字で示されているため、キャメルケースの単語として認識されます。キャメルケースの単語は通常辞書に含まれていないため、スペルチェッカーによりエラーとして認識されます。*キャメルケースの単語*オプションにより、キャメルケースの単語を大文字からはじまるパーツに分離して、個々のパーツをチェックすることによりこの問題を回避することができます。このオプションはデフォルトでチェックされています。

辞書の言語:
このコンボ ボックスではスペルチェッカーにて使用される辞書の言語が選択されます。デフォルトの選択はEnglish (US) となっています。Altova W eb サイトからはその他の言語の辞書を無料でダウンロードいただけます。

(C) 2015-2021 Altova GmbH

スペルチェッカーに辞書を追加

辞書の言語は、.aff ファイルと.doc ファイルという2 つのHunspell 辞書ファイルから構成されます。言語辞書は、以下の場所にある Lexicons フォルダーへインストールされます。全ての言語の辞書は次の場所にある `Lexicons` フォルダー内にインストールされます: `C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons`.

Lexicons フォルダー内部には、異なる言語に対応した辞書のファイルが、<言語名>¥辞書ファイル> という構成で各フォルダーに収められます。例えば、2つの(`English (British)` と`English (US)`) 英語辞書が以下のように配置されます:

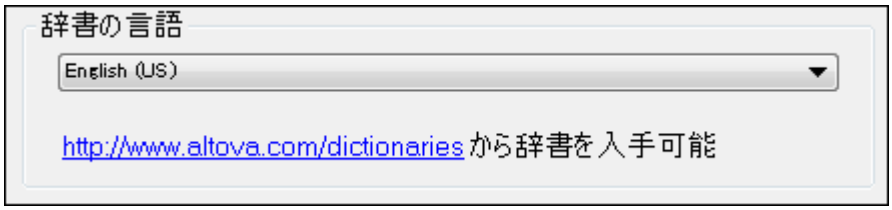
```
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (British)
\en_GB.aff
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (British)
\en_GB.dic
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (US)\en_US.dic
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (US)
\en_US.dic
```

スペルチェッカーオプション ダイアログでは、「辞書の言語」コンボボックスにあるドロップダウンリストに利用可能な辞書の言語が表示されます。Lexicons フォルダー以下にある言語サブフォルダーの名前がリストには表示されます。例えば上にある英語辞書の場合、English (British) とEnglish (US) という名前がリストに表示されます。

インストールされた辞書はコンピューターを使っているすべてのユーザー、そして(32 ビットか64 ビットかにかかわらず) 複数のAltova 製品により共有されます。

スペルチェッカーで使用される辞書は以下に示される2 種類の方法で追加することができ、どちらの方法でもファイルをシステムに登録する必要はありません

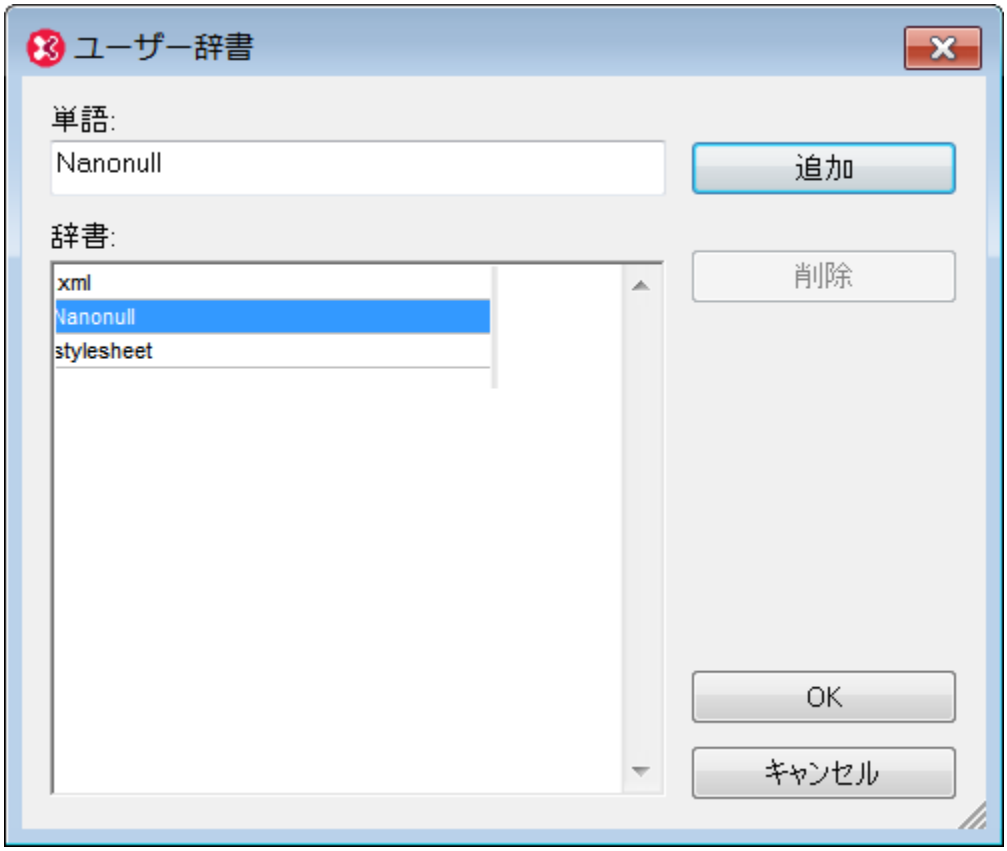
- Hunspell 辞書をLexicons フォルダーの新規サブフォルダーへ追加する。Hunspell 辞書は、例えば http://wiki.services.openoffice.org/wiki/Dictionaries や http://extensions.services.openoffice.org/en/dictionaries.といった場所からダウンロードすることができます (OpenOffice ではOXT という拡張子のZIP アーカイブが配布されているため、ファイル拡張子を.zip へ変換して、回答した .aff と.dic ファイルをファイルをLexicons フォルダー以下にある言語フォルダーへコピーしてください。Hunspell 辞書はMyspell 辞書をベースにしているため、Myspell 辞書を使用することもできます)
- 複数の言語辞書をコンピューターの適切な場所へインストールするAltova 辞書インストーラーを使用する。スペルチェックのオプションダイアログにあるリンクをクリックすることで、インストーラーをダウンロードすることができます(以下のスクリーンショットを参照)。



メモ 辞書に対して適用されるライセンスの条項に同意するか、そしてお使いのコンピューターにおける辞書の使用が適切なものかの判断は、お客様へ委ねられます。

ユーザー辞書と作業

コンピューター上の各ユーザーには、ユーザーにより許可された単語を収めるためのユーザー辞書が与えられます。スペルチェックを行うと、言語辞書とユーザー辞書に含まれている単語のリストに対してドキュメント内の単語がチェックされます。ユーザー辞書ダイアログでは、ユーザー辞書に単語を追加、また削除することができます(*以下のスクリーンショットを参照*)。スペルチェックのオプションダイアログにて「ユーザー辞書」ボタンをクリックすることで、この辞書へアクセスすることができます(*このセクションの2番目のスクリーンショットを参照してください*)。



ユーザー辞書へ単語を追加するには、単語フィールドに単語を入力し、「**追加**」ボタンをクリックしてください。入力された単語がアルファベット順で辞書ペインに追加されます。辞書から単語を削除するには、目的の単語を辞書ペインにて選択し、「**削除**」ボタンをクリックしてください。その単語が辞書ペインから削除されます。ユーザー辞書ダイアログにおける編集を終えた後には「OK」をクリックすることで、変更点がユーザー辞書へ保存されます。

スペルチェックの際に単語をユーザー辞書へ追加することもできます。 スペルチェックダイアログ[1414] は取るアクションのためのプロンプトをポップアップします。「**辞書に追加**」ボタンをクリックすると、未知の単語がユーザー辞書へ追加されます。

ユーザー辞書は以下の場所にあります: `C:\Users\<user>\Documents\Altova\SpellChecker\Lexicons\user.dic`

28.17.3 スクリプトエディター

スクリプトエディターコマンドにより、スクリプトエディターウィンドウが表示されます。スクリプトエディターの使用方法については、ドキュメンテーションの スクリプト環境[1505] セクションを参照ください。

メモ: スクリプトエディターを動作させるには、.NET Framework のバージョン 2.0 以降を、お使いのコンピューターにインストールする必要があります。

(C) 2015-2021 Altova GmbH

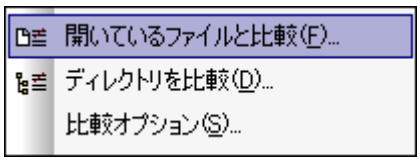
28.17.4 マクロ

マクロ コマンドにマウスオーバーすることで、XMLSpy にて現在アクティブなスクリプトプロジェクトにて定義されているマクロがサブメニューに表示されます（以下のスクリーンショットを参照）。

サブメニューにあるマクロをクリックすることで、マクロが開始されます（上のスクリーンショットを参照）。

28.17.5 比較

XMLSpy には比較（または差分）機能が搭載されており、XML やテキストファイル、フォルダー間の比較を行うことができます。



ツール メニューには、以下のアイテムが用意されており、ファイルやフォルダーに対して比較処理を行うことができます:

- 開いているファイルと比較 1423
- ディレクトリを比較 1427
- 比較オプション 1430

以下のサブセクションでは、これらのコマンドに関する記述を行います。

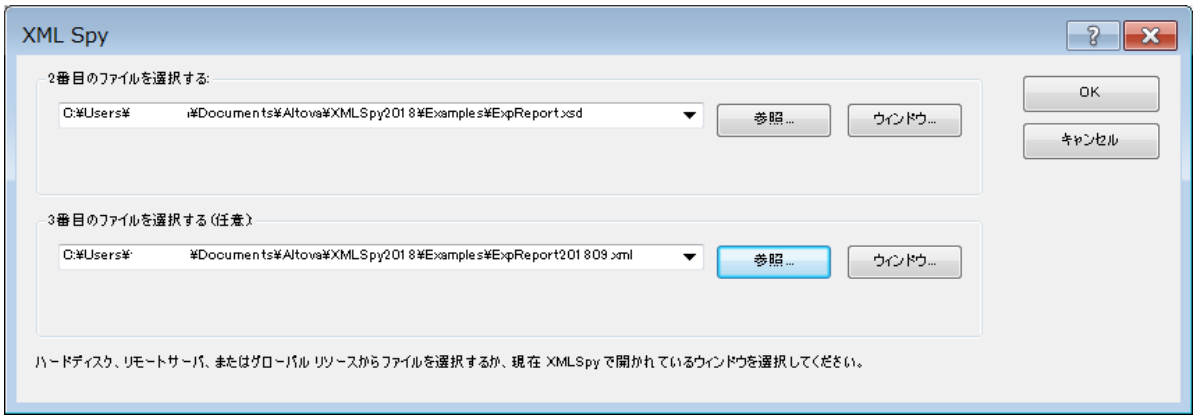
28.17.5.1 開いているファイルと比較

このコマンドによりファイルを他のファイルと比較することができます。比較は比較されるファイルをメインウィンドウ内で左右に表示し、ファイルの間に差分をハイライトして表示します。差分が2つのファイル間である場合、ハイライトされる差分は緑で表示されます。コンテンツが3つのファイル間で異なる場合（この機能は *Enterprise Edition のみで使用することができます*）差分はピンクで表示されます。（タグの構造とセマンティクスが顕著な場合）ファイルを XML ドキュメントとして、または、テキストドキュメントとして比較することができます。

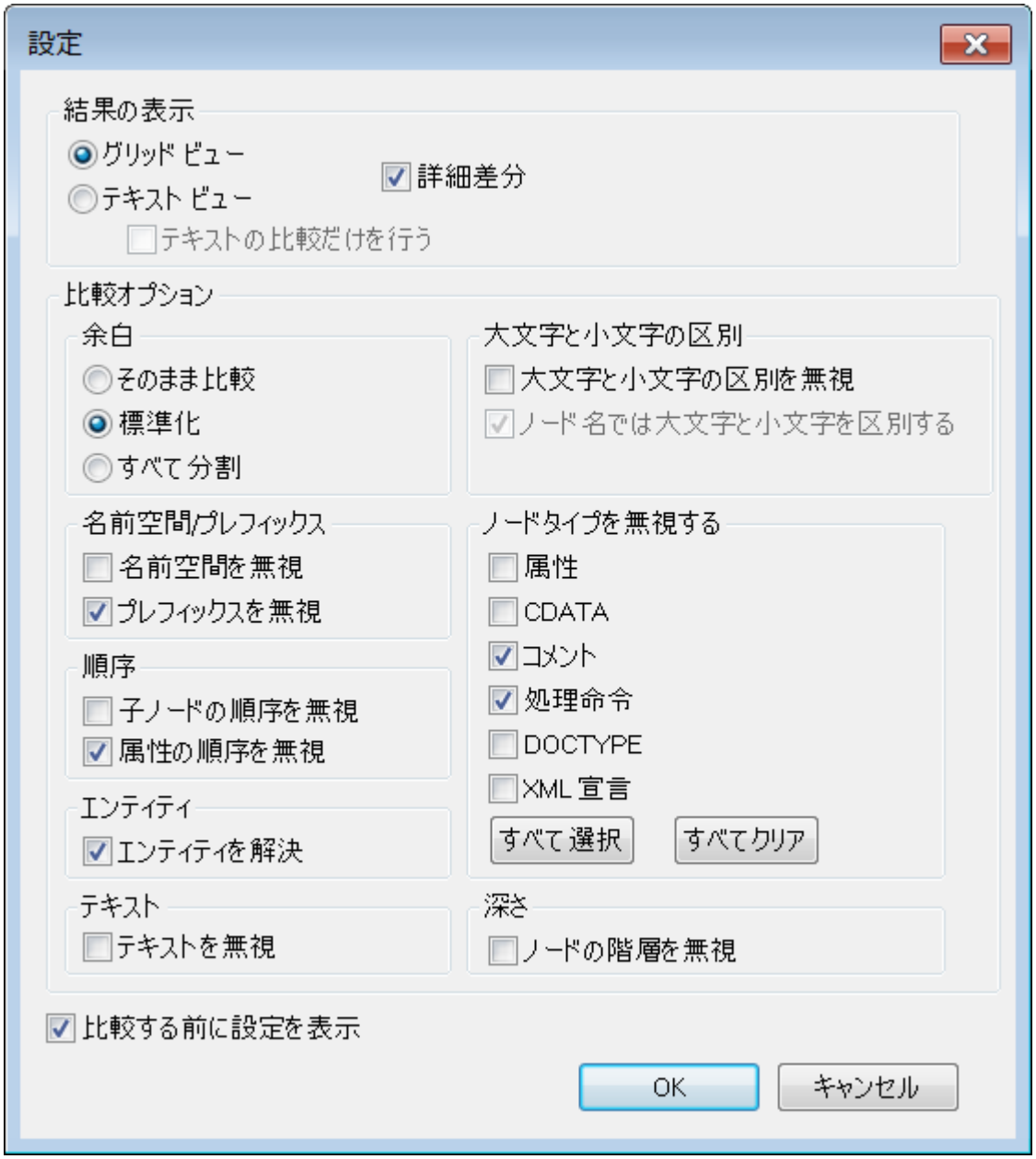
メモ 3 方向 ファイルの比較は**Enterprise Edition のみで使用することができます。**

（GUI 内で）アクティブなファイルを他のファイルと比較する場合、以下をおこないます:

1. メインウィンドウ内でアクティブなファイルを比較するには「**ツール | 開いているファイルと比較**」をクリックします（1度にアクティブに設定できるファイルは1つです）。アクティブなファイルと比較することのできる1つまたは2つのファイルを参照するダイアログが表示されます（*下のスクリーンショット*）。



2. ファイルエクスプローラー、グローバルリソース、またはURL を使用して 参照[1132] をクリックしてファイルを参照します。「ウィンドウ」をクリックして、XMLSpy のウィンドウ内で開かれているファイルを選択します。
3. 「OK」をクリックします。設定ダイアログが表示されます(*下のスクリーンショット*)。これらの設定は、トピック比較オプション[1430]内で説明されています。このダイアログを比較セッションが開始される都度開かないようにするには、(ダイアログの下にある)「比較の開始前に設定を表示する」チェックボックスのチェックを解除してください。



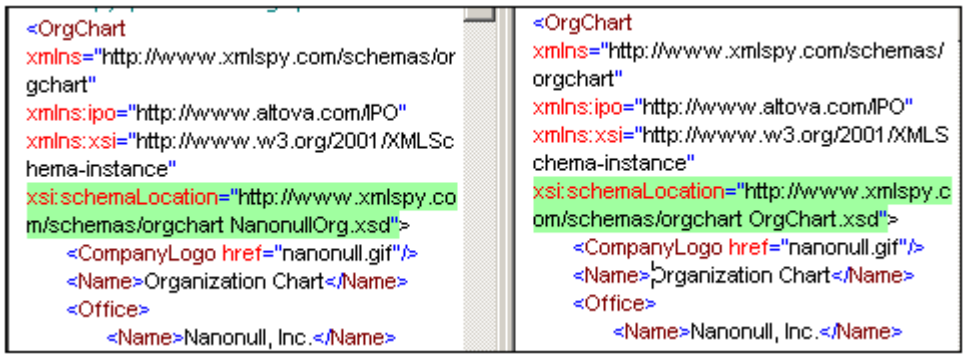
4. 必要な設定を選択し、「OK」をクリックします。次の2つの状態が発生します: (i) 個別のペイン内に比較するファイルが左右に並べられて表示されます。(ii) ファイルの比較 管理ウィンドウが表示されます。(次のセクション *ファイルの比較*[1426] *のスクリーンショット参照*)。
5. ファイルの比較 管理ウィンドウ内のボタンを使用して差分をナビゲートしマージします。(*使用方法に関しては ファイルの比較*[1426] *を参照してください*。)
6. 差分のレビューまたはマージが終わると、「完了」をクリックしてください。

メモ (「ツール | 比較オプション」[1430]選択することにより)比較セッション中に比較設定が変更される場合がありますが、影響は次の比較セッションからです。新規の設定は現在のセッションに影響を与えません。

ファイルの比較と差分のマージ

2つの種類の比較が存在します:

- *2 方向比較:* 2つのファイルが比較されます。左側のペイン内にアクティブなファイルが表示され、2番目のファイルに対して比較されます(*下のスクリーンショットを参照*)。



- *3 方向比較 (Enterprise Edition のみ):* 3つのファイルが比較されます。左側のペイン内にアクティブなファイルが表示されます。比較のために選択された2つの最初のファイルが中央のペインに表示されます。この中央のファイルがベースファイルと称されます。左側のファイルと中央のファイルの差分をマージする、または、右側のファイルと中央のファイルの差分をマージすることもできます。

メモ ファイルの比較管理ウィンドウ内のボタンが正確に対応しない場合があるためウィンドウ内でペインを移動しないでください。

差分と競合

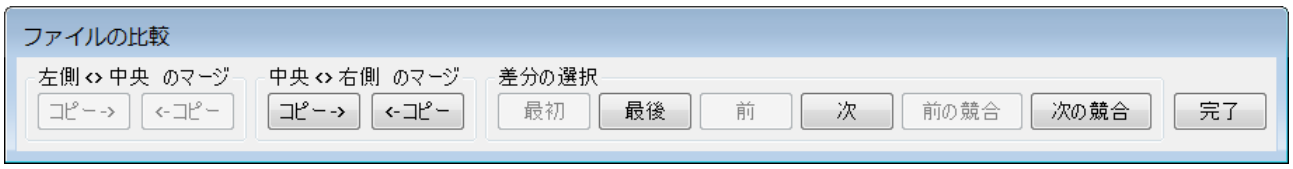
差分と競合は異なる色でハイライトされ、個別にナビゲートすることができます(*下の*ハイライト色[1427] *を参照*)。

- *差分:* 対応する場所の2つのファイル内のコンテンツが異なる場合。緑色でハイライトされます。
- *競合:* 対応する場所の3つのファイル内のコンテンツが異なる場合。ピンク色でハイライトされます。

メモ 2 方向比較は競合は表示せず、差分のみを表示することに注意してください。

差分と競合のナビゲート

ファイルの比較 管理ウィンドウの差分の選択ペイン内のボタンを使用してドキュメント内を移動することができます(*下には 2 方向比較と 3 方向比較が表示されています*)。ペインには2 方向比較のための4つのボタンと、3 方向比較のための6つのボタンが含まれています(*スクリーンショットを参照*)。



ファイルの比較 管理ウィンドウには次のボタンが含まれます:

- *最初:* ドキュメント内の全ての差分と競合の最初に移動します。
- *最後:* ドキュメント内の全ての差分と競合の最後に移動します。

- *前:* 現在アクティブな差分または競合から前の差分に移動します。
- *次:* 現在アクティブな差分または競合から次の差分に移動します。
- *前の競合:* 現在アクティブな差分または競合から前の競合に移動します。
- *次の競合:* 現在アクティブな差分または競合から次の競合に移動します。

差分と競合のマージ
マージペイン内のボタンをしようしてハイライトされたコンテンツを他のペインにコピーします。適切なペインの組合せのコピーの方向を使用してください。マージを有効化するには、次の比較オプション[1430]が満たされている必要があります:

- 詳細差分がチェックされており、
- ノードの階層を無視のチェックが解除されていなければなりません

差分のレビューとマージが終わると、「完了」をクリックします。

メモ マージする場合、比較セッションは停止し、変更を元に戻すファイルを選択して「編集 | 元に戻す」を選択し、「Ctrl+Z」を押してください。

メモ 比較セッションがアクティブな間、ビューの編集または変更は許可されていません。ビューの変更を試みると、比較セッションの中断を警告するメッセージが表示されます。

ハイライト色

2つのファイル間で対応するラインが異なる場合、両方のファイル内で差分は緑でハイライトされます。3つのファイル間でラインが異なる場合は全てのファイル内で差分がピンク色でハイライトされます。差分が選択されている場合、両方のファイルで濃い緑色で表示されます。競合が選択される場合、濃いピンク色で表示されます。

比較セッション内で使用されるハイライト色は以下のとおりです。

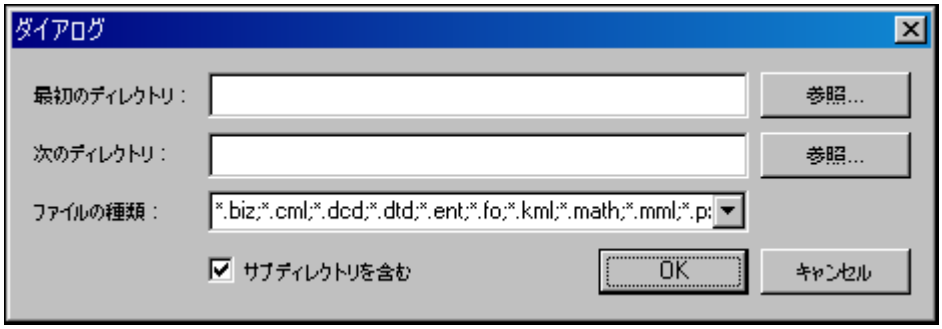
	差分
	現在の差分
	マージされた差分
	現在のマージされた差分
	競合
	現在の競合

28.17.5.2 ディレクトリの比較

ディレクトリの比較コマンドにより、2つのディレクトリをサブディレクトリを含めて、または含めないで比較することができます。ディレクトリの比較では、一方には無いファイルや、同名のファイルでも内容が同一であるかどうかが表示されます。

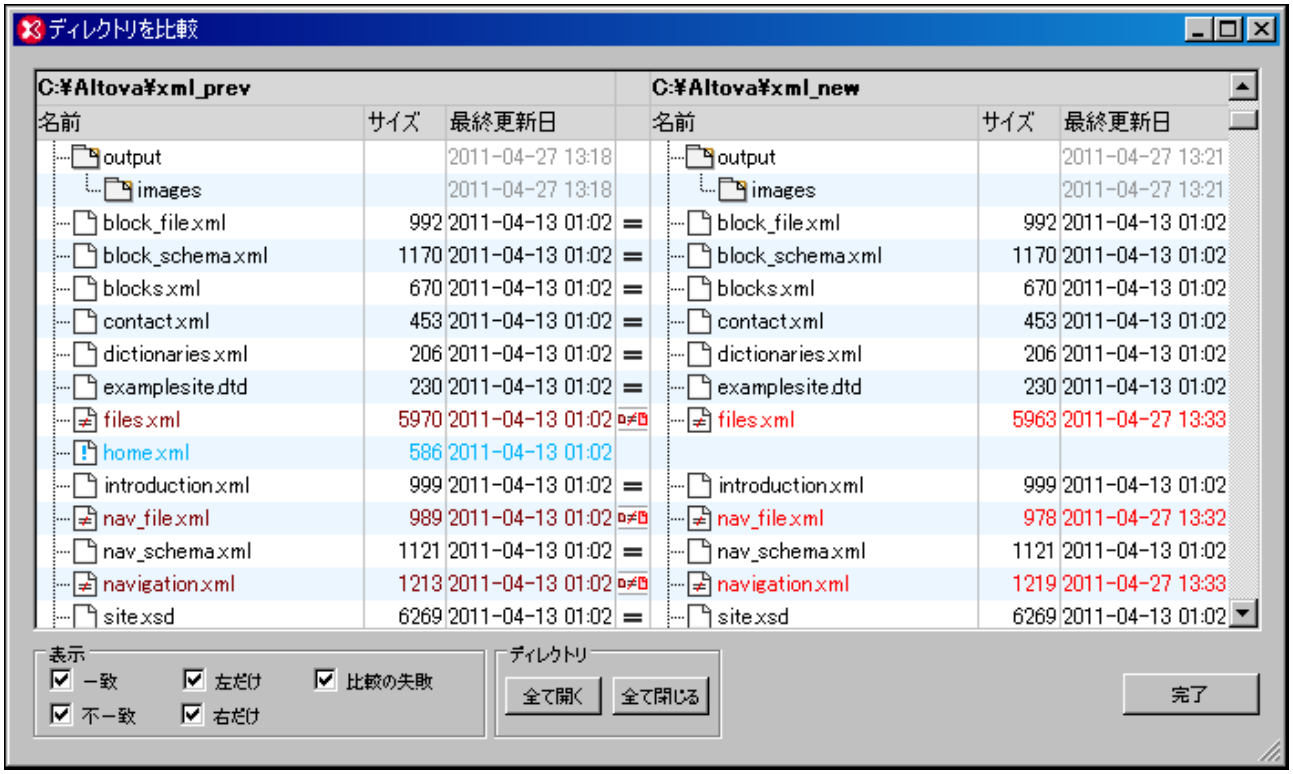
2つのディレクトリを比較するには:

1. 「ツール | ディレクトリの比較」をクリックします。以下のダイアログが表示されます:



2. 比較するディレクトリを指定します。比較する対象にサブディレクトリも含める場合は、**サブディレクトリを含む**チェックボックスにチェックを入れてください。
3. **ファイルの種類**フィールドにて、比較を行いたいファイルの種類を選択します。ドロップダウンメニューからは、(i) XML ファイルタイプ、(ii)(ii) XMLSpy にて定義されたファイルタイプ[1458](iii) 全てのファイルタイプを選択することができます。ZIP アーカイブはディレクトリとして扱われ、ディレクトリならびにZIP アーカイブ内部にあるファイルが結果ウィンドウに表示されます。ZIP アーカイブをディレクトリとして読み取るには、目的の拡張子がファイルの種類エントリーに含まれていることを確認してください。
4. 「OK」をクリックします。(比較オプション[1430]にて記述される) 設定ダイアログが表示されます。
5. ファイルの比較を行うための設定を選択します。
6. 「OK」をクリックします。比較の進捗状態を示すダイアログが表示されます。

比較の結果が、以下に示されるウィンドウに表示されます:



ディレクトリシンボル

ディレクトリの名前は、全て黒色で表示されます。

ディレクトリは縮退されており、コンテンツは表示されていません。

ディレクトリは展開されており、コンテンツがその行以下に表示されます。

ディレクトリにはファイルが含まれており、中に含まれるファイルは、比較することができないか、比較対象となるファイルと同一のものです。

比較対象のディレクトリには含まれていないファイルが、このディレクトリに含まれています。

比較対象のディレクトリに含まれているファイルとは内容が異なるファイルがこのディレクトリに含まれています。

比較対象のディレクトリには含まれていないファイルが1つ以上、また、比較対象のディレクトリに含まれているファイルとは内容の異なるファイルが1つ以上、このディレクトリには含まれています。

ファイルシンボル

比較ステータスにより、ファイル名の表示に使用される色が以下のように変化します:

比較対象となるファイルと比較を行うことができません。中間のカラムに疑問符が表示されます。ファイル名は黒色で表示されます。

比較対象となるファイルと同一のファイルです。中間のカラムに等号記号が表示されます。ファイル名は黒色で表示されます。

比較対象となっているディレクトリに、このファイルは存在しません。中間のカラムには何も表示されず、ファイル名は水色で表示されます。

比較対象となるファイルと内容が一致せず、比較対象のファイルと比べて、最終更新日時が新しいことを表します。(最終更新日時が新しい方のファイルは明るい赤色で表示され、中間カラムにあるアイコンも、新しいファイルがある方向に明るい赤色が表示されます。

表示オプション

- 結果ウィンドウの下部にある表示ペインにあるチェックボックスを操作することで、表示するファイルの種類の選択することができます.
- ディレクトリペインにあるボタンをクリックすることで、全てのサブディレクトリを展開、または縮退することができます。
- フォルダーアイコンをダブルクリックすることで、サブディレクトリを展開または縮退することができます。
- ファイルのタイトルバーを右クリックして、サイズと最終更新日をクリックすることで、サイズと最終更新日の表示を切り替えることができます。

- カラムをドラッグすることで、カラムの幅を変更することができます。
- 結果ウィンドウは、移動、最小化、そして大きさを変更することができます。

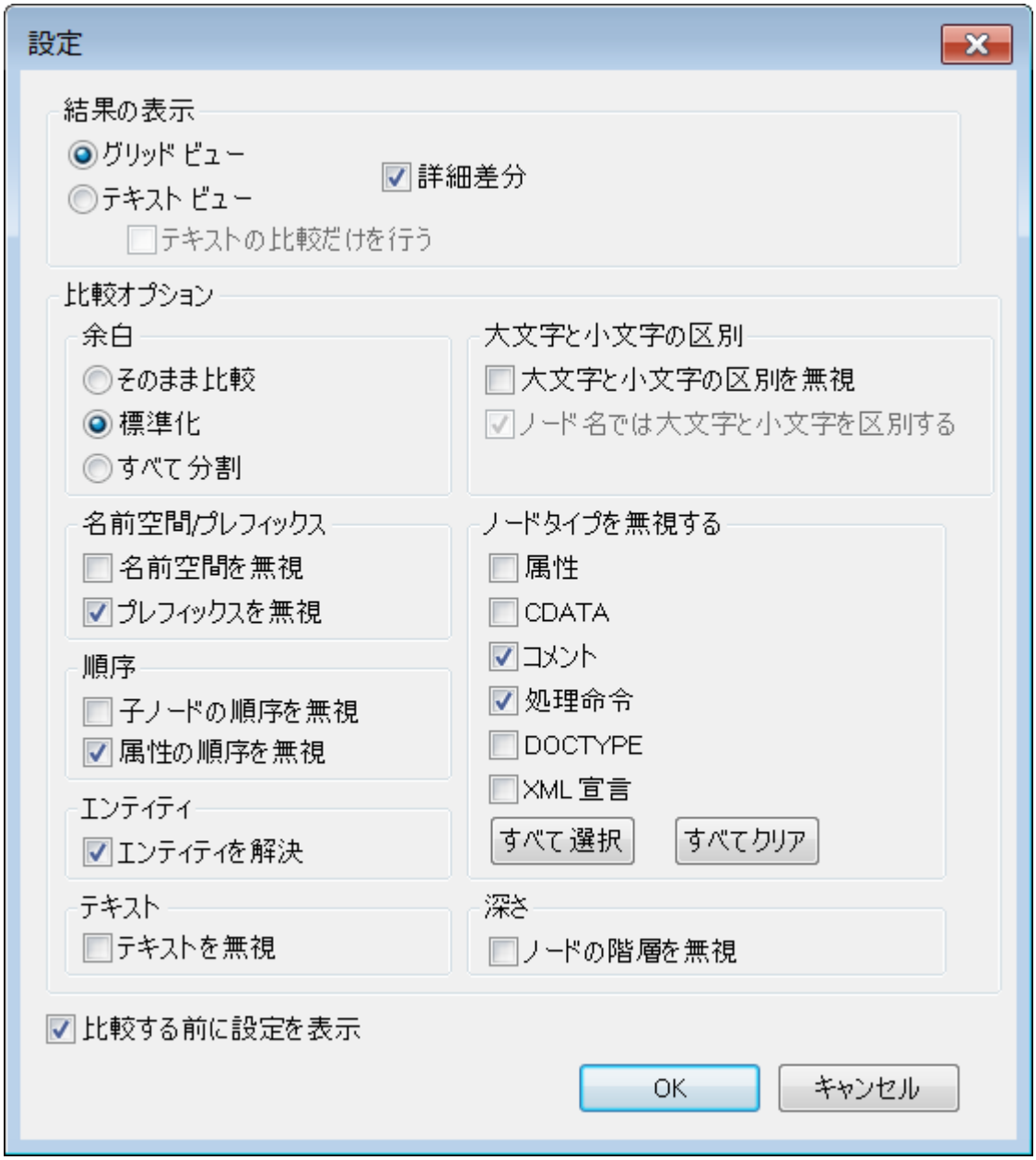
ファイルの比較とマージ

結果ウィンドウに表示されている行をダブルクリックすることで、両方のファイルがメインウィンドウで開かれ、両方のファイルに対してファイルの比較が行われます。その後は、通常の比較セッションと同様に進めることができます(開いているファイルと比較... 1423)を参照ください)。

28.17.5.3 比較オプション

「ツール | 比較オプション」を選択することで、設定ダイアログを開くことができます(以下のスクリーンショットを参照)。設定ダイアログでは、比較セッションの設定を指定することができます。設定ダイアログで指定されたオプションが、その比較セッションで使用されます。

(C) 2015-2021 Altova GmbH



結果の表示

比較の結果が表示されるビューを、以下のオプションから選択します:

- グリッドビュー(XML 比較)
- テキストの比較だけを行う、が無効なテキストビュー(XML 比較)
- テキストの比較だけを行う、が有効なテキストビュー(テキスト比較)

XML の比較結果を表示することができるビューを選択すると、ドキュメントは XML ドキュメントとして扱われ、XML 比較オプションが有効になります。テキスト比較が選択された場合は、テキストの比較でも使用することのできるオプション(余白と大文字/小文字の比較)だけが有効になり、その他のオプションは無効になります。

メモ: 差分のマージは、テキストビューならびにグリッドビューで、そしてXMLとテキストの比較モードで行うことができます。マージの取り消しを行いたい場合は、比較セッションを中止して、変更されたファイルを選択し、「**編集 | 元に戻す**」を選択するか、**Ctrl + Z** を押下してください。

詳細差分

このオプションが選択されていない場合、すぐそばにある兄弟要素も1つの差分として認識され、マージオプションが無効になります。このオプションが選択されている場合、すぐそばにある兄弟要素は異なる差分セグメントとして認識され、マージオプションが有効になります。

メモ: マージを行うには、「**詳細差分**」チェックボックスにチェックを入れる必要があります。

余白

余白として認識される対象は、タブならびに改行です。このオプションにより、ファイルを変更することなく余白の正規化を行うことができ、連続した余白が1つの余白としてみなされ、全ての余白が取り除かれます。余白オプションはXMLとテキスト比較の両方に対して使用することができます。

大文字と小文字の区別

「**大文字と小文字の区別を無視**」チェックボックスにチェックが入れられていない場合、(XMLの比較において)ノード名で使われている文字列の大文字と小文字を区別するか指定することができます。大文字と小文字の区別オプションは、XMLとテキスト比較の両方に対して使用することができます。

名前空間/プレフィックス

これらのオプションにより、差分の検索を行うにあたり、名前空間やプレフィックスを無視するか指定することができます。

順序

子ノードの順序を無視オプションが有効になっている場合、子ノードの順序は比較の際に考慮されなくなります。比較は子ノード全体に対して行われ、ドキュメント間にある子ノード同士の違いが、相対的な出現箇所だけである場合、それらの違いは無視されます。

メモ: 各子要素ノードは、その名前、属性、そして位置により特定されます。あるノードの名前が、兄弟要素の名前と同じ場合、これらのノードを特定するために(「子ノードの順序を無視」オプションが有効になっている場合でも)ノードの位置情報が使用されます。このオプションは、ドキュメントの各レベルに対して個別に適用されます。

子ノードの順序を無視オプションが無効になっている場合、ノードの順序による違いも差分として認識されます。属性の順序を無視オプションも利用することができ、各要素にある属性の順序に対して適用されます。

エンティティ

エンディティを解決オプションが有効になっている場合、ドキュメントにある全てのエンティティが解決されます。そうでない場合、ドキュメントにあるエンティティ名が、そのままの状態で比較されます。

テキスト

「**テキストを無視**」オプションが選択されている場合、対応するテキストノード間の比較は行われません。

無視するノードタイプ

比較セッションにて比較の対象には含まれないノードの種類にチェックを入れてください。無視することのできるノード型は、属性、CDATA、コメント、処理命令、DOCTYPE ステートメント、そしてXML宣言です。

(C) 2015-2021 Altova GmbH

深さ

「**ノードの階層を無視**」にチェックが入っていると、任意の要素における子ノード（子孫ノード）が無視されます。マージを行うには、このオプションを無効にする必要があります。

比較する前に設定を表示

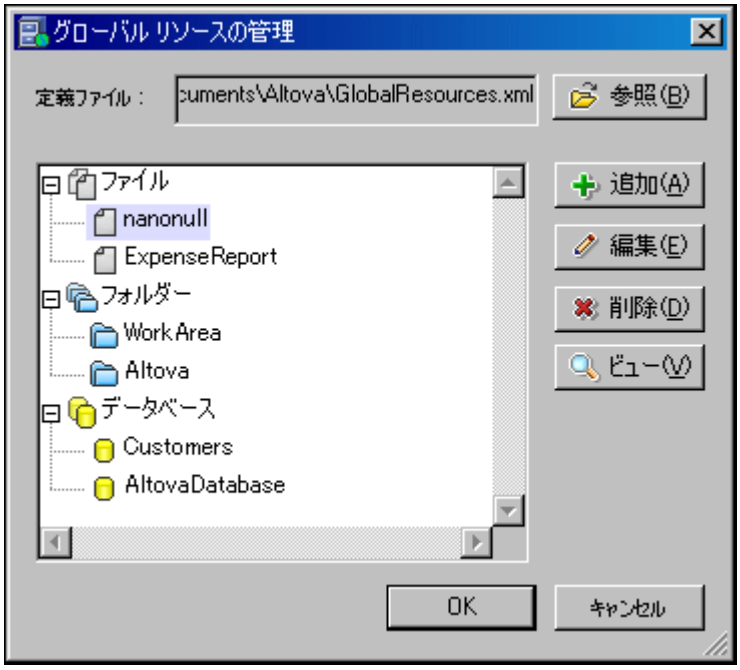
このオプションにチェックを入れることで、「**開いているファイルと比較**」コマンドが選択されるたびに、設定ダイアログが表示されます。比較を行う前に設定ダイアログを表示することで、比較設定のチェックや変更を、各比較に対して行うことができます。

このコマンドがチェックされていない場合、比較コマンドが指定された直後に、比較セッションが開始されます。

28.17.6 グローバルリソース

グローバルリソースコマンドにより、以下の操作を行うことができるグローバルリソースダイアログが表示されます（以下のスクリーンショットを参照）：

- グローバルリソースXML ファイルを指定。
- ファイル、フォルダー、データベースのグローバルリソース（またはエイリアス）を追加。
- 各グローバルリソース（エイリアス）に対して、各種構成を指定。各構成により、特定のソースへのマッピングが行われます。

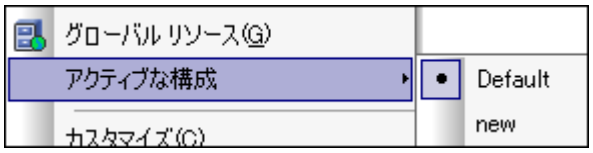


グローバルリソースの定義方法に関する詳細は、グローバルリソースの定義[934]セクションを参照ください。

メモ Altova グローバルリソースダイアログへは、グローバルリソースツールバー[1440]（「**ツール | カスタマイズ | ツールバー | グローバルリソース**」）からアクセスすることができます。

28.17.7 アクティブな構成

アクティブな構成メニューアイテムにマウスオーバーすることで、現在アクティブな[グローバルリソースXML ファイル](1433)にて定義されている全ての構成が表示されます(以下のスクリーンショットを参照)。



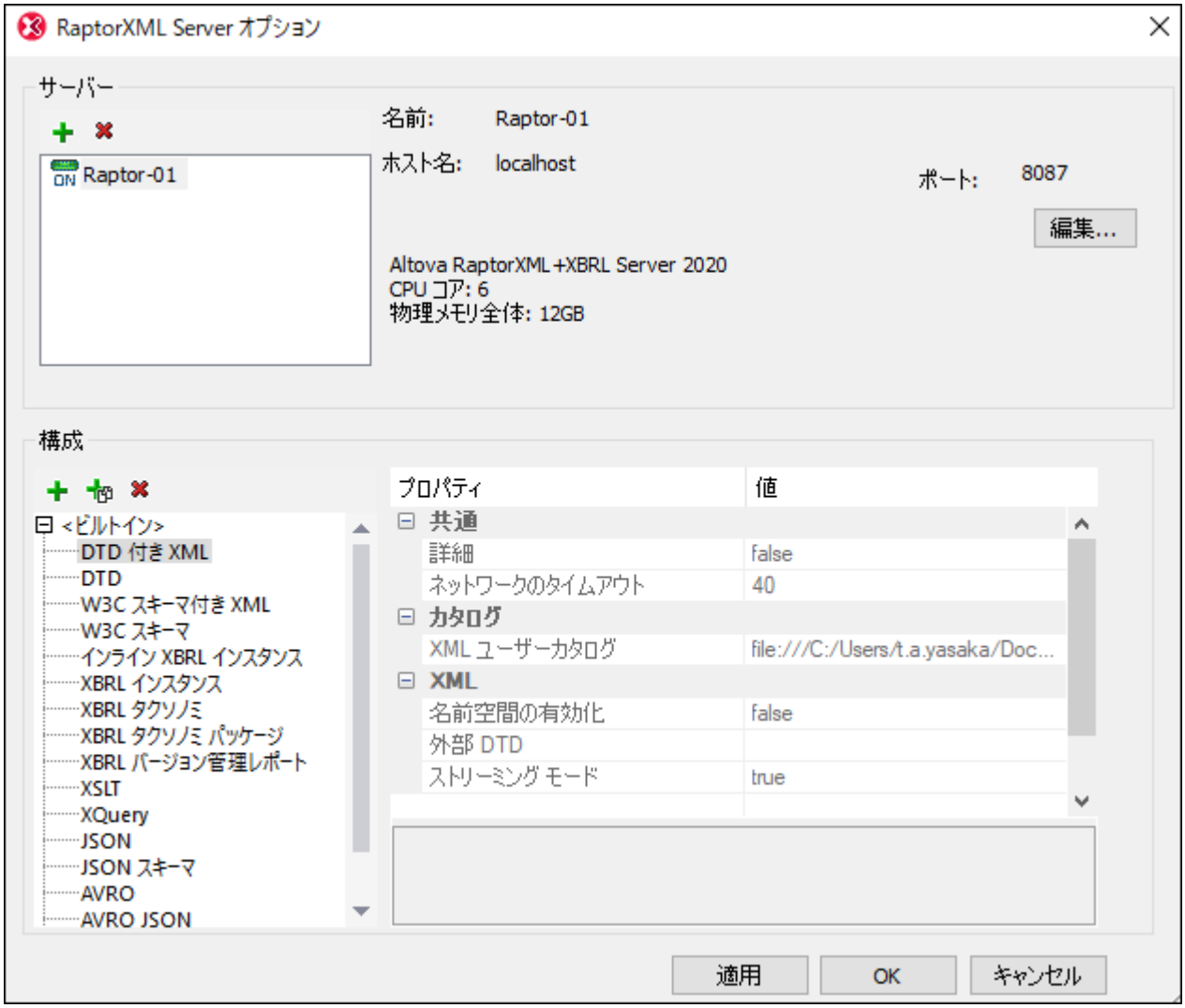
現在アクティブな構成は、中黒(・)により表示されます。上のスクリーンショットでは、現在アクティブな構成がDefault となっています。アクティブな構成を変更するには、アクティブにしたい構成を選択してください。

メモ: アクティブな構成は、[グローバルリソースツールバー](1440)(「**ツール | カスタマイズ | ツールバー | グローバルリソース**」)から選択することができます。

28.17.8 Raptor Server の管理

Raptor Server の管理 コマンドにより、Raptor Servers 使用することのできる Raptor Servers のプールに追加し、各サーバーのために複数の構成を定義することができます。XML とXBRL ドキュメントを検証するために RaptorXML Server を使用する場合は、[RaptorXML Server を使用して検証](957)を参照してください。

(C) 2015-2021 Altova GmbH



Raptor Server の追加

ダイアログの *サーバーペイン*（*下のスクリーンショット*）内から、**「サーバーの追加」**アイコンをクリックします。アイコンをクリックして、Raptor Server、がインストールされているマシンの名前(ホスト名)、およびRaptor Server のポートを認識するための名前を入力します。「OK」をクリックして設定を保存します。

- *名前:* 選択されるすべての文字列。XMLSpy 内では、特定のRaptorXML Server を認識するために使用されています。
- *ホスト名:* Raptor Server がインストールされているマシンのネットワークの名前 またはIP アドレス。処理はホスト名よりもIP アドレスを使用した場合のほうが速くなります。`localhost`（ローカルマシン）に対応するIP アドレスは`127.0.0.1` です。
- *ポート:* Raptor Server がアクセスされるポート。このポートは、（`server_config.xml` という名前の）Raptor の構成ファイル内で指定されています。ポートは、固定され、サービスへのリクエストが対応できるように知らされている必要があります。Raptor 構成ファイルの詳細は、*RaptorXML Server* および *RaptorXML+XBRL Server* のユーザーマニュアルを参照してください。

サーバー情報を入力すると、「OK」をクリックします。入力したサーバー名がペインの左側のサーバーリストに表示されます。サーバーの名前の横に緑のアイコンが表示され、サーバーが開始され、実行中であることを示しています。サーバーの詳細はペインに表示されています（*上のスクリーンショット参照*）。赤いアイコンはサーバーがオフラインであることを示します。サーバーを検出することができない場合、エラーメッセージが表示されます。

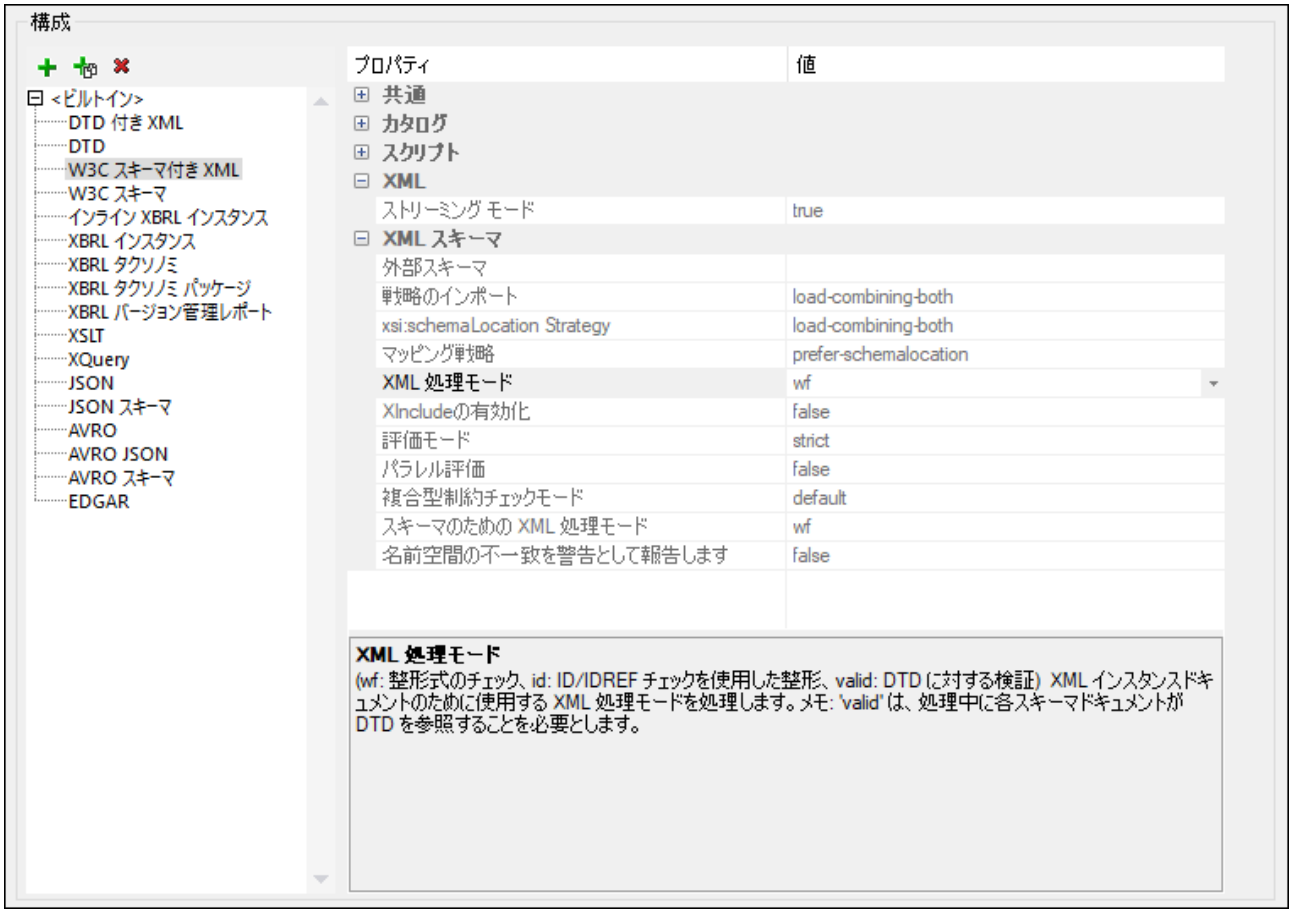
メモ Raptor Server は、サーバーが追加される際に実行中である必要があります。これはXMLSpy がサーバーの情報を取得し、保管するために必要です。サーバーが追加された後、サーバーがオフラインまたは検出されない場合、これらのシナリオは赤いアイコンまたはエラーメッセージで通知されます。

サーバーの名前、ホスト名、またはポートを編集する場合、左側のペインのサーバーを選択して、「**編集**」ボタンをクリックし、表示されるダイアログ内の情報を変更します。プールからサーバーを削除する場合は、サーバーを選択して「**選択されたサーバーを削除**」アイコンをクリックします。

Server 構成

構成とは、RaptorXML 検証オプションのセットです。サーバーが追加されると、構成を持ち、`default` と名前が付けられます。これはRaptorXML オプションのセットに設定されたデフォルトの値です。他のオプションの値を含む新しい構成を追加することもできます。複数のサーバー構成を追加した後、アクティブな構成になる1 つの構成を選択します。これが「**サーバー上で検証**」コマンドが実行される時に使用される構成です。

構成ペインは2 つの部分により構成されています: (i) それぞれが変更可能なドキュメント型のリストを含む構成を表示する左側のペイン、(ii) 左側のペインで選択されたドキュメント型のための検証オプションを表示する右側のペイン。右側のペインの下には選択されたオプションの説明が表示されています（*上のスクリーンショット参照*）。



構成の追加

RaptorXML Server オプションダイアログの構成ペイン内から(*上のスクリーンショット*)「**構成の追加**」をクリックします。デフォルトのオプション値に新規構成が追加されます。「**選択された構成をコピー**」をクリックすることにより新規構成を作成することもできます。これによりコピーされた構成のオプション値と同じ値を持つ新しい構成が作成されます。新たしい構成は、型 `config<X>` のデフォルトの名前と共に作成されます。構成の名前をダブルクリックして新しい名前を入力して編集します。構成のオプション値のすべてを編集することができます。

構成のオプション値の編集

左側のペインからドキュメント型を選択します。これにより右側のペイン内に選択されたドキュメント型の検証オプションが表示されます。オプションの値を編集するには、(オプションの値の型に従い)以下を行います:

- 値が定義済みの値のセットの1つの場合、そのオプションの値の列のコンボボックスから値を選択します。
- 値に制約がある場合、オプションの値フィールドをクリックして、値を入力します。
- 値がファイルパスの場合、値を入力できるだけでなく、オプションの値の列の「**参照**」ボタンを押して、使用するファイルを参照します。

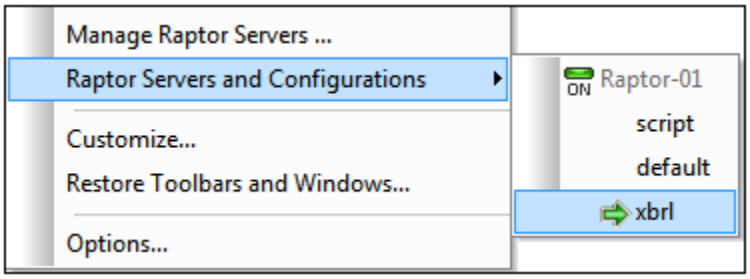
オプションを選択すると、右側のペインの下のボックスに詳細が表示されます。各オプションの詳細は、RaptorXML Server および RaptorXML(+XBRL) Server ユーザーマニュアルのコマンドラインインターフェイスのセクションを参照してください。

構成の削除

左側のペインから、削除する構成を選択して、「**選択された構成を削除**」をクリックします。

28.17.9 Raptor Server と構成

複数のサーバーで複数の構成を定義した場合、サーバーとそのアクティブな構成として1つの構成を選択することができます。アクティブな構成は、以降の検証に使用されます。カーソルを「ツール | Raptor Servers および構成」コマンドにポイントして、追加されたすべてのサーバーとその詳細が含まれるサブメニューから、を含む表示されるサブメニューが表示されます（*下のスクリーンショット参照*）。アクティブな構成と設定するサーバー構成を選択します。したのスクリーンショットでは、`Raptor-01` という名前のサーバーの(緑色の矢印で示されている) `xbrl` 構成がアクティブな構成として選択されています。



28.17.10 XBRL タクソノミ マネジャー

このコマンドはXBRL タクソノミ パッケージの管理を可能にするXBRL タクソノミ マネジャー ダイアログを開きます。

タクソノミをインストールするには、タクソノミの横のチェックボックスを選択、または、インストールするタクソノミを選択して[適用]をクリックします。

詳細に関しては[XBRL タクソノミ マネジャー](#)[722] を参照してください。

28.17.11 カスタマイズ

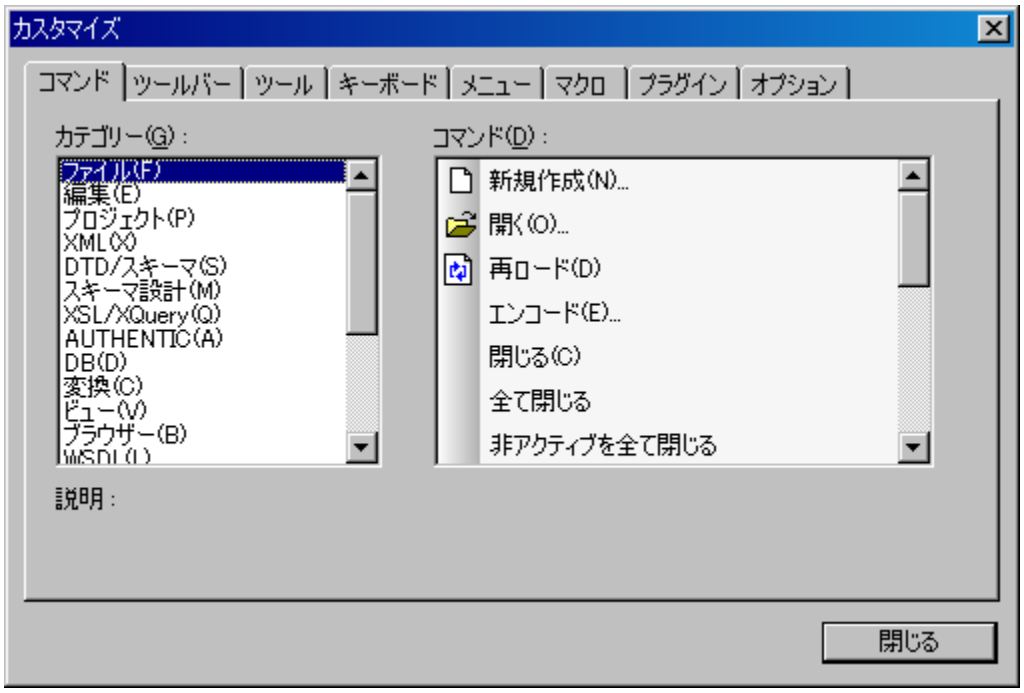
カスタマイズ コマンドにより、XMLSpy に搭載されたメニューやツールバーをカスタマイズすることができます。以下のタブが用意されています:

- [コマンド](#)[1439]: 全てのアプリケーションやマクロコマンドをメニューバー、メニュー、ツールバーへドラッグします。
- [ツールバー](#)[1440]: 個々のツールバーを有効化、無効化、リセットすることができます。
- [ツール](#)[1413]: 外部プログラムを開くコマンドをインターフェースへ追加することができます。
- [キーボード](#)[1443]: 個々のアプリケーションならびにマクロに対してキーボードのショートカットを作成することができます。
- [メニュー](#)[1446]: カスタマイズするメニューバーならびにコンテキストメニューを選択し、有効にすることができます。コマンドタブと一緒に操作します。
- [マクロ](#)[1448]: コマンドに対してマクロを関連付けることができます。
- [プラグイン](#)[1450]: プラグインの有効化と統合を行うことができます。
- [オプション](#)[1454]: ツールバーの表示オプションをセットすることができます。

このセクションではカスタマイズダイアログが開かれ、メニューバー、メニュー、またはツールバーアイテムが右クリックされた時に表示される[コンテキストメニュー](#)[1451] についても記述されます。

28.17.11.1 コマンド

コマンドタブでは、メニューやツールバーをカスタマイズすることができ、必要に応じてアプリケーションコマンドをメニューやツールバーに追加することができます。新たなアプリケーションコマンドやメニューを独自に作成することはできない点に注意してください。



以下の操作によりコマンドをツールバーやメニューへ追加することができます:

1. メニューオプションから「ツール | カスタマイズ」を選択します。カスタマイズダイアログが表示されます。
2. カテゴリーリストボックスから全てのコマンドカテゴリーを選択します。利用可能なコマンドがコマンドリストボックスに表示されます。
3. コマンドリストボックスに表示されたコマンドをクリックして、表示されているメニューやツールバーへドラッグします。許可されている場所へカーソルを移動させると、マウスポインターにI というアイコンが表示されます。
4. コマンドを挿入する場所でマウスボタンを離してください。

以下の点に留意してください:

- コマンドをドラッグすると、マウスポインターの端に小さなボタンが表示されます。これにより、ボタンがドラッグされていることが示されます。
- ポインター下部に表示された"x" は、現在マウスポインターがある位置にドロップできないことを表しています。
- コマンドをドロップすることができる位置(ツールバーまたはメニュー)にカーソルを移動すると、"x" が消えてI というアイコンが表示されます。
- コマンドはメニューまたはツールバーに配置することができます。独自のツールバーを作成した場合1440した場合、このカスタマイズ機能を使用することでコマンドを配置することができます。
- カーソルをメニュー近くに移動させると、そのメニューが開かれ、メニュー内部にコマンドを配置することができるようになります。

コンテキストメニューへコマンドを追加

以下の操作により、コンテキストメニューにコマンドを追加することもできます:

1. カスタマイズダイアログにて メニュータブ[1446]をクリックします。
2. コンテキストメニューペインにてコンボボックスから目的のコンテキストメニューを選択します。選択されたコンテキストメニューが表示されます。
3. カスタマイズダイアログにてコマンドタブへ切り替えます。
4. コマンドリストボックスから追加するコマンドを選択し、コンテキストメニューまでドラッグします。

コマンドまたはメニューの削除

以下の操作により、メニューやコンテキストメニュー、またはツールバーからコマンドを削除する、またはメニュー全体を削除することができます:

1. メニューオプションの**「ツール | カスタマイズ」**を選択してカスタマイズダイアログを開きます。
2. カスタマイズダイアログが表示されている状態で、メニューまたはメニューコマンドを右クリックして、表示されるコンテキストメニューから**削除**を選択します。また、メニューやメニューコマンドを、"x" がマウスポインターに表示される場所までドラッグアンドドロップすることでもメニューまたはメニューコマンドを削除することができます。

このセクションに記述された方法により削除されたメニューコマンドは、復元することができます。削除されたメニューを復元するには、メニューオプションから**「ツール | カスタマイズ | メニュー」**を選択し、アプリケーションフレームメニューペインにある**リセット**ボタンをクリックしてください。その他にも、**「ツール | カスタマイズ | ツールバー」**を選択し、左側に表示されたリストからツールバーを選択し、**リセット**ボタンをクリックすることもできます。

28.17.11.2 ツールバー

ツールバータブでは、(i) 特定のツールバーを有効/無効にする(つまりどのツールバーがインターフェースにて表示されるかを指定する)、(ii) 各ツールバーに表示されるアイコンをセットする、そして(iii) 独自のツールバーを作成することができます。

ツールバーには頻繁に使用されるメニューコマンドが収められています。カーソルをアイコン上部に移動することで、そのアイコンに関する情報がツールチップとステータスバー上に表示されます。ツールバーは、フロート状態のウィンドウとして画面上の任意の場所へ配置することができます。

メモ: ツールバーへコマンドを追加するには、目的のコマンドを コマンド[1439] タブにあるコマンドリストボックスからツールバーへドラッグする必要があります。ツールバーからコマンドを削除するには、カスタマイズダイアログが開かれた状態で、ツールバーからコマンドをドラッグしてください(詳細については コマンド[1439] のセクションを参照ください)。

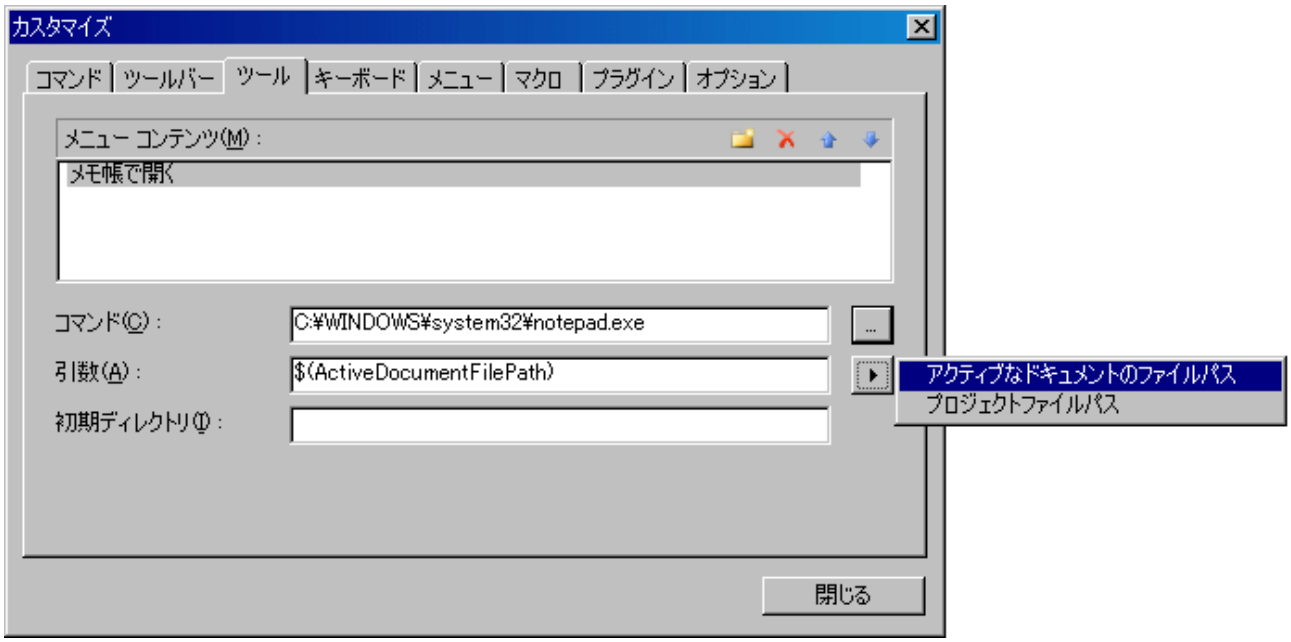
メモ: 特定のビューに対して定義されたツールバーの設定は、デフォルトでそのビューに対してだけ有効になっています。その設定を全てのビューに対して適用するには、ダイアログ下部にある**全てのビューに変更を適用**チェックボックスにチェックを入れてください。

以下の機能を利用することができます:

- ツールバーの有効化/無効化: ツールバーリストボックス内に表示されたチェックボックスをクリックしてください。
- 全てのビューに変更を適用: ダイアログ下部に表示されているチェックボックスにチェックを入れてください。チェックが入っていない場合、アクティブなビューに対してのみ変更が適用されます。全てのビューに変更を適用チェックボックスをクリックした後の変更だけが全てのビューに対して適用される点に注意してください。
- ツールバーを新たに追加: 新規ボタンをクリックして、表示されるツールバーの名前ダイアログにてツールバーの名前を入力します。作成されたツールバーにはコマンド[1439]タブからコマンドをドラッグすることができます。
- 追加されたツールバーの名前を変更: 追加されたツールバーをツールバーペインにて選択し、名前の変更ボタンをクリックします。表示されるツールバーの名前ダイアログにて名前の編集を行います。
- メニューバーのリセット: ツールバーペインにてメニューバーアイテムを選択し、リセットをクリックします。この操作によりアプリケーションがインストールされた初期の状態にメニューバーがリセットされます。
- ツールバーとメニューコマンドのリセット: 全てリセットボタンをクリックすることで、アプリケーションがインストールされた初期の状態に、全てのツールバーならびにメニューがリセットされます。
- ツールバーの削除: ツールバーペインにて削除するツールバーを選択し、削除ボタンをクリックします。
- 特定のツールバーにコマンドのテキストラベルを表示: 目的のツールバーを選択し、テキストラベルを表示チェックボックスにチェックを入れます。テキストラベルは各ビューに対して個別に有効化する必要がある点に注意してください。

28.17.11.3 ツール

ツールタブでは、XMLSpy 内部から外部アプリケーションを使用するコマンドをセットアップすることができます。これらのコマンドは、「ツール | ユーザー定義ツール」メニュー以下に表示されます。例えば、作成されたツールメニューのコマンドを「ツール | ユーザー定義ツール」以下からクリックすることで、XMLSpy のメインウィンドウに表示されているアクティブなファイルを、メモ帳のような外部アプリケーションにて開くことができます。



以下の操作により外部アプリケーションを使用するコマンドをセットアップすることができます:

1. メニューコンテンツペインにて、ペインのタイトルバーにある新規作成アイコンをクリックし、作成されたアイテムにメニューコマンドの名前を入力します。上のスクリーンショットでは、メモ帳で開くというメニューコマンドが作成されており、アクティブなドキュメントを外部アプリケーションの(Windows に搭載されている)メモ帳にて開くようなコマンドを作成します。新規作成アイコンをクリックすることで、更にコマンドを追加することもできます。アイテムを上へならびにアイテムを下へアイコンを使用することで、コマンドの相対的な位置を変更することができます。コマンドを削除するには、削除アイコンをクリックしてください。
2. 外部アプリケーションとコマンドを関連付けるには、メニューコンテンツペインにあるコマンドを選択し、コマンドフィールドにて外部アプリケーションの実行可能ファイルへのパスを入力するか、(入力フィールドの隣にあるボタンから)参照してください。上のスクリーンショットでは、メモ帳へのパスがコマンドフィールドへ入力されています。
3. 引数フィールドにあるボタンをクリックすることで、外部アプリケーションに対して利用可能なアクションが表示されます(上のスクリーンショットを参照)。これらのアクションについては以下にあるリストを参照ください。アクションを選択すると、そのアクションに対応したコード文字列が引数フィールドに入力されます。
4. 処理を行う際の初期ディレクトリを指定する場合、その値を初期ディレクトリフィールドに入力してください。
5. 閉じるボタンをクリックすることで処理を完了します。

作成されたコマンドは、「ツール | ユーザー定義ツール」メニューと、プロジェクトウィンドウにてファイルを右クリックすることで表示されるコンテキストメニューの「ユーザー定義ツール」サブメニュー以下に表示されます。

新規作成されたコマンドをツールメニューにてクリックすると、コマンドに関連付けられたアクションが実行されます。上のスクリーンショットに示されたコマンドの例の場合、XMLSpy のメインウィンドウでアクティブなドキュメントがメモ帳で開かれます。外部アプリケーションコマンドへは、プロジェクトウィンドウのユーザーツールコンテキストメニューからもアクセスすることができます(プロジェクトウィンドウにあるファイルを右クリックすることで

(C) 2015-2021 Altova GmbH

コンテキストメニューが表示されます)。プロジェクトウィンドウでは、複数のファイルを選択し、コンテキストメニューからコマンドを選択することで、複数のファイルに対して関連付けられたアクションを実行することができます。

引数

引数フィールドでは外部アプリケーションコマンドにより実行されるアクションが指定されます。以下の引数を利用することができます:

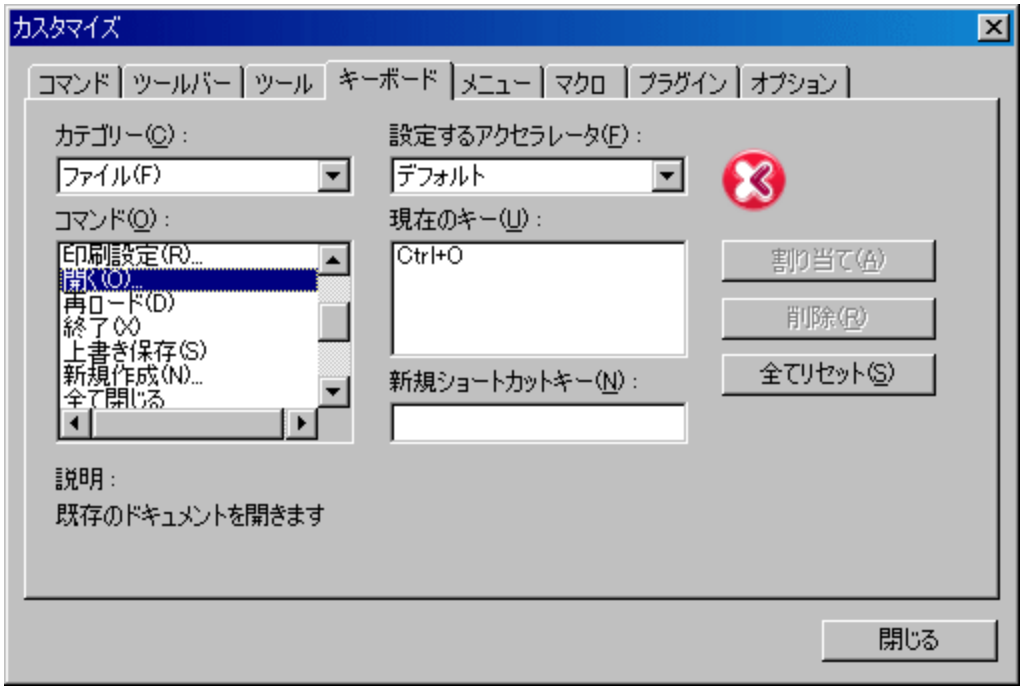
- アクティブなドキュメントのファイルパス:「ユーザー定義ツール」メニューにあるコマンドから、XMLSpy にてアクティブになっているドキュメントを外部アプリケーションで開きます。プロジェクトウィンドウのコンテキストメニューからコマンドへアクセスした場合、選択されたファイルが外部アプリケーションで開かれます。
- プロジェクトファイルパス: XMLSpy のプロジェクトファイル(.spp ファイル)を外部アプリケーションにて開きます。

初期ディレクトリ

オプションとして、外部アプリケーションを実行する際にカレントディレクトリとして使用されるディレクトリへのパスを、**初期ディレクトリ**フィールドへ入力することができます。

28.17.11.4 キーボード

キーボードタブでは、キーボードショートカットを新たに作成したり、既存のショートカットを変更することができます。



以下の操作により、新たなショートカットをコマンドへ割り当てる、または既存のショートカットを変更することができます:

1. **カテゴリー**コンボボックスから**全てのコマンド**を選択します。関連付けられたコマンドとしてマクロが選択された[1448]場合、カテゴリーコンボボックスにマクロをセットすることもできます。
2. コマンドリストボックスにて、ショートカットを割り当てる、またはショートカットを変更するコマンドを選択します。
3. 新規ショートカットキーテキストボックスをクリックし、そのコマンドに対して割り当てるショートカットキーを押下します。入力されたショートカットが新規ショートカットキーテキストボックスに表示されます。そのショートカットがどのコマンドにも割り当てられていない場合、

割り当てボタンが選択可能状態になります。ショートカットが他のコマンドに既に割り当てられている場合、そのコマンド名がテキストボックス以下に表示され、**割り当て**ボタンが選択不可能状態になります。**新規ショートカットキー**テキストボックスに入力された内容をクリアするには、Ctrl、Alt、またはShift のいずれかのキーを押下してください。

4. **割り当て**ボタンをクリックすることで、ショートカットの割り当てを行うことができます。ショートカットが現在のキーリストボックスに表示されます。複数のショートカットを同じコマンドへ割り当てることもできます。
5. **閉じる**ボタンをクリックすることで、変更を確定します。

ショートカットの削除

同じショートカットを複数のコマンドに対して割り当てることはできません。ショートカットを削除するには、現在のキーリストボックスにあるショートカットを選択し、**削除**ボタンをクリックし、**閉じる**ボタンをクリックすることで削除を確定してください。

設定するアクセラレータ

現在割り当てられている機能はありません。

デフォルトのキーボードショートカット

インストール時に割り当てられているコマンドのリストを(機能ごとに)以下に示します: アプリケーションではキーボードマップダイアログ(「ヘルプ | キーボードマップ[1497]」)では、ショートカットとその説明とともに、コマンドのリストを確認することができます。

関数キーのショートカット (検証と変換のためのショートカットも含む)機能キーのショートカット

F1	ヘルプメニュー
F1 + Alt	最後に開かれたファイルを開く
F3	次を検索
F4 + CTRL	アクティブなウィンドウを閉じる
F4 + Alt	XMLSpyを閉じる
F5	更新
F6 + CTRL	開かれているウィンドウを循環する
F7	整形式チェック
F8	検証
F10	XSL 変換
F10 + CTRL	XSL:FO 変換

ファイルとアプリケーションコマンド

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

その他のキー

(C) 2015-2021 Altova GmbH

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

編集コマンド

CTRL + A	すべて選択
CTRL + F	検索
CTRL + G	ライン/文字に移動する
CTRL + H	置き換える
CTRL + V	貼り付け
CTRL + X	切り取り
CTRL + Y	やり直し
CTRL + Z	元に戻す

テキストビューコマンド

CTRL + E	開始/終了タグ間を移動する
CTRL + Shift + E	カーソルを含む要素を選択する
CTRL + Alt + E	親要素に移動する
CTRL + "+"	ズームイン
CTRL + "-"	ズームアウト
CTRL + 0	ズームのリセット
CTRL + マウスホイールを前にロールする	ズームイン
CTRL + マウスホイールを後ろにロールする	ズームアウト

グリッドビューコマンド

CTRL + D	CDATA の追加
CTRL + E	要素の追加
CTRL + I	属性の追加
CTRL + M	コメントの追加
CTRL + T	テキストの追加
CTRL + Shift + D	CDATA の挿入
CTRL + Shift + E	要素の挿入
CTRL + Shift + I	属性の挿入
CTRL + Shift + M	コメントの挿入

CTRL + Shift + T	テキストの挿入
CTRL + Alt + D	子 CDATA の追加
CTRL + Alt + E	子要素の追加
CTRL + Alt + I	子属性の追加
CTRL + Alt + M	子コメントの追加
CTRL + Alt + T	子テキストの追加

スキーマビューコマンド

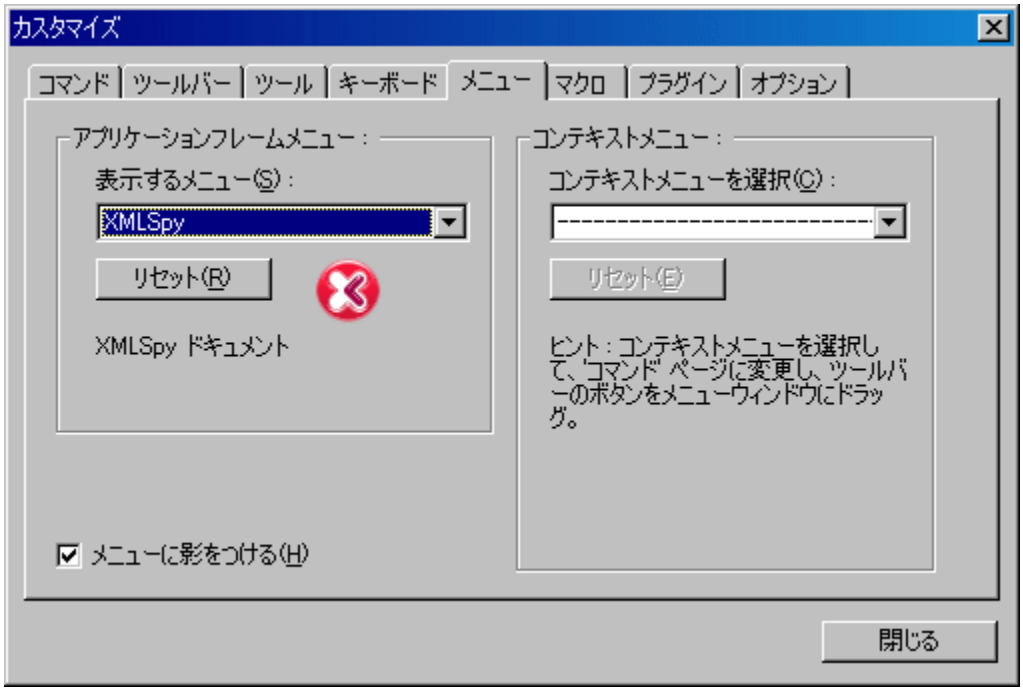
CTRL + 要素をダブルクリック	要素の定義の表示

デバッガーコマンド

F9	ブレークポイントの挿入/削除
F9 + Shift	トレースポイントの挿入/削除
F9 + CTRL	ブレークポイントの有効化/無効化
F9 + Shift + CTRL	トレースポイントの有効化/無効化
F11	ステップイン
F11 + Shift	ステップアウト
F11 + CTRL	ステップオーバー
F11 + Alt	デバッガーを開始する

28.17.11.5 メニュー

メニュータブでは、デフォルトならびにアプリケーションメニューバーという2つのメインメニューバー、そしてアプリケーションのコンテキストメニューをカスタマイズすることができます。



デフォルトメニューバーとアプリケーションメニューバーのカスタマイズ

デフォルトのメニューバーとは、メインウィンドウにてドキュメントが開かれていない状態で表示されるメニューバーのことです。アプリケーションメニューバーとは、1つ以上のドキュメントがメインウィンドウにて開かれている状態で表示されるメニューバーのことです。各メニューバーは、お互いに干渉しないかたちで個別にカスタマイズすることができます。

メニューバーをカスタマイズするには、「表示するメニュー」コンボボックスから目的のメニューを選択し、カスタマイズダイアログのコマンドタブ[1439]へ切り替えます。その後コマンドリストボックスに表示されたコマンドを、メニューバーまたはメニューへドラッグします。

コマンドの削除とメニューバーのリセット

メニュー内のメニュー全体またはコマンドを[削除]するには以下を行います:

1. アプリケーションフレームメニューペイン内で *デフォルト*（ドキュメントが開かれていない場合）または（1つ以上のドキュメントが開かれている場合使用可能なメニューを表示する）*XMLSpy* を選択します。
2. 開かれているカスタマイズダイアログから（i）アプリケーションのメニューバーから削除するコマンド（ii）これらのメニューから削除するコマンドを選択します。
3. （i）メニューバーからメニューをドラッグ、または、メニューからメニューコマンドをドラッグ、または（ii）メニューまたはメニューコマンドを右クリックして[削除]を選択します。

「*表示するメニュー*」コンボボックスにてメニューを選択し、「リセット」ボタンをクリックすることで、2つのメニューバー（デフォルトならびにアプリケーションメニューバー）のそれぞれをリセットすることができます。

アプリケーションのコンテキストメニューをカスタマイズする

コンテキストメニューとは、アプリケーションインターフェースに表示されたオブジェクトを右クリックすることで表示されるメニューのことです。以下の方法により、コンテキストメニューのカスタマイズを行うことができます:

1. コンテキストメニューを選択コンボボックスにて目的のコンテキストメニューを選択します。コンテキストメニューが表示されます。

2. カスタマイズダイアログのコマンドタブ[1439]へ切り替えます。
3. コマンドリストボックスからコンテキストメニューへコマンドをドラッグします。
4. コンテキストメニューからコマンドを削除するには、コンテキストメニューに表示されたコマンドを右クリックして、「削除」を選択します。コンテキストメニューからコマンドをドラッグすることでも、そのコマンドを削除することができます。

コンテキストメニューを選択コンボボックスにてメニューを選択し、「リセット」ボタンをクリックすることで、コンテキストメニューをオリジナルの状態にリセットすることができます。

メニューに影を付ける

「メニューに影を付ける」チェックボックスを有効にすることで、全てのメニューに影が与えられます。

28.17.11.6 マクロ

マクロタブでは、XMLSpy のスクリプトエディターにより作成されたマクロに対するアプリケーションコマンドを作成することができます。関連付けられたマクロを実行するこれらアプリケーションコマンドは、マクロタブから直接、またはカスタマイズダイアログのコマンドタブ[1439]を使用することにより、メニューやツールバーにて利用することができるようになります。アプリケーションコマンドとして、カスタマイズダイアログのキーボードタブ[1443]にてショートカットを割り当てることもできます。

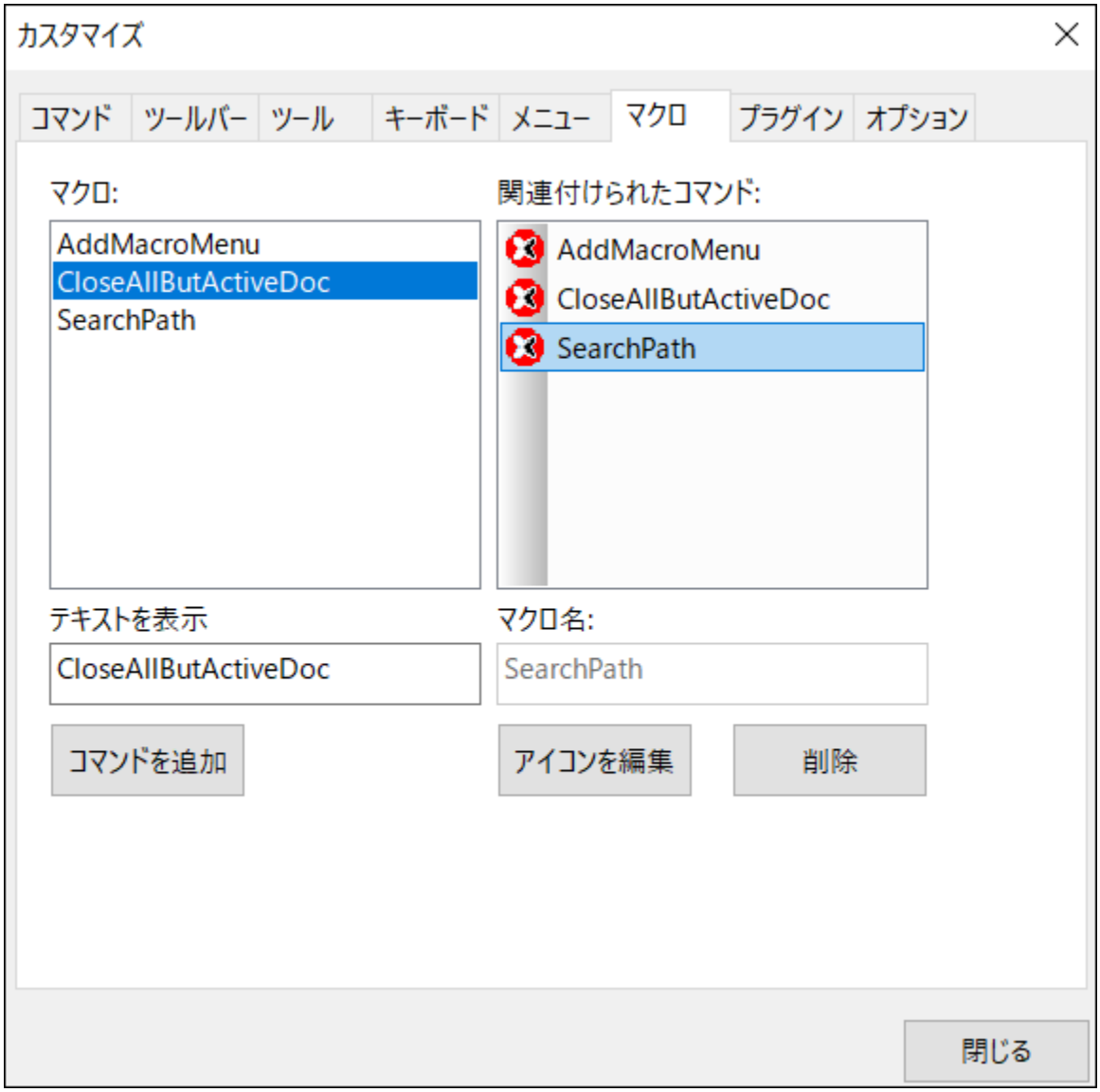
XMLSpy におけるマクロの作動

XMLSpy におけるマクロは以下のように動作します:

- XMLSpy のスクリプトエディター[1505]により、Altova スクリプトプロジェクト(.asprj ファイル)が作成されます。スクリプトプロジェクトには、XMLSpy で使用することのできるマクロを含めることができます。
- XMLSpy では、(i) オプションダイアログのスクリプトタブ[1488]にて指定されるアプリケーションプロジェクトと、(ii) スクリプト設定ダイアログ[1194](プロジェクト | スクリプト設定[1194])内で指定されるアクティブな XMLSpy プロジェクト[950]スクリプトプロジェクト。
- これら2つのスクリプトプロジェクト内にあるマクロは「ツール | マクロ」メニューから利用することができ、カスタマイズダイアログのマクロタブではマクロをアプリケーションコマンドとしてセットすることができます(以下のスクリーンショットを参照)。マクロをアプリケーションコマンドとしてセットすると、メニューやツールバーにそのコマンドを配置することができるようになります。

マクロに対してアプリケーションコマンドを作成する

スクリプトエディター[1505](ツール | スクリプトエディター[1505]」)にて目的のマクロを作成し、スクリプトプロジェクトとして保存します。このファイルを(オプションダイアログのスクリプトタブ[1488]から)アプリケーションスクリプトプロジェクトとして、または(アプリケーションプロジェクトのスクリプト設定ダイアログ[1194](プロジェクト | スクリプト設定[1194]」)から)アクティブなアプリケーションプロジェクトのスクリプトプロジェクトとして指定することができます。スクリプト内部にあるマクロがマクロタブの「マクロ」ペインに表示されます(以下のスクリーンショットを参照)。
マクロに対するアプリケーションコマンドを作成するには、「マクロ」ペインにてマクロを選択し、「テキストを表示」テキストボックスにてコマンドのテキストを入力した後に、「コマンドを追加」ボタンをクリックします(以下のスクリーンショットを参照)。選択されたマクロに関連付けられたコマンドが、「関連付けられたコマンド」リストボックスに追加されます。



関連付けられたコマンドのアイコンを編集するには、そのコマンドが選択された状態で**アイコンを編集**ボタンをクリックします。**削除**ボタンをクリックすることで、関連付けられたコマンドを削除することができます。

マクロに関連付けられたコマンドをメニューやツールバーに配置する

マクロに関連付けられたコマンドをメニューやツールバーに配置するには、2つの方法があります:

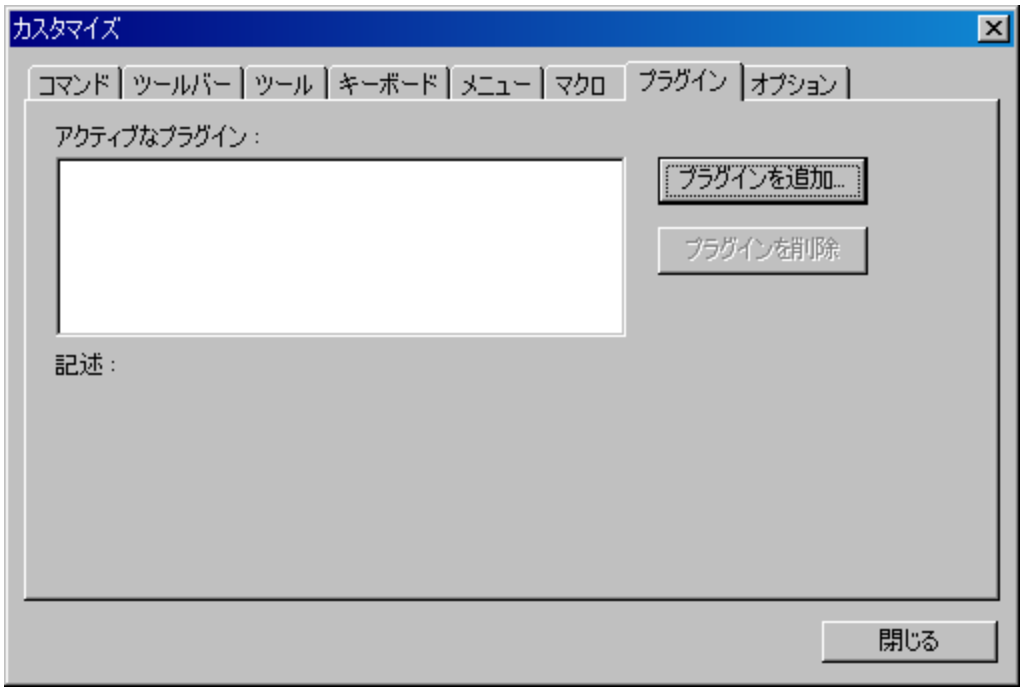
- 関連付けられたコマンドリストボックスから、メニューやツールバー上にある目的の場所へコマンドをドラッグする。
- カスタマイズダイアログのコマンドタブ[1439]に用意されているメカニズムを使用する。

どちらの場合でも、指定された場所にコマンドが作成されます。メニューまたはツールバー上にあるコマンドをクリックすることで、関連付けられたマクロが実行されます。

メモ: 関連付けられたコマンドとしてマクロがセットされた場合、キーボードのショートカット[1443]を割り当てることができます。カスタマイズダイアログのキーボードタブ[1443]にて、カテゴリーコンボボックスからマクロを選択し、目的のマクロを選択した状態でショートカットをセットしてください。

28.17.11.7 プラグイン

「プラグイン」タブでは、アプリケーションやツールバー動作するようプログラムされたプラグインの統合とコマンドの配置を行うことができます。(*下のスクリーンショット*) プラグインタブでは、「プラグインを追加」をクリックすることで、プラグインのDLL ファイルを参照することができます(*下の「プラグインの作成」を参照してください*)。「OK」をクリックすることでプラグインが追加されます。アプリケーションには複数のプラグインを追加することができます。



プラグインの追加に成功すると、プラグインの説明がダイアログに表示され、プラグインを削除ボタンが選択可能になります。プラグインコードによりツールバーやメニューが作成されると、それらがアプリケーションインターフェースに表示されます。プラグインを削除するには、プラグインを削除ボタンをクリックしてください。

プラグインの作成

サンプルプラグインのソースコードはアプリケーションのマイドキュメント フォルダー[33]マイドキュメント フォルダー: `Examples\IDEPlugin` フォルダー内に収められています。ソースコードからプラグインを作成するには以下を行います:

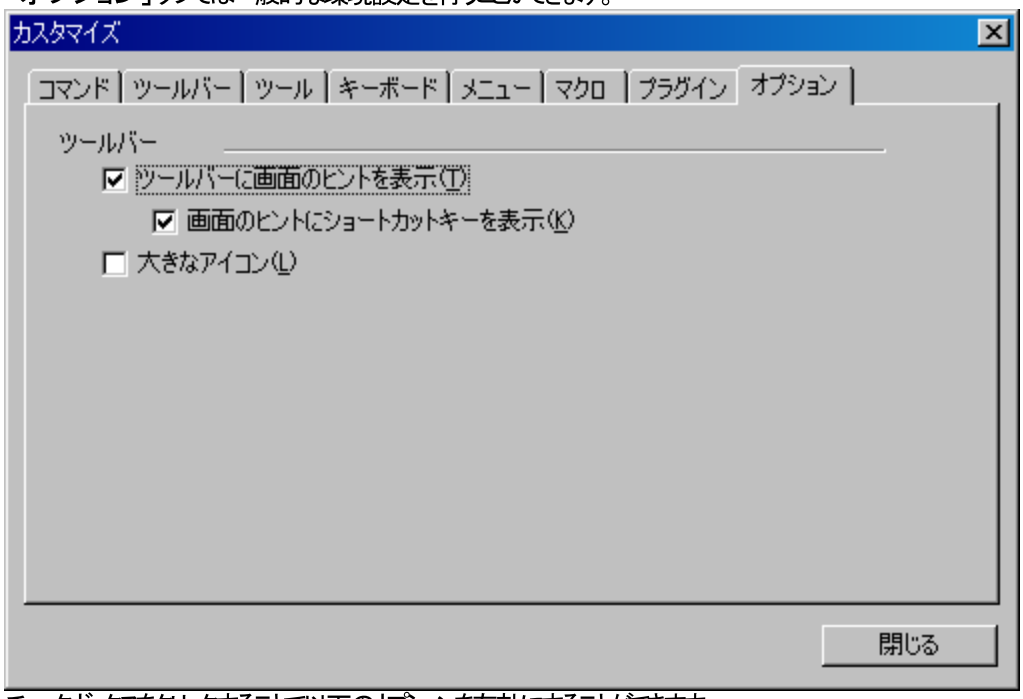
1. Visual Studio 内でプラグインとして作成するソリューションを開きます。
2. ビルドメニュー内のコマンドを使用してプラグインを作成します。
3. プラグインのDLLファイルは`Bin` または`Debug` フォルダー内で作成されます。このDLL ファイルは、プラグインとして追加される必要があるファイルです(*上記参照*)。

プラグインの作成に関する詳細については、IDE プラグイン[1530]のセクションを参照ください。

(C) 2015-2021 Altova GmbH

28.17.11.8 オプション

「**オプション**」タブでは一般的な環境設定を行うことができます。

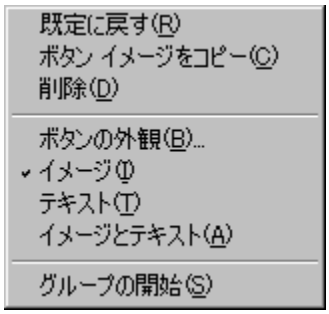


チェックボックスをクリックすることで以下のオプションを有効にすることができます:

- **ツールバーに画面のヒントを表示**: ツールバーのアイコン上にマウスポインターが配置されると、ポップアップが表示されます。ポップアップにはアイコン機能の説明と、(**ショートカットキーを表示**オプションが有効になっていれば) 関連付けられたショートカットが表示されます。
- **画面のヒントにショートカットキーを表示**: 画面のヒントにショートカットキーに関する情報を表示するか指定することができます。
- **大きなアイコン**: ツールバーアイコンの大きさを通常のサイズと大きめのサイズへ切り替えることができます。

28.17.11.9 コンテキストメニューのカスタマイズ

コンテキストメニューのカスタマイズコマンドは、カスタマイズダイアログが表示されている状態で、メニュー、メニューコマンド、またはツールバーアイコンを右クリックすることにより表示されます。

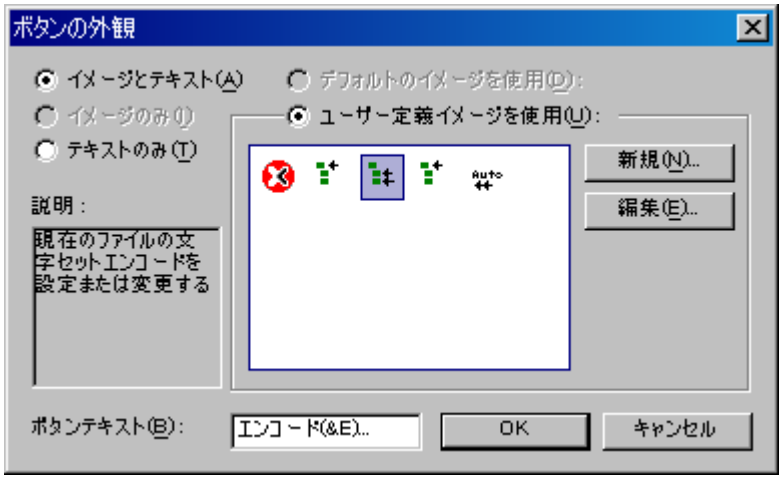


以下の機能を利用することができます:

- 規定に戻す: 現在機能は与えられていません。
- ボタンイメージをコピー: 右クリックしたアイコンをクリップボードにコピーします。
- 削除: 選択されたメニュー、メニューコマンド、またはツールバーアイコンを削除します。削除されたアイコンを再度表示する方法については、下記を参照ください。
- ボタンの外観: ボタンの外観ダイアログが表示され、選択されたツールバーアイコンの外観を定義するためのプロパティをセットすることができます。詳細については以下の記述を参照ください。
- イメージ、テキスト、イメージとテキスト: 相互に排他的なオプションで、選択されたツールバーアイコンをアイコンだけ、テキストだけ、またはアイコンとテキストの両方で表示するかを指定します。いずれかのオプションを選択することで、必要な変更が行われます。
- グループの開始: 選択されたツールバーアイコンの左側にグループを分離する垂直の線を挿入します。これにより選択されたツールバーアイコンがグループ最初のアイコンとなります。

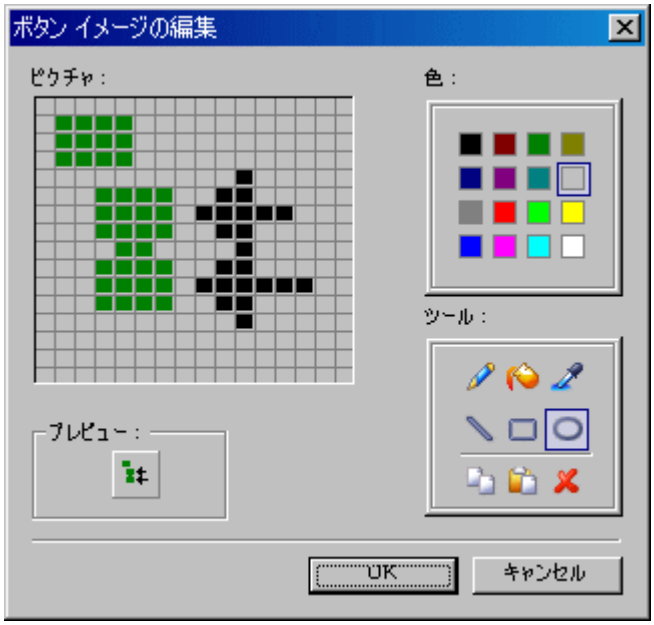
ボタンの外観ダイアログ

ツールバーアイコンを右クリックして、「**ボタンの外観**」を選択することで、ボタンの外観ダイアログが表示されます(以下のスクリーンショットを参照)。このダイアログでは、ツールバーアイコンのイメージやテキストを選択することができます。現在このダイアログでは、マクロならびにプラグインのツールバーアイコンだけを編集することができます。



選択されたツールバーのアイコン(**コンテキストメニューのカスタマイズ**にて右クリックされたアイコン)に対して以下の編集機能を使用することができます。

- イメージのみ、テキストのみ、イメージとテキスト: 目的のラジオボタンを選択して、ツールバーのアイコンが表示される形式を指定することができます。
- イメージの編集: イメージのみまたはイメージとテキストが選択された場合、イメージの編集オプションが有効になります。「新規」ボタンをクリックすることでユーザー定義の新たなイメージが作成され、イメージペインに追加されます。イメージを選択して、「編集」ボタンにより編集を行うことができます。



- イメージ選択: イメージペインから目的のイメージを選択し、OK をクリックすることにより選択されたイメージをツールバーで使用することができます。
- テキストの編集とセレクター: テキストのみまたはイメージとテキストが選択された場合、ボタンテキストテキストボックスが選択可能になります。テキストを入力または編集し、OK をクリックすることで、ツールバーアイコンのテキストを変更することができます。

メモ ボタンの外観ダイアログでは、メニューコマンドのテキストを編集することもできます。カスタマイズダイアログが開かれている状態でメニューコマンドを右クリックして、「**ボタンの外観**」を選択し、「**ボタンテキスト**」テキストボックスにてメニューコマンドのテキストを編集してください。

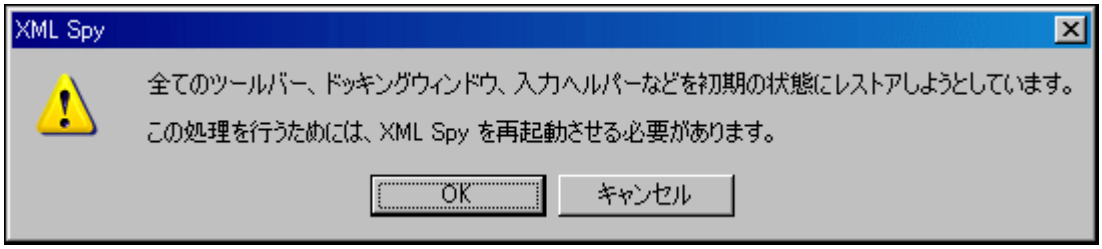
削除されたメニュー、メニューコマンド、ツールバーコマンドを復元

コンテキストメニューのカスタマイズにある**削除**コマンドにより、メニューやメニューコマンド、ツールバーアイコンが削除された場合、以下の方法により復元することができます:

- メニュー: メニューオプションから「ツール| カスタマイズ| メニュー[1446]」を選択し、アプリケーションフレームメニューペインにあるリセットボタンをクリックしてください。他にも「ツール| カスタマイズ| ツールバー[1440]」を選択し、ダイアログ左側でメニューバーを選択した状態でリセットボタンをクリックすることもできます。
- メニューコマンド: メニューオプションから「ツール| カスタマイズ| コマンド[1439]」を選択し、コマンドリストボックスから目的のコマンドをメニューへドラッグしてください。.
- ツールバーアイコン: メニューオプションから「 ツール| カスタマイズ| コマンド[1439]」を選択し、コマンドリストボックスから目的のコマンドをツールバーへドラッグしてください。

28.17.12 ツールバーとウィンドウの復元

ツールバーとウィンドウの復元コマンドにより、XMLSpy をデフォルト設定の状態で再起動することができます。終了する前に XMLSpy の終了を確認するダイアログが表示されます(以下のスクリーンショットを参照)。

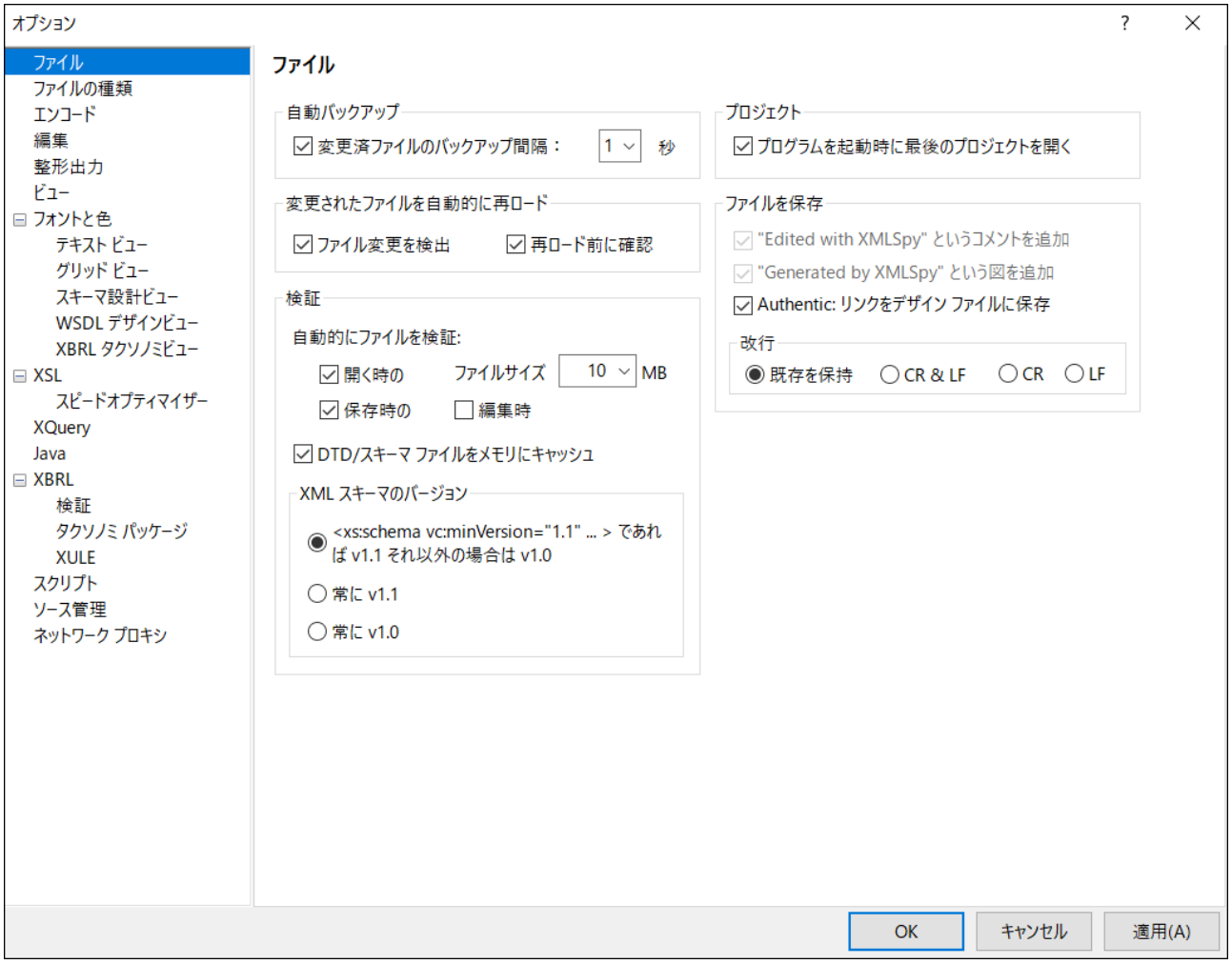


このコマンドは、ツールバーやウィンドウのサイズ変更、移動、または非表示にするように指定した後、ツールバーやウィンドウをオリジナルの状態に戻したい場合に使用することができます。

28.17.13 オプション

[**ツール | オプション**] コマンドによりグローバルなアプリケーション設定を定義することができます。これらの設定は、セクションにより整理されています（*下のスクリーンショット*）。例えば、ファイル セクション 1455（*下のスクリーンショットを参照*）には、XMLSpy によるファイルの開き方、及び、保存の方法を指定することのできるオプションが含まれています。特定のセクションのオプションを指定するには、左側のペイン内のそのセクションを選択し、必要とするプロパティの値を指定します。「**OK**」ボタンにより、レジストリへの変更が保存され、ダイアログが閉じられます。「**適用**」ボタンにより現在開かれているドキュメントへの変更を表示することができます。

(C) 2015-2021 Altova GmbH



オプションダイアログの各セクション詳細については、このセクションのサブセクションを参照ください。

28.17.13.1 ファイル

ファイルセクションでは、XMLSpy にてファイルを開く、または保存する際の振る舞いを定義することができます。エンコードセクション[1460]にも関連する設定が存在します。

自動的なバックアップ

このオプションが有効化されていると現在編集中のファイルは自動的にバックアップされます。5 秒から 60 秒までの間のバックアップの間隔をコンボボックスから選択、または 300 秒までのカスタム値を入力することができます。詳細に関しては、*ファイルの自動的なバックアップ* 128 のセクションを参照してください。

変更されたファイルの自動的な再ロード

マルチユーザー環境内で作業する場合、または、サーバー上で動的に生成されるファイルと作業する場合、インターフェイス内で現在開かれているファイルへの変更を確認することができます。XMLSpy が開かれているドキュメント内の変更を検知する都度、変更されたファイルを再ロードするかプロンプトされます。

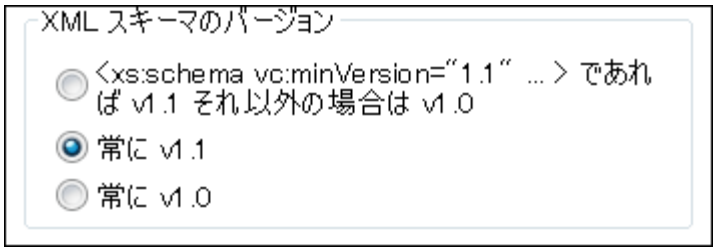
検証

DTD やスキーマにより XML ドキュメントの構造を定義している場合、ドキュメントを開くまたは保存する際に検証を行うかどうか指定することができます。自動的に行われる検証は、ドキュメントの上限を定めることで、効果的な編集作業を行うことができます。ドキュメントが妥当でない場合、エラーメッセージが表示されます。ドキュメントが妥当な場合は、メッセージが出力されることなく、処理が行われます。XMLSpy ではこれらのファイルをメモリ上にキャッシュすることで、(例えば、参照されているスキーマが URL によりアクセスされた場合など) 不必要な再ロードを抑えることができます。スキーマ位置の宣言に URL を使用している場合、「DTD/スキーマファイルをメモリにキャッシュ」オプションを無効にすることで、スキーマに対して行われた変更を、キャッシュを介さず直接保存することができます。

編集中 オプションが選択されているとテキストビュー[130]とJSON グリッドビュー[610]での入力中に検証が実行されます。このオプションはデフォルトで選択されています。詳細に関してはテキストビュー内でのXML 検証[302]を参照してください。

XML スキーマバージョン

スキーマビュー内で有効化されたXSD モードは以下により異なります(i) 属性の存在/不在—存在する場合、XSD ドキュメントの`/xs:schema/@vc:minVersion` 属性 の値(ii) オプションダイアログのファイルセクション(「**ツール | オプション**」内で選択されるXML スキーマバージョンのオプション。*してのスクリーンショット*)。



以下のシチュエーションが可能になります。下のテーブル内 *XML スキーマ バージョン*は、上に示されるXML スキーマ バージョン ペイン内の選択を参照しています。テーブル内の`vc:minVersion` 値は、XML スキーマドキュメント内の`xs:schema/@vc:minVersion` 属性の値を参照しています。詳細に関しては、編集ビュー| スキーマビュー| XSDMode [161] のセクションを参照してください。

XML スキーマバージョン	`vc:minVersion` 属性	XSD モード
常にv1.0	不在、または、存在していても値を持っていない	1.0
常にv1.1	不在、または、存在していても値を持っていない	1.1
@ vc:minVersion の値	属性は、`1.1` の値を持っている	1.1
@ vc:minVersion の値	属性が不在、または、存在していても値が`1.1` 以外	1.0

プロジェクト

XMLSpy を起動すると、最後に使用されたプロジェクトが自動的に開くように指定することができます。

ファイルを保存

XML ドキュメントを保存する際、XMLSpy により、`<!-- Edited with XMLSpy http://www.altova.com -->` というコメントが、ドキュメントの上部に挿入されます。このオプションを無効にするにはユーザーのライセンス登録が必要で、拡張グリッドビューまたはスキーマデザインビューにて編集を行い、ファイルを保存した際に有効になります。

(「**スキーマ設計 | ドキュメンテーションの生成**」により)コンテンツモデルダイアグラムを保存する際、XMLSpy によりXMLSpy のロゴが挿入されます。このオプションを無効にするには、ライセンスの登録が必要になります。

XML ファイルにStyleVision Power Stylesheet が関連付けられている場合、「Authentic：リンクをデザインファイルに保存」オプションを有効にすることで、XML とともにStyleVision Power Stylesheet を保存することができます。

改行

既存を保持が選択されていると、ファイルを開いたときに、ドキュメント内にあった改行コード文字がそのまま保持されます。または、3つのロード内のロードの変更を選択することができます: **CR&LF** (PC)、**CR** (MacOS)、または**LF** (Unix)。

保存して完了

設定が完了したら、「OK」をクリックして完了します。

28.17.13.2 ファイルタイプ

ファイルタイプセクションでは、各種ファイルに対するXMLSpy の振る舞いを指定することができます。

ファイルタイプリストボックスからファイルの種類を選択し、その種類のファイルに対する動作をカスタマイズすることができます。

Windows エクスプローラー設定

Windows エクスプローラーで使用されるファイルタイプの説明と、MIMIE 準拠のコンテンツタイプを定義するほか、この種類のドキュメントを編集するデフォルトエディターとしてXMLSpy を指定することができます。

準拠

XMLSpy では、様々な種類のファイルに対して使用することのできる数多くの機能が搭載されています。特定のファイルタイプに対して使用されるオプションの種類は、この準拠オプションによりセットされます。XMLSpy ではファイルタイプをXML, XQuery 442、ZIP 840、JSON 587、Avro 675そしてその他のテキスト形式に準拠するよう設定することができるほか、XML に準拠したファイルも、XML、DTD、そしてXML エンティティファイルタイプと細分化することができます。JSON 準拠ファイルはプレーンJSON またはAvro スキーマに従い区別されます。*Avro 準拠 オプション*はAvro バイナリファイルのためのサポートを呼び出します。ファイルの型の大半はファイル型に適切なデフォルトの準拠により定義されています。新規のファイル型を追加、または、準拠の他の型にファイルを設定することを希望する場合を除いてこれらの設定を変更しないことが奨励されます。

(C) 2015-2021 Altova GmbH

デフォルトビュー

このグループでは、各ファイルの種類に対して使用されるデフォルトのビューを指定することができます。特定の準拠が特定のビューでのみ確認できる場合、ビューはデフォルトで選択されており、ビューセクションは無効化されています。例えば、XQuery Update ドキュメントはテキストビューでのみ確認することができ、このビューはデフォルトで選択されており、ビューセクションは無効化されています。同様に、Avro 準拠ドキュメント（Avro バイナリ）はグリッドビューでのみ確認することができます。

グリッドビュー

このチェックボックスにより、グリッドビューで自動的にテーブル表示が使用されるか指定することができます。

テキストビュー

このチェックボックスにより、特定の種類のファイルに対して、構文の色分けを行うよう指定することができます。

自動検証を無効化

このオプションにより、各ファイルに対して自動検証を有効にすることができます。自動検証は通常、ファイルが開かれたとき、またはビューが切り替えられたときに行われます。

空の要素を省略して <E /> で保存する

XML ドキュメントや、XML ドキュメントにより出力されるドキュメントを使用するアプリケーションの中には、XML 1.0 仕様により定義されている（<Element/> という形式の）空要素が原因でエラーを起こすことがあります。このオプションを無効にすることで、XMLSpy は、<Element></Element> という形式で要素の保存を行います。

RaptorXML Server を使用して開く/保存時に検証する

ファイが開かれて保存される際に選択されたファイル型のファイルを検証するために RaptorXML Server が使用されるかを指定します。
RaptorXML Server はセットアップされ、構成されている必要があります。 1434

新たなファイル拡張子を追加

新しいファイルタイプを、ファイルタイプリストに追加します。ファイルタイプを追加した後には、このタブにあるオプションを使用することで、このファイルタイプの定義を行う必要があります。

選択されたファイル拡張子を削除

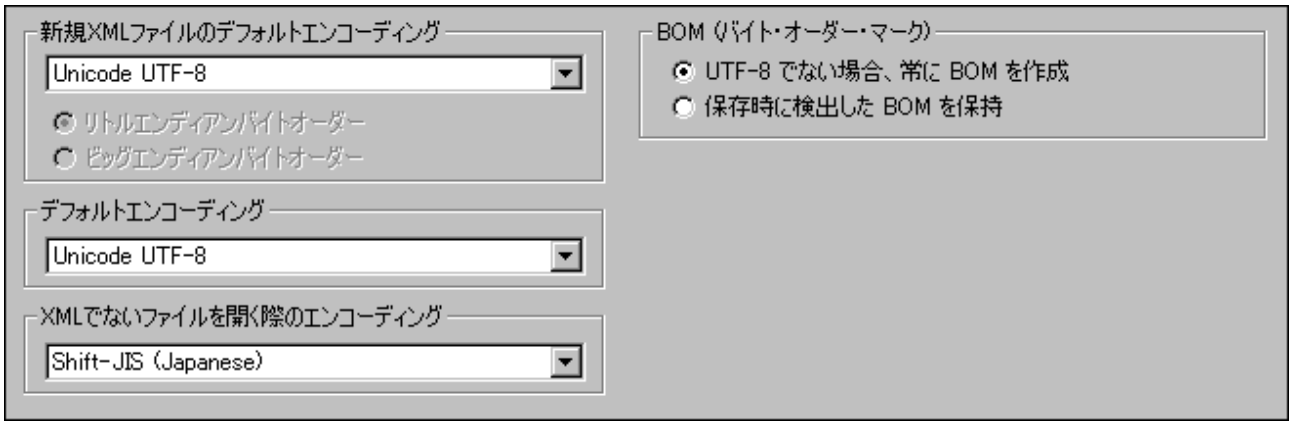
現在選択されているファイルタイプと、全ての関連する設定を削除します。

保存して完了する

設定が完了したら、「OK」をクリックして完了します。

28.17.13.3 エンコード

エンコード セクションでは、ファイルのエンコーディングを指定することができます。



新規 XML ファイルのデフォルト エンコーディング

ドロップダウン リストから新規 XML ファイルで使用されるデフォルト エンコーディングを選択することができます。ここで指定したエンコーディングが、新たなドキュメントを作成する際、XML 宣言にて使用されます。デフォルト エンコーディングに 2 バイトまたは 4 バイトのエンコーディング (UTF-16、UCS-2 または UCS-4) をデフォルト エンコーディングとして選択した場合、リトル エンディアン、またはビッグ エンディアンを選択することができます。

既存の XML ファイルにおけるエンコーディングはそのまま保持されますが、「 ファイル | エンコード 1137 」コマンドにより変更することができます。

XML ファイルを未知のエンコードを使用して開く

XML ファイルのエンコーディングを決定することができない、または XML ドキュメントにてエンコーディングの指定がなされていない場合、このコンボボックスで選択したエンコーディングがファイルを開く際に使用されます。.

XML でないファイルを開く際のエンコーディング

既存の、そして新規に作成された XML 以外のファイルは、このコンボボックスで選択したエンコーディングにより開かれます。ドキュメントのエンコーディングは、「 ファイル | エンコード 1137 」コマンドにより変更することができます。

BOM (バイト・オーダー・マーク)

2 バイトまたは 4 バイトのエンコーディングによりドキュメントを保存する場合、ドキュメントの保存を、(i) リトル エンディアン オーダー、または リトル エンディアン BOM により行うか (**UTF-8 でない場合、常に BOM を作成**)、(ii) 検出された バイト オーダー または 検出された BOM により行うか (**保存時に検出した BOM を保持**) 指定することができます。

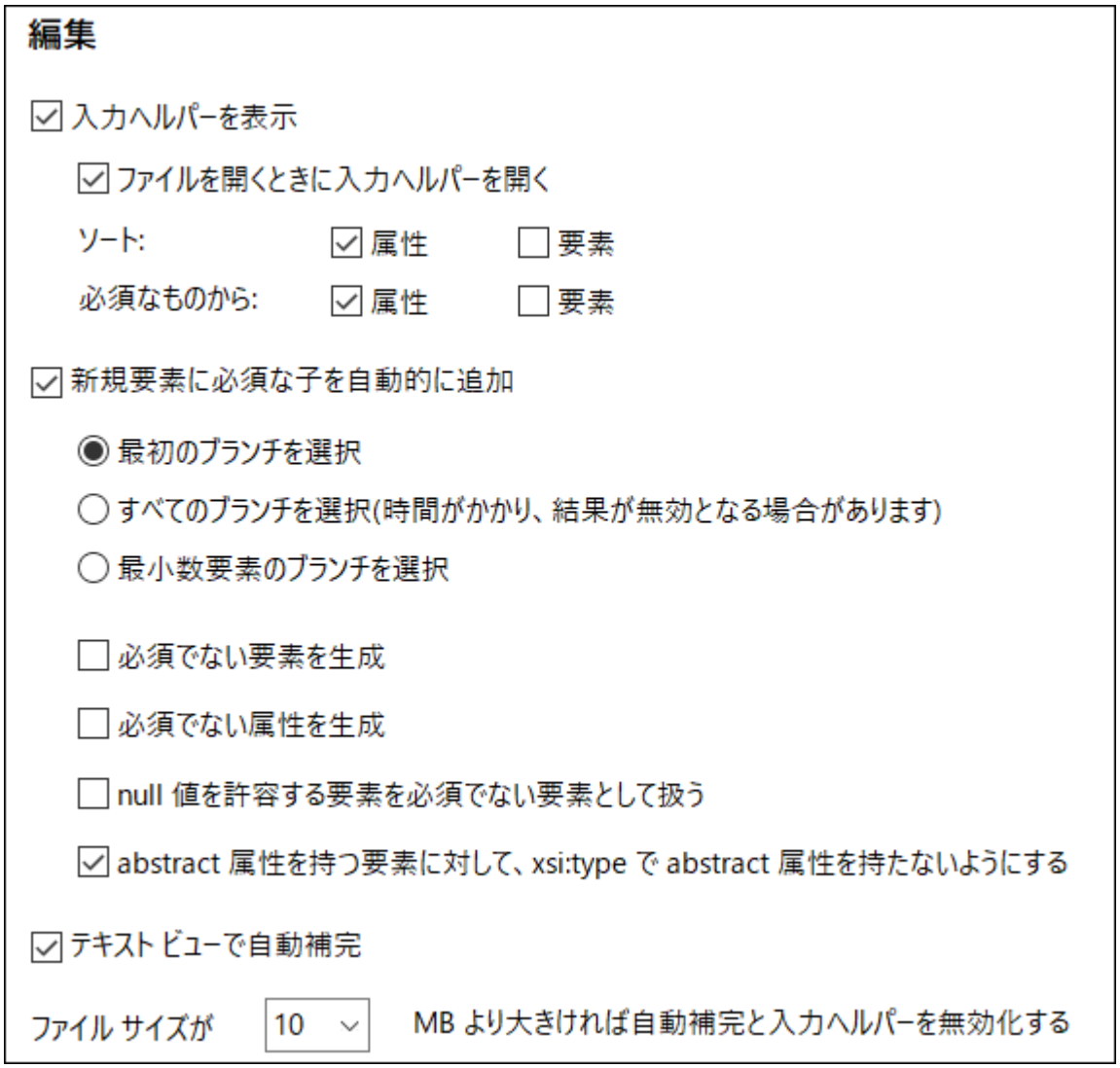
保存して完了する

「OK」をクリックして完了してください。

(C) 2015-2021 Altova GmbH

28.17.13.4 編集

編集セクションでは、XMLSpy の編集における振る舞いを指定することができます。



入力ヘルパー

ドキュメントの編集中 XMLSpy はこれらの設定をベースにしてインテリジェントな編集機能を提供します。このペイン内で入力ヘルパー内に表示されるアイテムの順序などを含む入力ヘルパーの振る舞いの多種のアスペクトをカスタム化することができます。編集中のファイル型に適切な場合にここで行われるカスタム化の設定は適用されます。例えば、ファイルを開く際に入力ヘルパーをロードし属性を置き換えるオプションは DTD または XQuery ドキュメントには適用することができません。

XML 構造を XML スキーマから作成する

XML スキーマをベースにする新規の XML ドキュメントを作成する場合、スキーマ内の定義から派生した構造を持つドキュメントが生成されます。下で説明されている設定はこの構造の作成に関連する抽象的なアスペクトのいくつかを決定します。

必須の子を自動的に付加する
スキーマ内の `choice` グループの必須の子要素はこのペイン内での設定をベースにして自動的に付加されます。(i) `choice` グループの最初のブランチ(要素) (ii) 全てのブランチ、または(iii) 子孫要素の最小数を持つブランチが生成されます。`choice` グループから1つのブランチのみが許可されているためNote that the *全てのブランチ* の選択は無効なドキュメントを生成する場合があることご注意してください。

抽象型の必須ではないノードと要素
必須ではない要素または属性を追加するには、対応するオプションを選択してください。これらのオプションが選択されていない場合、必須のノードのみが追加されます。また(i) nillable 要素の要素コンテンツを必須では内に設定し(ii) 抽象型の要素の `xsi:type` として必須ではない型を使用します。o

テキストビュー

*自動補完*オプションにより、より曖昧性の少ない構造のコンポーネントを作成することができます。例えば、要素の開始タグを閉じる山括弧が入力されると、その要素の終了タグも自動的に追加されます。

テキストビューでは *自動補完の無効化* コンボボックス内で指定されているよりファイルが大きい場合自動補完や入力ヘルパーを無効にすることができます。大きなファイルに対する編集のスピードを向上させる場合この機能は役に立ち、自動補完機能と入力ヘルパーを使用しないで行うことができます。このオプションのために指定されているサイズよりもファイルサイズが大きい場合、テキストビューコンテキストメニューには自動補完と入力ヘルパーの使用をオンとオフに切り替えるためのトグルコマンドが含まれます。(このオプションのために指定されているサイズよりも大きいファイルの場合)編集エイドを編集中にオンとオフにいつでも切り替えることができます。このオプションのために指定されている値が開かれているファイルのサイズよりも小さい場合、エラーメッセージ内で指定されているロケーションはテキストビュー内のロケーションに正確に対応しません。

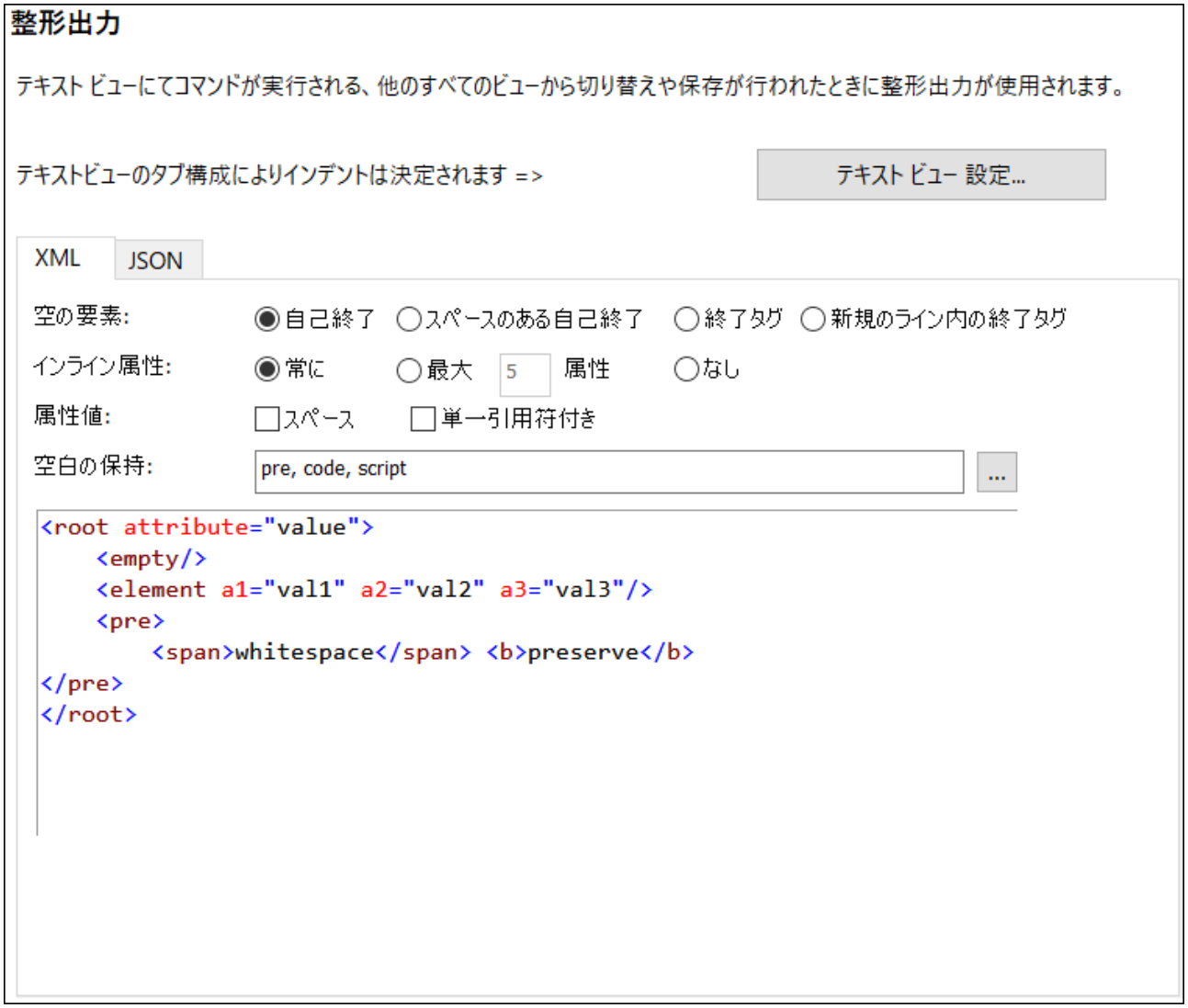
保存して完了

設定が完了したら、「OK」をクリックして完了します。

28.17.13.5 整形出力

整形出力 セクション *(下のスクリーンショット参照)* によりテキストが テキストビューこのセクションの定義は以下のカテゴリに整理されています:

- XML 設定(これらの設定にアクセスするためにXML タブを選択します。 *下の左側のスクリーンショット*)
- JSON 設定(これらの設定にアクセスするためにJSON タブを選択します。 *下の右側のスクリーンショット*)
- テキストビュー設定(設定ダイアログにアクセスするためにボタンをクリックします)


整形出力
テキスト ビューにてコマンドが実行される、他のすべてのビューから切り替えや保存が行われたときに整形出力が使用されます。
テキストビューのタブ構成によりインデントは決定されます =>
テキスト ビュー 設定...
XML
JSON
空の要素:
自己終了
スペースのある自己終了
終了タグ
新規のライン内の終了タグ
インライン属性:
常に
最大
5
属性
なし
属性値:
スペース
単一引用符付き
空白の保持:
pre, code, script
...
<root attribute="value">
<empty/>
<element a1="val1" a2="val2" a3="val3"/>
<pre>
whitespace preserve
</pre>
</root>

XML 設定

以下の設定を使用することができます:

- どのように要素が作成されドキュメント内で表示されるか
- インライン属性は親要素として同じラインまたは新規のラインに表示されます。
- (等号記号の両端に空白が存在するか、および値が単一または二重引用符で囲まれているかなど)属性値の表記方法
- 空白として保持する要素の設定
- テキストビュー設定 1363 ダイアログ内で指定されているインデントが(ペインの上のチェックボックスを介して指定され)使用されているかどうか。

オプションを選択すると、その効果は[**OK**]で確認する前に効果を確認することのできるダイアログの下のプレビューペインに表示されます。

JSON 設定

選択する各オプションのために、効果はすぐダイアログの下のプレビューペインに反映されます。

(C) 2015-2021 Altova GmbH

- *インライン配列:* 配列のアイテムを単一のライン（またはインライン）に表示します。それ以外の場合、アイテムは個別のラインに表示されます。配列にインライン書式設定を決して適用しない、またはインライン書式設定を空の配列または特定のサイズまでの配列に適用することを選択することができます。
- *インラインオブジェクト:* オブジェクトのプロパティを単一のライン（またはインライン）に表示します。それ以外の場合、プロパティは個別のラインに表示されます。インライン書式設定をオブジェクトに決して適用しない、または空のオブジェクト、または、プロパティの特定の数を持つオブジェクトにインライン書式設定を適用することを選択できます。
- *インラインパディング:* 選択されると空以外のインライン配列と空以外のインラインオブジェクト間に空白を追加します。このオプションは空以外のインライン配列または空以外のインラインオブジェクトオプションが選択されている場合有効化されます。
- *インラインパディングを空にする:* 選択されると空のインライン配列の区切り文字の中に空白を追加します。配列またはオブジェクトが（空または空ではない）インラインに設定されている場合このオプションは有効化されます。
- *単一引用符に囲まれた文字列（JSON5):* 選択されるとJSON5ドキュメント内のすべての引用符を一重引用符に変換します。
- *引用符で囲まれていない文字列（JSON5):* 選択されている場合JSON5ドキュメント内で（`key:value` ペアの）全てのキーの周りから引用符を削除します。

上のチェックボックスを介してテキストビュー設定[1363]ダイアログ内でインデントが使用されるかまたはインデントを使用しないかを指定することができます。

テキストビューの設定

テキストビューをクリックしてインデント、ブックマーク余白、自動ハイライトなどのテキストビューのプロパティを有効化できるテキストビュー設定ダイアログを開きます。テキストビュー設定ダイアログはメニューコマンド [表示 | テキストビュー設定][1363]を介してアクセスすることができます。ダイアログについて説明されています。

28.17.13.6 ビュー

ビューセクションでは、XMLSpy における XML ドキュメントの表示方法をカスタマイズすることができます。

プログラムロゴ

アプリケーションの起動時に表示されるスプラッシュ画面を無効にすることで、起動時間を短縮することができます。更に、ライセンスを購入されている場合、プログラムロゴ、著作権情報、登録に関する詳細の印刷を XMLSpy にて行わないように指定することができます。

ウィンドウタイトル

各ドキュメントウィンドウのためにウィンドウタイトルは、ファイル名、または、フルパス名になります。

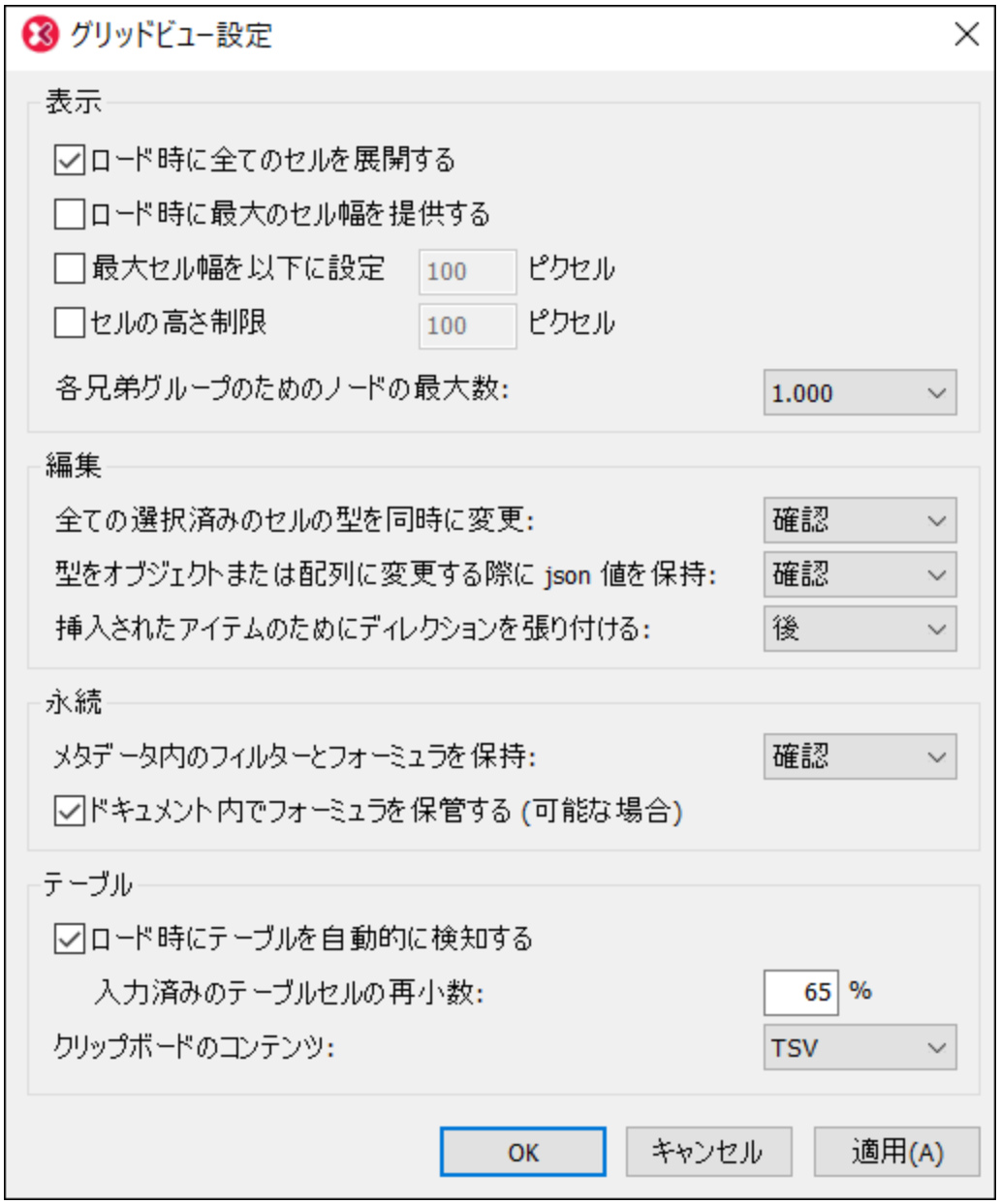
(C) 2015-2021 Altova GmbH

テキストビューの設定

テキストビュー をクリックしてインデント、ブックマーク余白、自動ハイライトなどのテキストビューのプロパティを有効化できるテキストビュー設定ダイアログを開きます。テキストビュー設定ダイアログはメニューコマンド [表示 | テキストビュー設定] 1363 を介してアクセスすることができます。ダイアログについて説明されています。

グリッドビュー設定

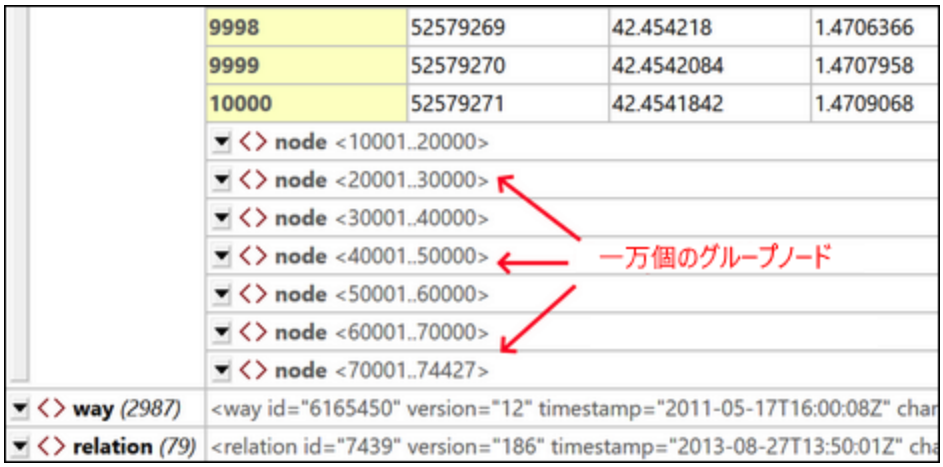
グリッドビュー設定 をクリックしてグリッドビュー設定ダイアログを開きます。グリッドビューのために以下のオプションをセットすることができます:



表示

表示 セクション内のチェックボックスは自明です。以下は更に説明が必要な場合のメモです。

- ロードする際に全てのセルが展開されていない場合ルートノードとそのすべての子孫は折りたたまれています。ドキュメントを更に深く移動するためには各ノードを展開する必要があります。
- 最適な幅がオンに切り替えられている場合グリッドの幅全体が表示されます。これを達成するには、セルの一部のテキストがラップされます。
- 兄弟ノードは100、1000 または10000 ノードの兄弟グループで表示することができます（*下のスクリーンショットを参照してください*）。多数の兄弟ノードを持つファイル内でスペースの表示を省略する場合



編集

編集 セクション内のチェックボックスは自明です。以下は更に説明が必要な場合のメモです。

- 複数の選択されたセルの型を変更する場合、設定のアクションを継続するかを決定する以下のオプションが与えられます: (ユーザーにより決定される) *常に*、*なし* または *確認* 。
- JSON 型を動的な型からオブジェクトまたは列挙に変更する場合、設定のアクションを継続するかを決定するかを決定する以下のオプションが与えられます: (i) *確認* (動的な型の値が名前を持たない子 `key:value` ペアの値として保持される、または破棄されるかを確認) (ii) *常に* (名前を持たない子 `key:value` ペア内の値を保持する) (iii) *なし* (値を保持する。すなわち値を破棄する)。
- *ディレクションの貼り付け* オプションはクリップボード内の選択範囲が選択されたセルの上または下に張り付けられるかを決定します。

永続

フィルターとフォーミュラを XML と JSON ドキュメントのメタデータ内に保管することができます。フォーミュラは XML ドキュメント内の処理命令として、そして JSON ドキュメント内のコメントとして保管されます。これらの設定はフィルターとフォーミュラの保管の方法を指定します。フォーミュラの場合 *可能な場合* という用語はフォーミュラが実際にすべての JSON ドキュメント (JSON5 と JSONC ドキュメント内でのみ) のコメントとして保存できないことを意味します 。

テーブル

以下の設定を使用することができます:

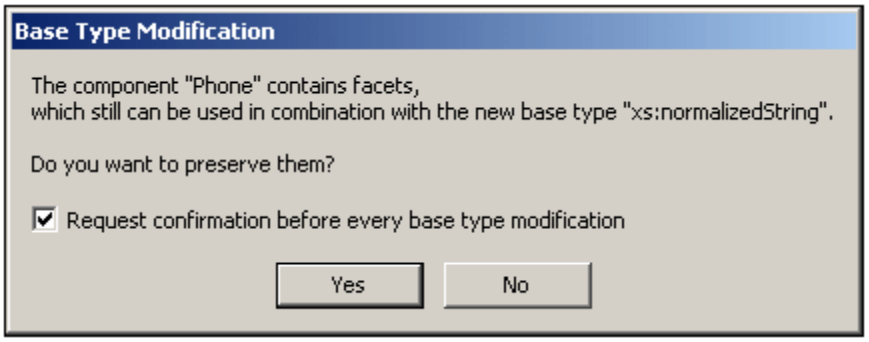
- 設定がロード時にテーブルを自動的に検知する場合、テーブルをテーブルとして検知するために評価する記入されたテーブルセルの最小のパーセントを選択することができます。記入されたテーブルセルの数値がこのレベルを越さない場合、構造はそれぞれが下に表示されている繰り返し要素を持つ通常のグリッドとして表示されます。
- クリップボードコンテンツがタブにより区切られた値 (TSV) または (ドキュメントの型に従い) XML/JSON として保管するかを選択することが可能です。これはとても役に立つ機能です: クリップボードからテーブルを他のドキュメントに張り付ける場合、この設定によりTSV としてまたはマークアップとしてテーブルがコピーされるかを選択できるようになります。(違いを確認するには、テーブルをそれぞ

れのフォーマットでクリップボードにコピーしテキストエディターにテーブルを張り付けてください。）

スキーマビュー

XML スキーマデータ型は他のデータ型から派生することもあります。例えば `xs:string` を基底型として、電子メールに使用される要素のデータ型を派生させることができます（`xs:string` データ型を特定の文字セットに制限します）。その後基底型が変更された場合、以下にあるオプションのいずれかを選択することができます:

- *コンテンツの保持*: 新たな基底型でも既存の定義を使用することができる場合、このオプションを有効にすることで、定義が自動的に保持されます。.
- *変更を確認する*: 基底型を変更すると、（*下のスクリーンショットに示される*）ダイアログが表示され、それまでの定義を保持したまま新たな基底型を使用し続けるか尋ねられます。



Authentic View

このオプションが有効になっている場合、**StyleVision Power Stylesheet** をベースにしたXML ファイルがAuthentic View にて自動的に開かれます。

ブラウザービュー

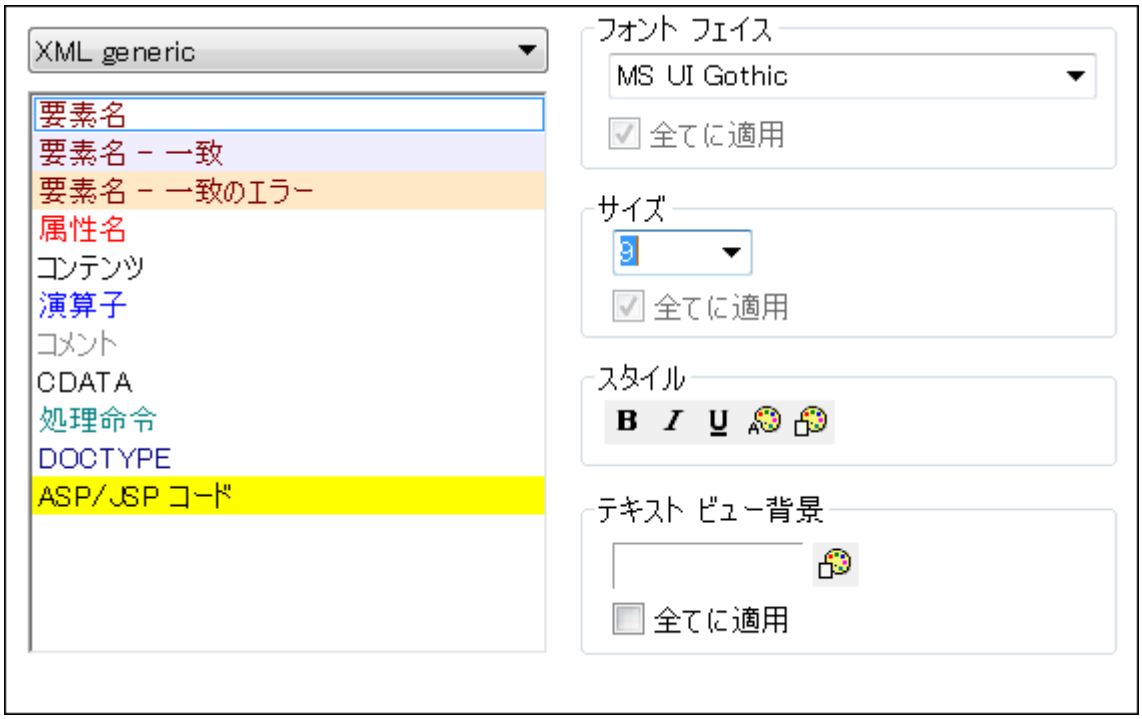
ブラウザービューを独立したウインドウとして表示して、編集を行なっているビューウインドウに隣合わせで表示することができます。

保存して完了

設定が完了したら、「OK」をクリックして完了します。

28.17.13.7 フォントと色

フォントと色 セクションでは、XMLSpy の異なるビュー内のテキストアイテムの外観のためのカスタム化オプションが与えられています。



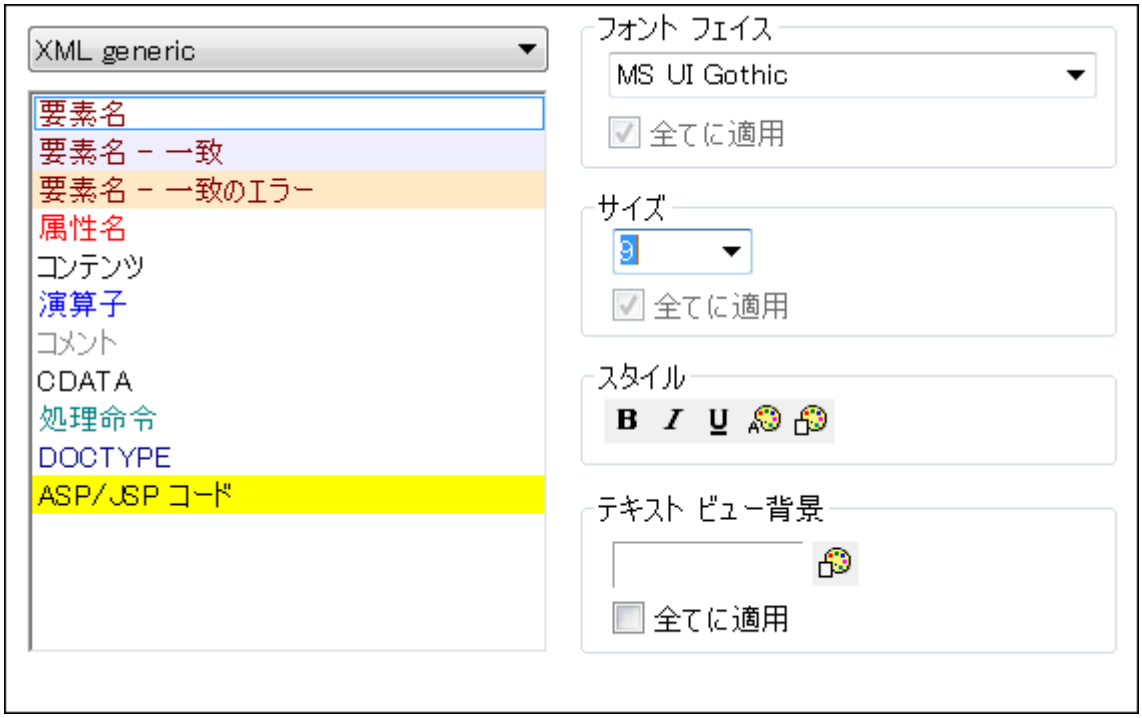
次のビューのためのオプションを使用することができます:

- テキストビュー[1470]
- グリッドビュー[1472]
- スキーマ設計ビュー[1475]
- WSDL デザインビュー[1477]
- XBRL タクソノミビュー[1478]

ダイアログの左側のペイン内で、カスタマイズするビューを選択します。書式設定することのできるテキストアイテム型は、右側のペイン内に表示されます(*上のスクリーンショット*)。書式設定するテキストアイテム型を選択し、希望する書式設定のプロパティの値を割り当てます。

28.17.13.7.1 テキストビュー

例えば、要素名と属性名を別々の色で表示することができます。(*下のスクリーンショット および ドキュメントの型内の ドキュメントのリスト セクション内の ドキュメント を参照してください*)。例えば、異なる書式設定をXMLドキュメント内の要素名と属性名の表示のために、または、XQueryドキュメント内のキーワードと変数を表示するために指定することができます。



ドキュメントの型

次のドキュメントのテキストビューアイテムの書式設定を行うことができます:

- XML ジェネリック
- XQuery
- CSS
- JSON
- C-family
- Python
- XULE
- その他
- 出力

カスタム化の方法

以下の操作により、テキストフォントのカスタマイズを行うことができます:

1. 左上にあるコンボボックスにてカスタマイズしたいテキストフォントを選択します。この際、選択された種類のドキュメントで表示されるテキストアイテムがコンボボックス以下のボックスに表示されます(上のスクリーンショットでは、`XML generic` がドキュメントの種類として選択されています)。
2. カスタマイズを行うテキストアイテムクリックにより選択します。(*上のスクリーンショットでは* 要素名 *が選択されています。*)
3. 右側のペイン内のオプションを使用し、選択されたテキストアイテム型のフォントプロパティを設定します。*テキストビュー背景* オプションにより、テキストビュー全体のための背景色の選択が有効化されます。

以下の点に注意してください:

- (XML ジェネリックドキュメントの型などの) 特定のドキュメントの型のすべてのテキストアイテム型のための同じフォントフェイス、サイズ、スタイルです。ドキュメントの型内で個別のテキスト型の、テキストの色と背景色のみを変更することができます。これにより構文色分け機能が有効化されます。
- *要素名* テキスト型は3つのサブ型により構成されています: (i) *要素名* は、選択されていない要素名に適用されます。(ii) *要素名 – 一致* は開始タグ名が終了タグ名が一致する箇所以外で選択されている要素名(カーソルが置かれている名前)に適用されます。(iii) *要素名 – 一致 エラー* は、選択された要素名に適用することができます。編集される要素名は、このために開始タグ名が終了タグ名 に一致するかに従い、異なる背景色にハイライトされます。景色に対応する背景色を変更することによりこれらのハイライトの色を変更することができます。デフォルトで、ハイライトはオンに切り替えられ、テキストビュー設定[1363] ダイアログ内の*ハイライト 要素* オプションの選択を解除することによりオフに切り替えることができます。
- XULE カテゴリ内で適用される設定: (I) メインウィンドウ内のXULE ドキュメント および(ii) XULE ウィンドウ[824] 内に入力されるXULE rルール。
- その他のカテゴリ: (i) アクティブな選択は、現在選択されているテキストのコンテンツを指します。アクティブでは内選択は、同じテキストコンテンツのドキュメント内の他の箇所での発生を指します。(ii) アクティブなマーカーの検索は、現在選択されている検索の結果の発生を*指します*。マーカーの検索は、他の(アクティブではない) 検索結果の発生を指します。(iii) *デバッグ呼び出し マーカー*は、デバッグセッション内の現在選択されているステップを指します。(iv) *レ表示されている空白* は、ドキュメント内の空白を指します。ドキュメント内の空白は空白マーカーの切り替え[1363]により表示を切り替えることができます。(v) (iii) アクティブなバックマッピングは、バックマッピングのドキュメント[1285] (`XML-XSLT/XQuery-Result`)内で現在選択されているコンテンツを指します。*アクティブではない* バックマッピングは、他の2つのドキュメント内のバックマップされたコンテンツを指します。

デフォルトに設定

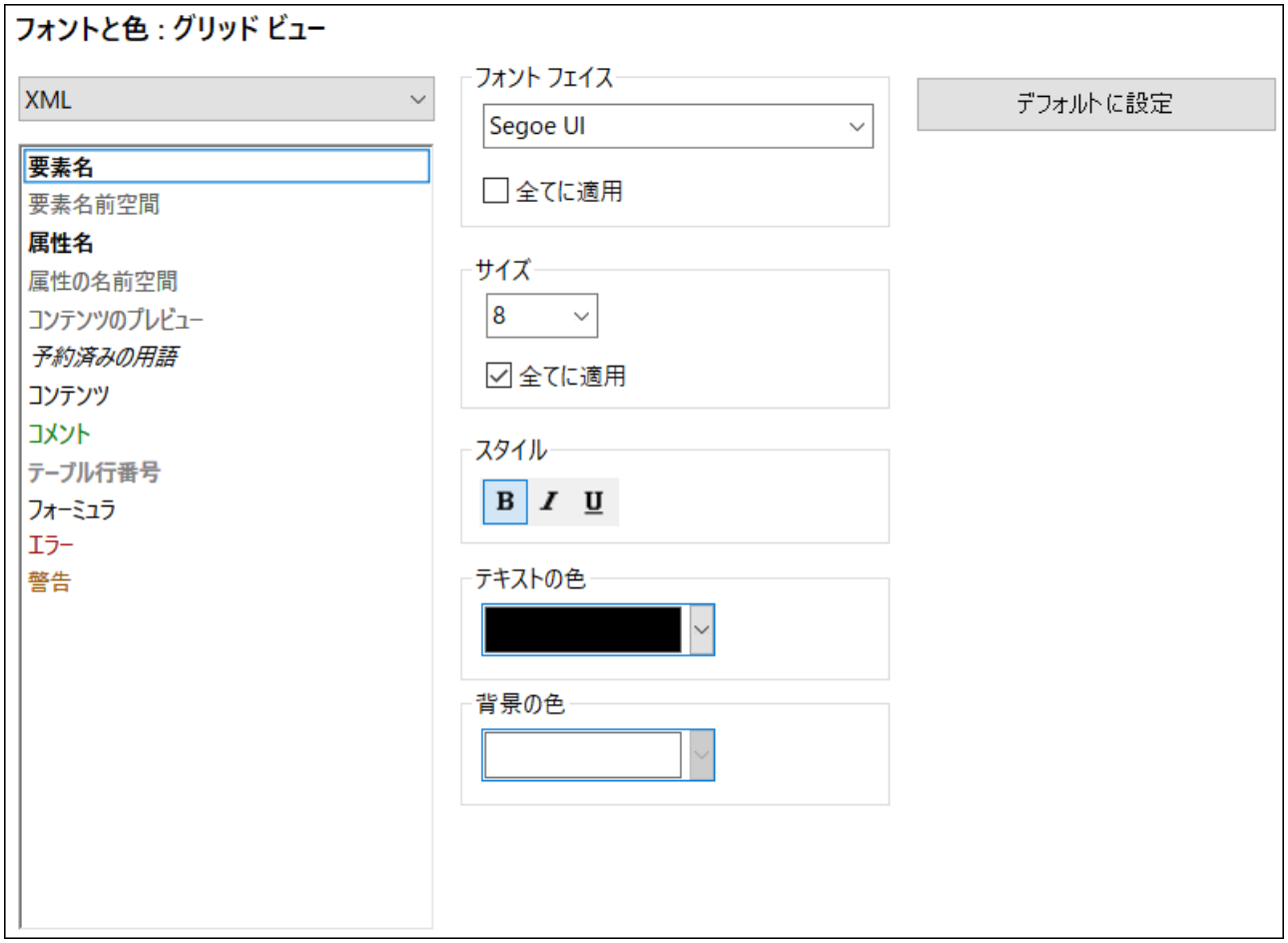
デフォルトに設定ボタンによりオリジナルのインストール設定にフォントがリセットされます。

保存して完了する

設定が完了したら、「OK」をクリックして完了します。

28.17.13.7.2 グリッドビュー

グリッドビューのセクション (*下のスクリーンショット*) により XML グリッドビュー[149]、JSON グリッドビュー[600] および Avro グリッドビュー[681] 内のテキストの外観をカスタム化することができます。グリッドコンポーネントがテーブルで表示されている場合 *セルの色* オプション (*スクリーンショット*) によりセルの色を設定することができます。コンボボックス内でグリッドビューを構成するためのドキュメントの表示を選択します。書式設定するテキストの型を選択し、(*下にリストされる*) 希望するフォーマットプロパティに割り当てます。



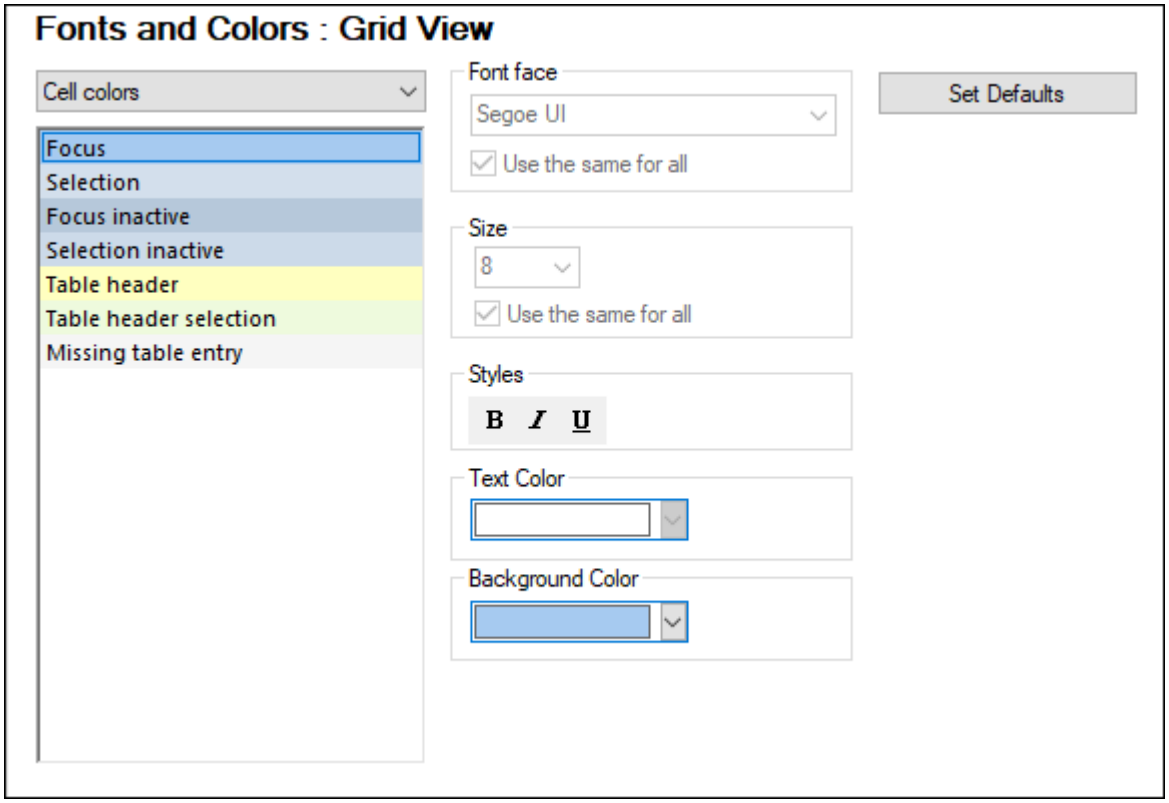
XML グリッド、JSON グリッド、Avro グリッド

選択されたテキスト アイテム型のために次のプロパティを設定することができます:

- *フォント フェイスとサイズ*: 選択されたフォントはグリッド ビューの印刷内でも使用することもできます。全てのテキスト アイテム型のために同じフォント フェイスまたはサイズを使用する場合、対応する、*「全てに対して同じ設定を使用する」* チェックボックスをチェックします。
- *テキスト スタイル、テキストの色、テキストの背景色*: 個々のテキスト アイテム型のスタイル、色、及び背景を設定します。現在の設定は左側のペインのリスト内で直ぐに反映され、テキスト アイテムをプレビューすることができます。

(全てのグリッド内のテーブルの表示の) セルの色

全てのグリッド (XML、JSON、Avro) 内のテーブル表示を言語に関連するセマンティックスではなく設定することが可能です (例えば、XML 要素を XML コメントとは異なるように書式設定することが可能になります)。セル関数をベースにグリッド ビューをフォーマットすることもできます: 例えば、テーブル コンポーネントの選択された/選択を解除された状態の区別することなどができます。このような区別は異なる背景を各アイテムに割り当てることで達成することができますが (*下のスクリーンショット参照*) セルの異なる型のために他の /追加フォーマット プロパティを使用することもできます。



以下のコンポーネントのための設定を使用することができます:

- *(選択されていない)テーブルヘッダー* と *テーブルヘッダーの選択*: これらは列と行ヘッダーを参照します。下のスクリーンショットは選択を解除されたヘッダーを表示しています。背景色は上のダイアログで設定されている通りです。個別のヘッダーが選択されておらず、全てのヘッダーが選択されていると*選択された*ヘッダーの色が有効化されます。列と行が交わるセルをクリックすることですべてのヘッダーを選択、または、テーブルまたはその祖先のいずれかを選択することにより全てのヘッダーを選択することができます。

Name	Administration		
Person (3)			
	First	Last	Title
1	Vernon	Callaby	
2	Frank	Further	Accounts Receivable
3	Loby	Matise	Accounting Manager

- *不足しているテーブルエントリ*: ドキュメント内の既存ではない要素または属性を参照します (*下のスクリーンショット参照*)。
- *選択* と *フォーカス*: 選択されたセルとフォーカスを持つセルを参照します。例えば、下のスクリーンショットでは `Person` テーブルの全体が選択されており、右下のセルにフォーカスが存在します。

Name	Administration		
Person (3)			
	First	Last	Title
1	Vernon	Callaby	
2	Frank	Further	Accounts Receivable
3	Loby	Matise	Accounting Manager

- *非アクティブな選択、非アクティブなフォーカス*: セルのブロックが選択されており、そのブロック内の1つまたは複数のセルがフォーカスに設定されている場合、グリッドビューがアクティブな間は両方の選択範囲とフォーカスがアクティブです。しかしながら、グリッドビュー内の選択範囲またはフォーカスが削除されず他のダイアログまたはウィンドウがアクティブに設定された場合、グリッドビュー内の選択範囲とフォーカスは非アクティブになります。グリッドビューがアクティブに戻されると、選択範囲とフォーカスもまたアクティブに戻ります。

メモ ここで定義された色以外にも、XMLSpy ではWindows の設定にて選択されたメニューの色も反映されます。

デフォルトに設定

デフォルトに設定ボタンによりオリジナルのインストール設定にフォントがリセットされます。

保存して完了する

設定が完了したら、「OK」をクリックして完了します。

28.17.13.7.3 スキーマ設計ビュー

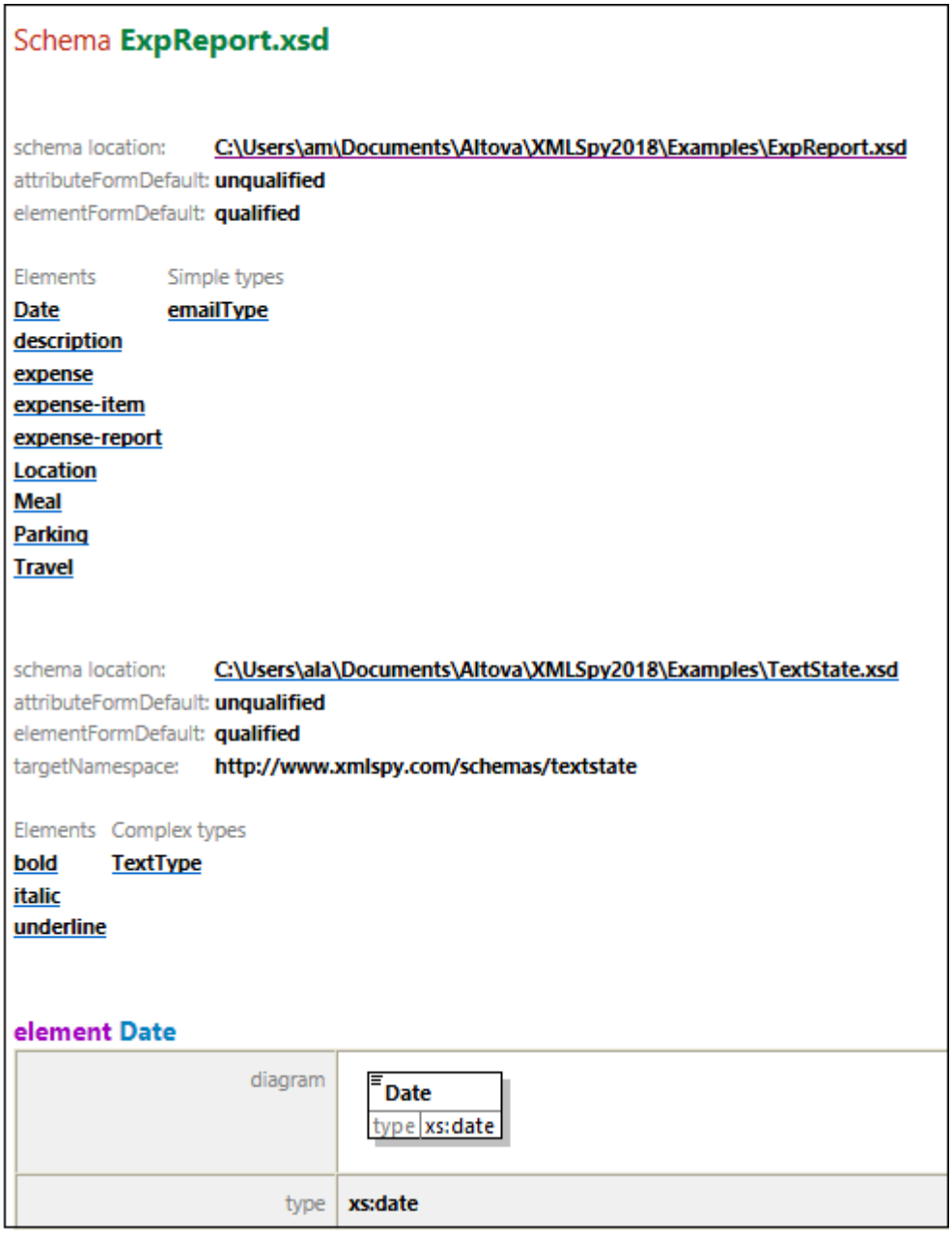
スキーマ設計ビュー セクションで[XML スキーマ](386) および[JSON スキーマ](592) ドキュメント[スキーマビュー](159) の表示の外観をカスタム化することができます。

カスタム化の方法

選択されたドキュメントの型の個別のテキストアイテム型をカスタム化するには、次を行います:

1. 左上ののコンボボックスから、XML スキーマ または JSON スキーマを選択します。選択されたドキュメントのテキストアイテム型がコンボボックスの下に表示されます。
2. 書式設定するテキストアイテム型をクリックして選択します。
3. 選択されたテキストアイテム型のプロパティを、右側のペイン内のオプションを使用して設定します。

メモ
ドキュメントスキーマヘッダー(2) 及び ドキュメント要素ヘッダー(2) テキストアイテム型 は、それぞれ、スキーマの生成されたドキュメント 1259 内のスキーマヘッダー、要素ヘッダーを指します。下のスクリーンショット内の生成されたドキュメントのスキーマヘッダーと要素ヘッダーの色を上のダイアログ内のプロパティの色と比較してください。



デフォルトに設定

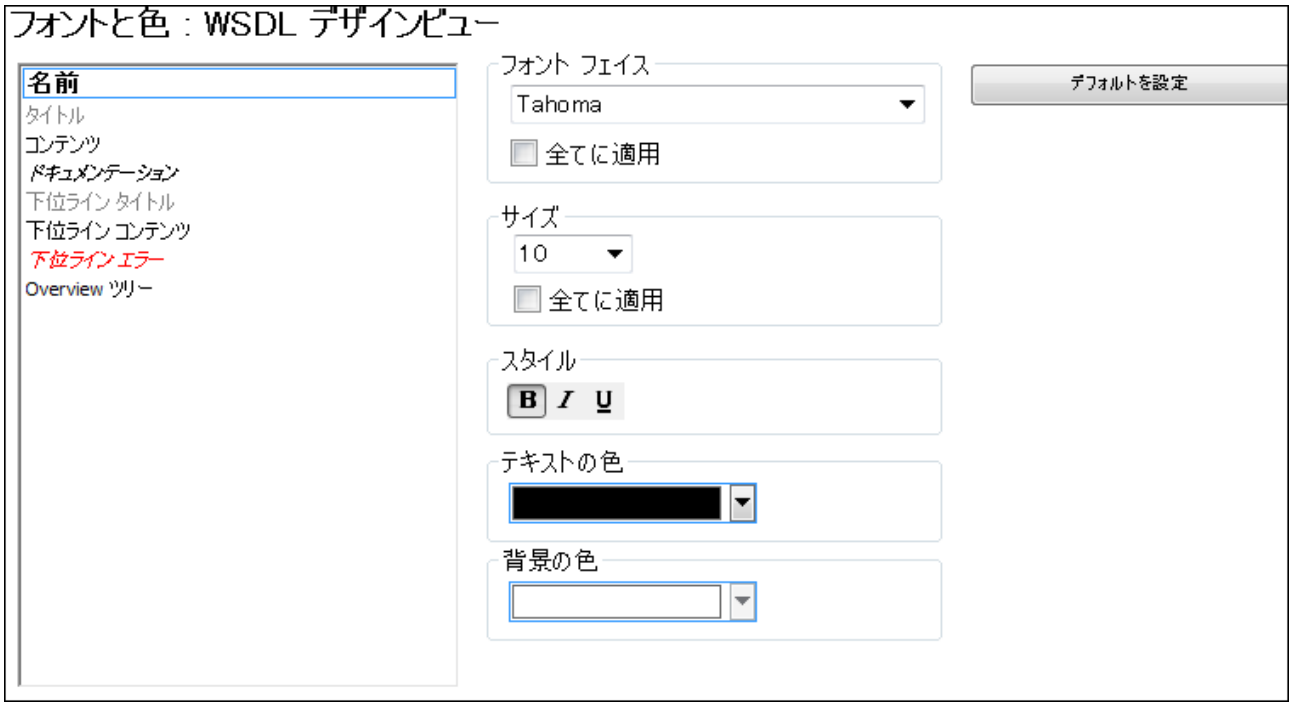
デフォルトに設定ボタンによりオリジナルのインストール設定にフォントがリセットされます。

保存して完了する

設定が完了したら、「OK」をクリックして完了します。

28.17.13.7.4 WSDL デザインビュー

WSDL デザインビュー セクションでは、WSDL ビュー[231] 内のテキストアイテムの外観のためのカスタム化オプションが与えられています (*下のスクリーンショットを参照してください*)。



選択されたテキストアイテム型のために次のプロパティを設定することができます:

- *フォントフェイスとサイズ*: 選択されたフォントはWSDL デザインビューの印刷内でも使用することもできます。全てのテキストアイテム型のために同じフォントフェイスまたはサイズを使用する場合、対応する、「全てに対して同じ設定を使用する」チェックボックスをチェックします。
- *テキストスタイル、テキストの色、テキストの背景色*: 個々のテキストアイテム型のスタイル、色、及び背景を設定します。現在の設定は左側のペインのリスト内で直ぐに反映され、テキストアイテムの外観をプレビューすることができます。

メモ

WSDL ドキュメントの生成されたドキュメント[1375] 内のWSDL ヘッダーとサービス名の書式設定はそれぞれ以下の設定から選択されます: ドキュメントスキーマヘッダー(2)、ドキュメント要素ヘッダー(2)、グリッドビュー[1472] のテキストアイテム型

デフォルトに設定

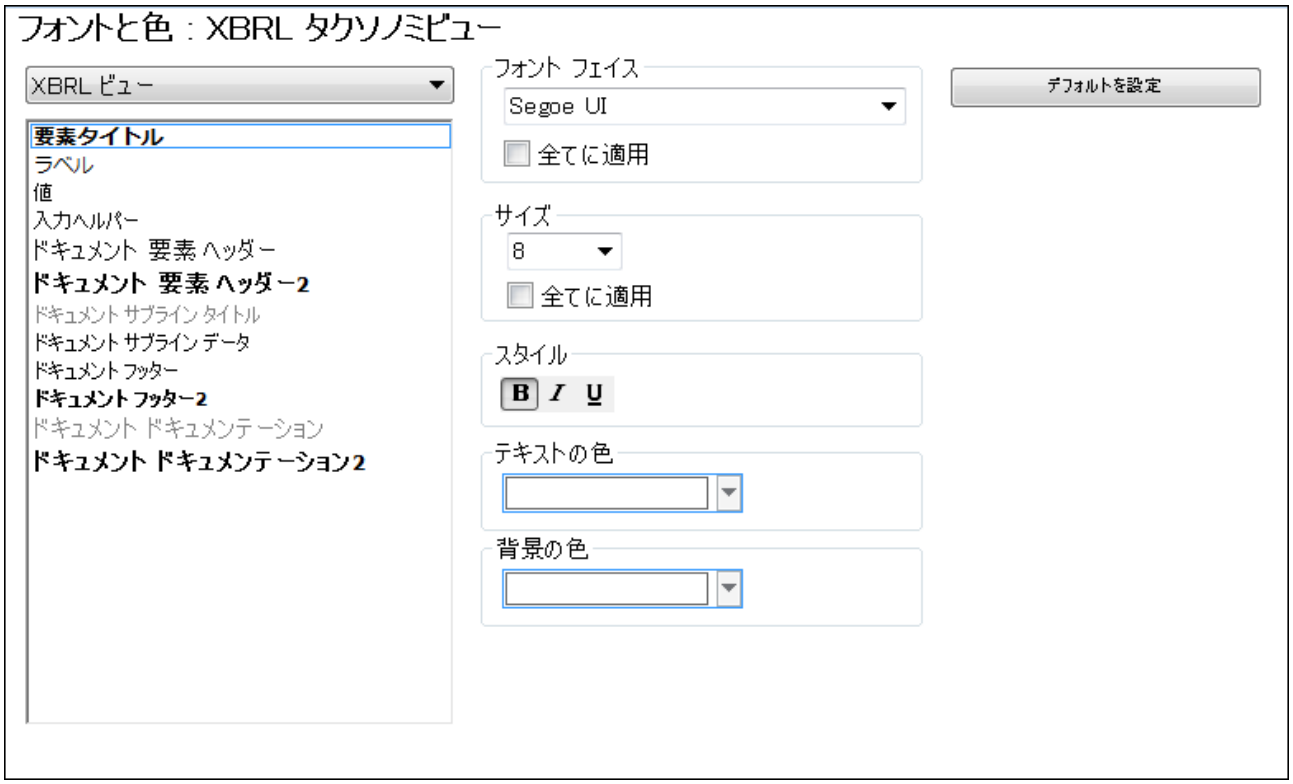
デフォルトに設定ボタンによりオリジナルのインストール設定にフォントがリセットされます。

保存して完了する

設定が完了したら、「OK」をクリックして完了します。

28.17.13.7.5 XBRL タクソノミビュー

XBRL タクソノミビュー セクションでは、XBRL ビュー[243] 内のテキスト アイテムの外観のためのカスタム化オプションが与えられています（*下のスクリーンショットを参照してください*）。



選択されたテキスト アイテム型のために次のプロパティを設定することができます:

- *フォント フェイスとサイズ*: 選択されたフォントは XBRL タクソノミビューの印刷内でも使用することもできます。全てのテキスト アイテム型のために同じフォント フェイスおよびサイズを使用する場合、対応する、*「全てに対して同じ設定を使用する」* チェックボックスをチェックします。
- *テキスト スタイル、テキストの色、テキストの背景色*: 個々のテキスト アイテム型のスタイル、色、及び背景を設定します。現在の設定は左側のペインのリスト内で直ぐに反映され、テキスト アイテムをプレビューすることができます。
- XBRL テーブル プレビュー オプションにより テーブル XBRL ビューのタブ内でテーブル ビューすることのできる XBRL テーブルのコンポーネントのための設定を有効化することができます。

デフォルトに設定

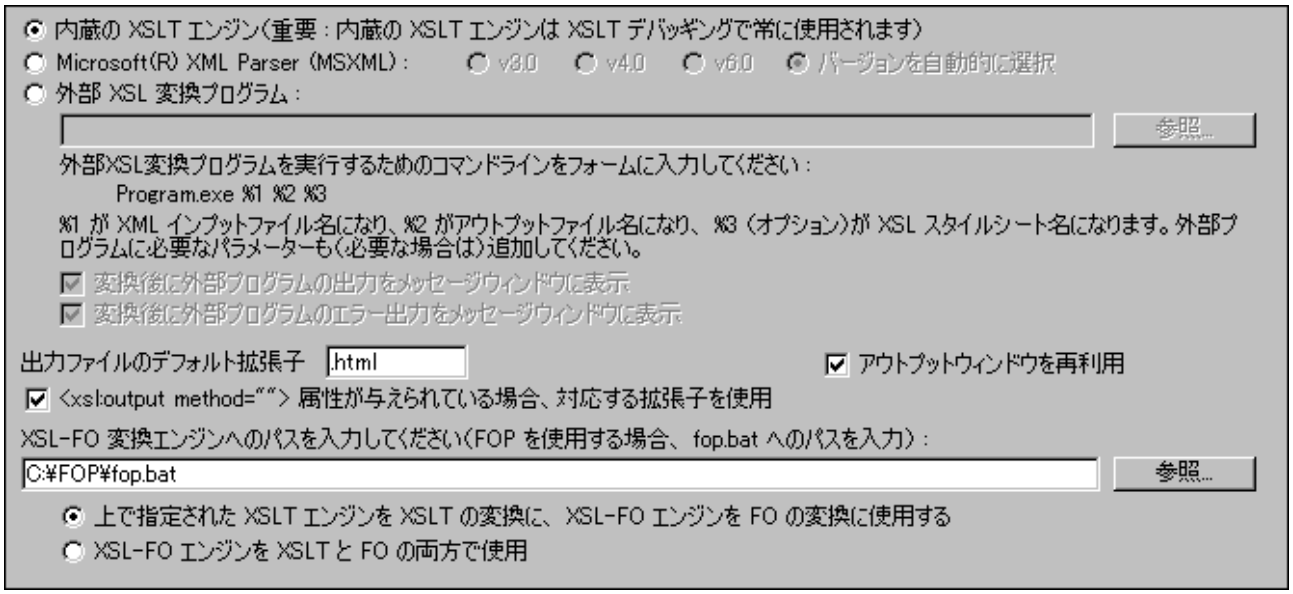
デフォルトに設定ボタンによりオリジナルのインストール設定にフォントがリセットされます。

保存して完了する

設定が完了したら、「OK」をクリックして完了します。

28.17.13.8 XSL

XSL セクションでは、アプリケーション内から行われるXSLT 変換[1479] とXSL-FO 変換[1480] におけるオプションを定義することができます



XSLT 変換

XMLSpy にはAltova XSLT 1.0 エンジンならびにAltova XSLT 2.0 エンジンが搭載されており、XSLT 変換に使用することができます。xsl:stylesheet またはxsl:transform 要素のversion 属性の値により、適切なXSLT エンジン(1.0 または2.0)が変換には使用されます。これはXSLT 変換だけではなく、XMLSpy を使ったXSLT/XQuery デバッガーのデバッグでも適用されます。

XSLT を使ったXML ドキュメントの変換には、以下のどれかを使用することができます:

- 内蔵の Altova XSLT エンジン(Altova (XSLT 1.0、XSLT 2.0、およびXSLT 3.0)。
- MSXML 3.0、4.0、または 6.0 パーサー：お使いのコンピューターで使用されている MSXML のバージョンが明らかな場合、選択を行うことができ、そうでない場合、アプリケーションによるバージョンの選択を行わせるべきでしょう(バージョンを自動的に選択オプションがデフォルトでは選択されています)。このオプションが選択されている場合、利用できる最新バージョンが選択されます。
- 選択した外部 XSLT プロセッサー：外部 XSLT プロセッサーを指定するコマンドライン文字列を入力する必要があります。コマンドライン文字列を構築する際に、以下の変数を使用することができます：

  %1 = 処理する XML ドキュメント
  %2 = 生成する出力ファイル

%3 = 使用する XSLT スタイルシート（XML ドキュメント内に、スタイルシートへの参照が含まれない場合）

例えば以下のコマンドにより、Saxon (XSLT 1.0) を使った単純な変換を行うことができます：

```
myxsltengine.exe -o output.xml input.xml stylesheet.xslt parameter-name=parameter-value
```

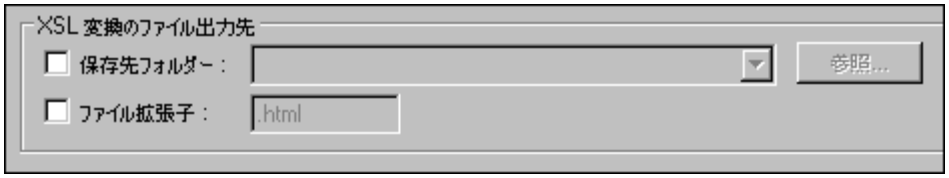
XMLSpy からこのコマンドを実行するには、外部 XSL 変換プログラムラジオボタンを選択し、以下の文字列をテキストボックスに入力してください。

```
c:\myxsltengine\myxsltengine.exe -o %2 %1 %3 parameter-name=parameter-value
```

適切なチェックボックスにチェックを入れることで、外部プログラムから得られた出力やエラーメッセージを XMLSpy のメッセージウィンドウに表示することができます。.

メモ: XMLSpy のXSLT 入力パラメーターダイアログ[1281] にてセットされたパラメーターは、内蔵の Altova XSLT エンジンに対してのみ渡されます。デフォルトとしてセットされたその他の XSLT エンジンに値は渡されません。

アウトプットウィンドウを再利用オプションを使用することで、後に続く変換の結果を、同じ出力ウィンドウにて表示することができます。XML ファイルがプロジェクトに格納されており、アウトプットウィンドウを再利用オプションが無効になっている場合、この設定はプロジェクトプロパティ[1194] の出力ファイルパスにおける保存先フォルダーも無効になっている場合のみ適用されます。



XSL-FO 変換

FO ドキュメントは FO プロセッサーにより処理され、FO プロセッサーの実行可能ファイルへのパスは、XSL-FO 変換エンジンのテキストボックスにて指定する必要があります。変換処理は、「 XSL/XQuery | XSL-FO 変換[1280] 」メニューコマンドから行うことができます。ソースファイル(IDE 内部からコマンドが指定された場合は、アクティブなドキュメント）が XSL-FO ドキュメントの場合、変換に FO プロセッサーが呼び出されます。ソースドキュメントが XML ドキュメントの場合、まずは XSLT 変換を使って XML ドキュメントを XSL-FO ドキュメントへ変換する必要があります。この XSLT 変換では、アプリケーションのデフォルトエンジンとして指定した XSLT エンジンのほか、デフォルト FO プロセッサーとして指定した FO プロセッサー内部に組み込まれた XSLT エンジンを使用することもできます(*上を参照*[1479])。これらオプションを選択するには、適切なラジオボタンをクリックしてください。

設定が完了したら、「OK」をクリックして、オプションダイアログを閉じてください。

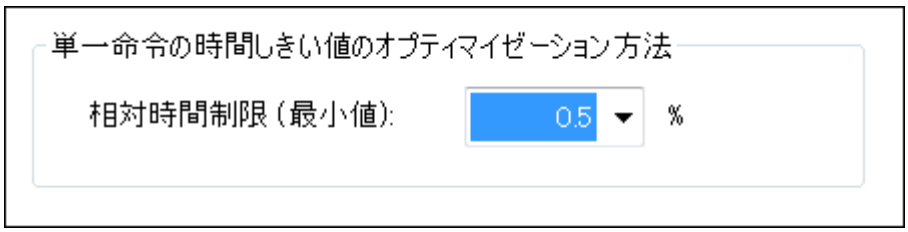
メモ: Apache XML プロジェクト のFOP プロセッサをインストールするオプションを解除しない限り、次のフォルダー内にインストールされます C:\ProgramData\Altova\SharedBetweenVersions。インストールされると、XSL-FO エンジン 入力ボックス内にパスが自動的に入力されます。FO プロセッサへのパスを希望するように設定することができます。しかしながら、FO プロセッサを使用する他の Altova 製品により同じパスにより使用される場合、FO プロセッサを選択して設定をすることもできます（StyleVision とAuthentic Desktop）。

保存して完了する

設定が完了したら、「OK」をクリックして、オプションダイアログを閉じてください。

28.17.13.8.1 スピードオプティマイザー

XSL スピードオプティマイザータブ（下のスクリーンショット）により XSL スピードオプティマイザー[440]のためにオプションを定義することができます。



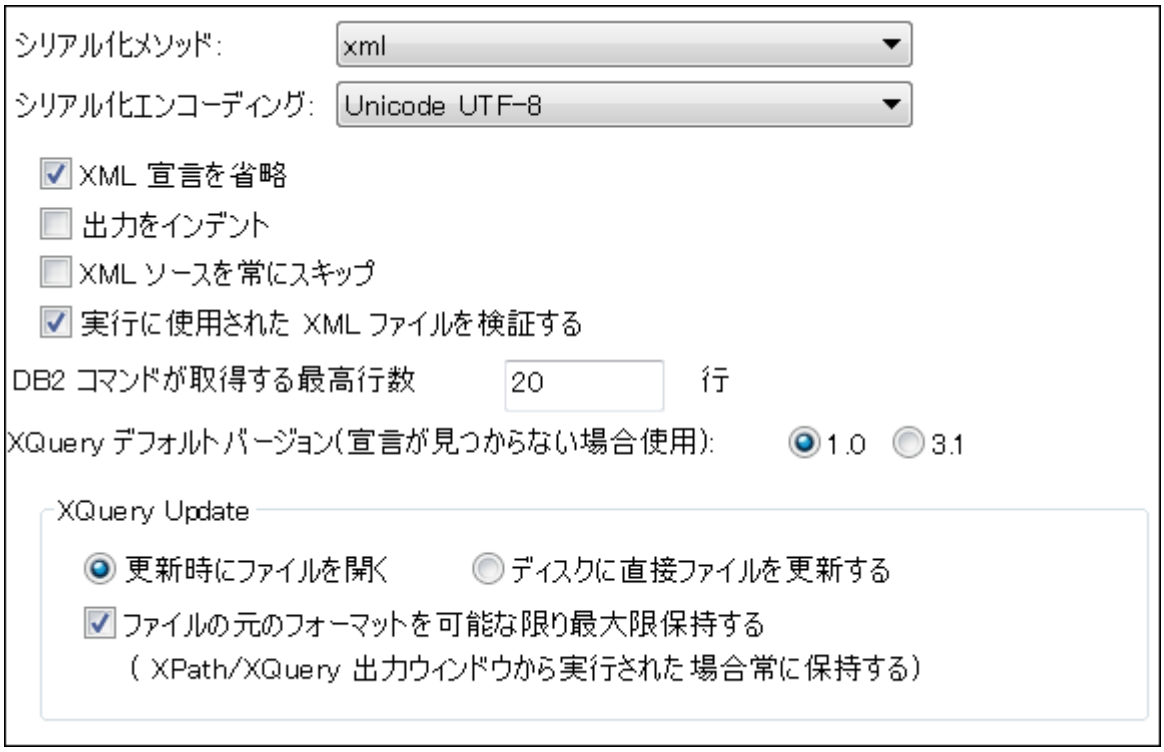
XSLT スタイルシート内の単一の XSLT 命令ための時間のしきい値をオプティマイザーのために指定することができます。値の範囲は、変換時間全体の99% から変換時間全体の0.1% 間です。命令がしきい値として指定されている時間よりも実行に時間がかかる場合、最適化の分析が呼び出されます。それ以外の場合、分析は行われません。最適化の分析に失敗した場合、時間のしきい値が高すぎる場合があります。値を低くすることを考えてください。

保存して完了する

設定が完了したら、「OK」をクリックして完了します。

28.17.13.9 XQuery

XQuery タブは、XQuery とXQuery Update ドキュメントの編集と実行に関連するオプションを定義します（*下のスクリーンショット*）。



一般的な XQuery オプション

以下のオプションから選択することができます:

- *シリアル化:* シリアル化は、XQuery ドキュメントが実行された時に生成される出力ドキュメントを指します。シリアル化 *メソッド*（アダプティブ*、HTML、JSON、テキスト、XHTML、*または*XML）と*シリアル化エンコード*、および、入力の異なる型のシリアル化の暗号化を選択することができます。出力のシリアル化は、XML 入力とJSON 入力のために個別に選択することができます（*メモ* アダプティブなメソッドにより、エラー無しにインスタンスドキュメントを処理することができます。入力ドキュメントをベースにしてシリアル化メソッドを自動的に決定します）。
- *XML 宣言の証言:* シリアル化された（出力）ドキュメント内のXML 宣言を省略します。
- *インデント 出力:* 出力ドキュメントをインデントして、ドキュメント階層構造を表示します。
- *XML ソースを常にスキップする:* XQuery ドキュメントが実行されると、XMLSpy は、XQuery ドキュメントを実行するためのXML *ソース*を問うことができます。プロンプトダイアログによりXML ファイルを参照することができます。このオプションをスキップして、XQuery ドキュメントを直接実行することもできます。このオプションが選択されると、XQuery ドキュメントは、XML ドキュメントにパスされることなく正確に実行されます。これはXML ドキュメントが必要とされないため、またはXML データがXQuery ドキュメント内の関数によりアクセスされるためです。
- *XML ファイルの検証:*XQuery ドキュメントの実行内で使用されるXML ファイルを検証します。無効なXML ファイルがフラッグされ、XQuery ドキュメントが処理されません。
- *DB2 行の取得 :* DB データを表示で、取得される行の最大数を指定することができます。XMLSpy は、`.xqr` ファイル拡張子[1458] をXQuery-for-DB ファイルとして認識します。

- *XQuery デフォルトのバージョン:* XQuery ドキュメントの実行のために使用する `version` キーワードを持たない XQuery エンジンバージョンを指定します。これは XQuery と XQuery Update ドキュメントの両方に適用され、使用するデフォルトの XQUery エンジンを選択します。

XQuery Update オプション

次の XQuery Update オプションを使用することができます:

- *更新:* XQuery Update ファイルが実行されると、ターゲット XML ファイルは直接ディスク上で更新することができます。または、XMLSpy 内で開き、メモリ内で更新することもできます。更新時にファイルを開くオプションにより、ファイルを更新し、ディスクにファイルを保存することができます。また、(保存せずに)更新を拒否することもできます。
- *元のフォーマットを保持する:* 更新されたドキュメントの元のフォーマットを可能な限り保管します。

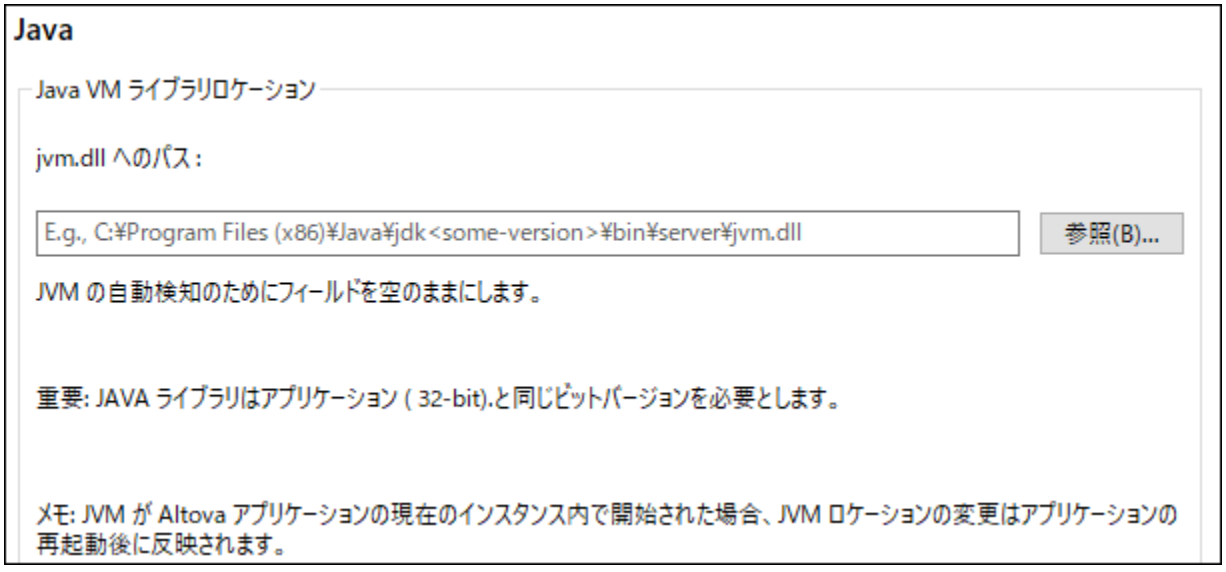
保存して完了します。

設定が完了したら、「OK」をクリックして完了します。

28.17.13.10 Java

Java タブ上では、ファイルシステム上で Java VM (仮想マシン) へのパスを任意で入力することができます。カスタム Java VM パスの追加は常に必要なわけではありません。デフォルトで、XMLSpy は、W indows レジストリと JAVA_HOME 環境変数を(この順序で)読み取ることにより、Java VM パスを自動的に影響しようとします。このダイアログボックスに追加されるカスタムパスは、自動的に検知される VM パスより先優先順位を与えられます。

カスタム Java VM パスを追加する必要がある場合、例えば、インストーラを持たず、(例えば、Oracle の OpenJDK などの) レジストリエントリを作成する必要のない Java 仮想マシンを使用している場合、カスタム Java VM パスを追加する必要がある場合があります。XMLSpy により自動的に検知された Java VM パスを何らかの理由でオーバーライドする場合、このパスを設定する必要があります。



以下の点に注意してください:

- Java VM パスは Altova デスクトップと(サーバーではない) アプリケーションにより共有されています。この結果、アプリケーション内

で変更されると、他のAltova アプリケーションに自動的に適用されます。
- パスはJDK がインストールされているディレクトリに相対する**\bin\server** または**\bin\client** ディレクトリからの**jvm.dll** ファイルを指す必要があります。
- XMLSpy プラットフォーム(32-ビット、64-ビット) がJDK と同じである必要があります。
- Java VM パスの変更後、新規の設定を反映するためにXMLSpy を再起動する必要があります。

Java VM パスの変更は以下のエリアに影響を与えます:

- JDBC 接続性
- XSLT/XPath のためのJava 拡張関数

28.17.13.11 XBRL

XBRL セクションは、XBRL インスタンスドキュメントの検証[1485] とXBRL タクソノミパッケージの追加と管理[1486] のためのオプションを提供します。

28.17.13.11.1 検証

XBRL 検証タブ（*下のスクリーンショップ*）によりXBRL*インスタンス*の検証*のための*オプションが提要されます。

指定することのできる検証オプションは以下の通りです:

- Dimensions 1.0 とUnits Registry 1.0 を使用して適合性をチェックします。無効な使用を警告としてレポートするオプション。
- 計算には以下が含まれます（i）合計アイテムが計算される際の不整合性、（ii）満たされていないアサーション。
- 計算に含まれる重複を考慮しない（完全な重複と不整合性の重複）。ファクトが複製されている個所では、最も正確なファクトが使用され、その他は無視されます。
- 重複のレポート。重複がどのように分類されるかに関する詳細は、 XBRL とインライン XBRL 1.0 内での重複ファクトの扱い 仕様を参照してください。
- （スペース文字、タブ、改行、キャリッジ リターン）空白の処理を指すインライン XBRL 内での空白の正規化。
  - *拡張された空白の正規化* が選択されている場合、改行なしスペースも空白とみなされます。
  - *保管* は空白をそのままにします。
  - *トリム* は文字列の両側の空白を削除します。
  - タブ、改行、キャリッジ リターンのすべての*置換* 発生がスペースと置き換えられます。
  - *折りたたみ* はすべての隣接したスペースが単一のスペースに折りたたまれる *置換* の拡張です。

28.17.13.11.2 タクソノミパッケージ

XBRL タクソノミ パッケージ は、オフラインのXBRL タクソノミのコピーを含むzip 形式のアーカイブです。パッケージは、タクソノミのファイルにURI をマップするカタログ XML ファイルを含み、アプリケーションがタクソノミをオフラインで使用できるようにします。タクソノミ パッケージの指定のルールが構成され、タクソノミ パッケージ内で説明されています。詳細はXBRL 機構の推奨を参照してください。

タクソノミ パッケージのダウンロード後、パッケージのエントリポイントカタログファイルを識別し自動的に使用するようにXMLSpy をセットアップすることができます。これは、アクティブなタクソノミ パッケージのリストにパッケージを追加することによりおこなうことができます。アクティブなパッケージのカタログファイルは、XBRL 検証などのオペレーションのためのリソースを検索するために使用されます。追加可能な2種類のタクソノミ パッケージが存在します:

- Altova XBRL タクソノミマネジャー[722] (詳細はここ[722])を使用して全てのAltova 製品のために簡単かつ便利に管理することのできる標準のパッケージ。
- このダイアログを使用して追加し管理するカスタム パッケージ。

メモ アクティブなパッケージのカタログファイルにより指されるリソースは、そのリソースを必要とするすべてのXMLSpy オペレーションで使用されます。このようなリソースがXMLSpyにより以前使用されたリソースと異なる場合、操作が実行されると結果はエラーを返す場合があります。詳細に関してはこのトピックの下の注意書きを参照してください。

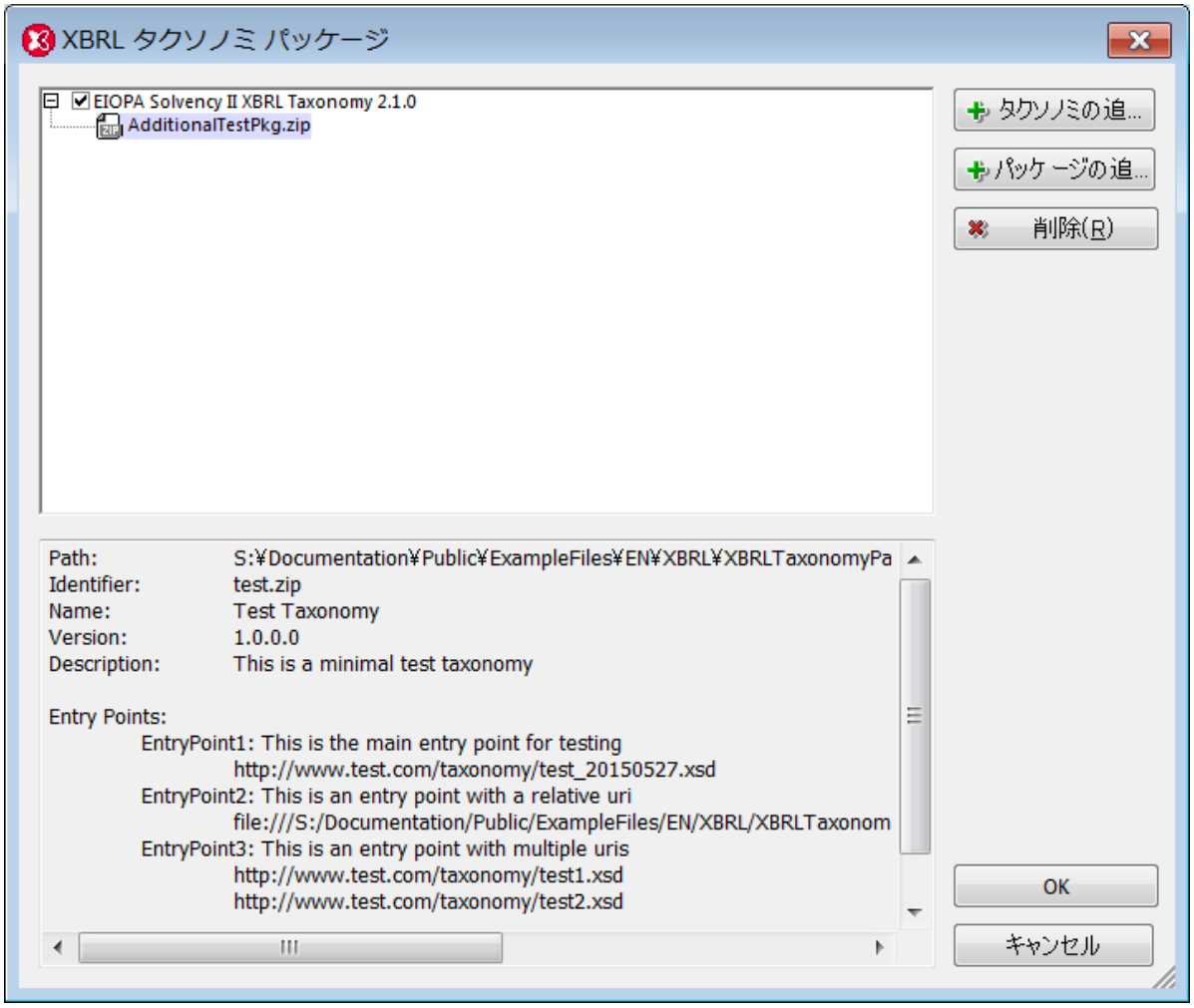
タクソノミ パッケージの追加と管理

[ツール | オプション | タクソノミ パッケージ] を選択してタクソノミ パッケージペインを表示します (下のスクリーンショット)。

標準のタクソノミ パッケージを追加するにはXBRL タクソノミマネジャーをクリックしてください。Altova XBRL タクソノミマネジャー[722]の使用方法に関する詳細はここ[722]の説明を参照してください。

カスタム パッケージを追加するには、以下のステップに従ってください。

1. (*上のスクリーンショット*) 「**編集**」をクリックして、XBRL *タクソノミ パッケージ* ダイアログを表示します (*下のスクリーンショット*)。



2. 「**タクソノミの追加**」をクリックして、*タクソノミ パッケージ*の場所を参照し、「**開く**」をクリックします(複数の*パッケージ*を選択して1度に追加することもできます)。*パッケージ*は、*ダイアログ内のタクソノミ パッケージ*リストに追加されます。リストは2つのレベルのツリーとして表示されます。最初のレベルは*タクソノミ*を表し、2番目のレベルは、*そのタクソノミの パッケージ*を表します。*タクソノミ*エントリの左側の*チェックボックス*は、*そのタクソノミ*が*アクティブ*かどうかを示しています。新規に追加された*タクソノミ*はデフォルトで*アクティブ*化されています。
3. **[OK]** をクリックして完了します。新規に追加された*パッケージ*は、*オプション ダイアログのタクソノミ パッケージ* ペインに表示されます (*上の最初のスクリーンショット*)。

以下の点に注意してください:

- 追加の*パッケージ*を*タクソノミ*に追加する場合、以下を行います: XBRL *タクソノミ パッケージ ダイアログ内のタクソノミ*を選択し、「**パッケージの追加**」ボタンを使用して追加*パッケージ*を追加します (*上のスクリーンショット*)。追加された*パッケージ*は、*そのタクソノミ*の第2のレベルに表示されます。
- XBRL *タクソノミ パッケージ ダイアログのタクソノミ パッケージ*が上のペイン内のリスト内で選択されていると、(オフラインの場所を含む)詳細はダイアログの下のペインに表示されます (*上のスクリーンショット*)。
- *タクソノミ*を無効化するには、*チェックボックス*のチェックを解除してください。*タクソノミ*を無効化するには、*チェックボックス*のチェックを解除します。2つの*タクソノミ*間を切り替えるために、無効化は役に立ちます。
- *パッケージ*を選択し「**削除**」をクリックして削除します。

- 次のAltova アプリケーションは、タクソノミ パッケージ登録をサポートします: XMLSpy、MapForce、およびStyleVision。タクソノミ パッケージリストは、これらのアプリケーションに対して共有することができます。1つのアプリケーション内のリストを編集するには、他のアプリケーション内でも変更されたリストは表示されます。1つのアプリケーション内でパッケージリストが編集されると、同時に開かれている他のアプリケーションで、編集を反映するためにパッケージリストを再ロードするか問われます。

注意点: パッケージ カタログは、互換性の無いリソースにリダイレクトされる可能性があります
アクティブなパッケージのカタログファイルを指すリソースが、リリースに必要とされるすべてのXMLSpy オペレーションのために使用されます。上記のようなリソースの例は、XML 検証とXBRL 検証に使用されるXML スキーマです。パッケージのカタログファイルによりロケートされるオフラインのリソースは、使用中の既存の環境と互換性が無いためにエラーが発生する場合があります。この場合、タクソノミ パッケージを無効化して、エラー情報をパッケージの作成者に伝えてください。

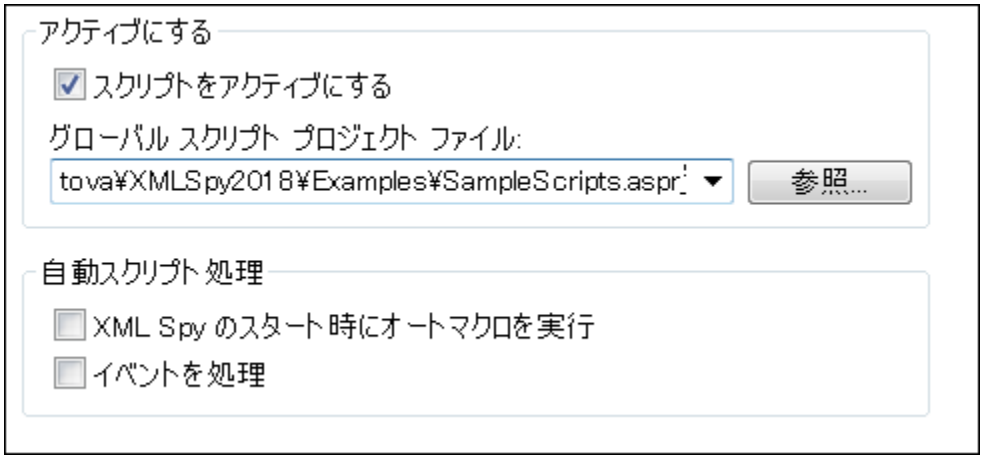
28.17.13.11.3 XULE

XBRL XULE タブは(*下のスクリーンショット*) XULE 処理のためのオプションを提供します。

- *複製ファクトを無視:* HTML コード内で同じファクトが一度以上メモされると、複製ファクトが発生します。通常インライン XBRL 内で発生します。このオプションは複製ファクトが一度出力されるように指定します。
- *出力:* XULE ドキュメントが「XBRL | XULE の実行」コマンドにより処理されると、出力は以下に生成されます: (i) メッセージウィンドウ、または(ii) XMLSpy 内に新規のドキュメントは表示され、一時的にメモリに保管されます。このドキュメントは「ファイル | 名前を付けて保存」[1138] コマンドを使用してファイルに保管されます。

28.17.13.12 スクリプト

スクリプトタブでは、アプリケーションの起動時にスクリプト環境[1413] を有効にすることができます。有効にするには、*スクリプトをアクティブにする* チェックボックスにチェックを入れてください(以下のスクリーンショットを参照)。



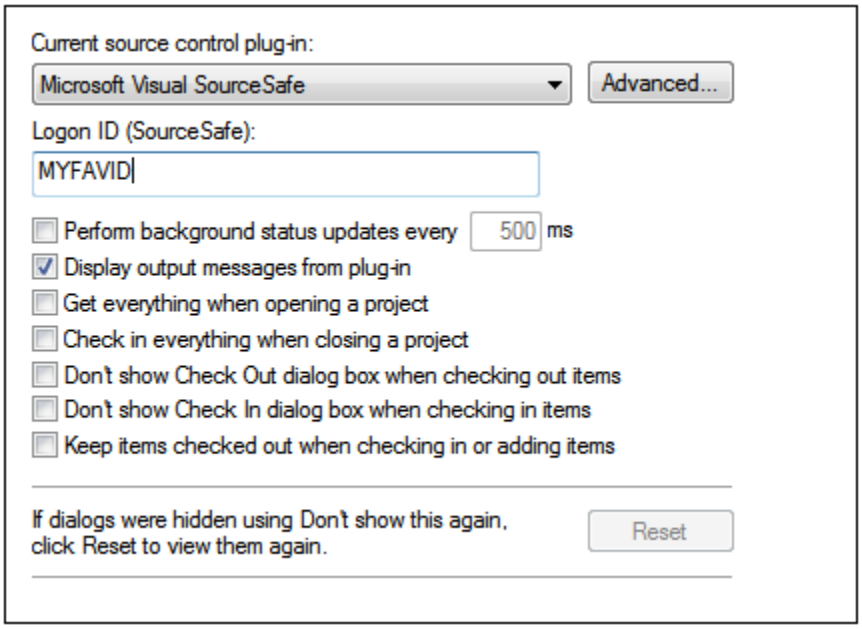
XMLSpy に対してグローバルスクリプトプロジェクトを設定するには、スクリプトをアクティブにするチェックボックスにチェックを入れた後、必要な Altova スクリプトプロジェクト (.asprj) ファイルを指定します。その他にも、(i) XMLSpy の起動時にマクロを自動的に実行するか、(ii) プロジェクト内のアプリケーションイベントハンドラーを自動的に実行するかを有効に(または無効に)することができます。

保存して完了する

「OK」をクリックして完了してください。グローバルリソーススクリプトプロジェクト内のマクロがマクロコマンドのサブメニューに表示されます。

28.17.13.13 ソース管理

ソース管理セクションではソース管理プロバイダーの指定を行い、デフォルトのログイン ID やユーザー設定を各種ソース管理プロバイダーに対して指定することができます。



ソース管理プラグイン

現在インストールされている中から、使用するソース管理プラグインをコンボボックスのドロップダウンリストを使って選択することができます。目的のソース管理を選択したら、次のテキストボックスにてログイン ID を入力します。「**詳細**」ボタンをクリックすることで、選択されたソース管理プラグインに特化したダイアログが表示され、ソース管理プラグインに対して設定を行うことができます。これら設定の内容はソース管理プラグインにより変化します。

ユーザー設定

以下のようなユーザー設定を利用することができます:

- ユーザーが指定した時間間隔ごとに、バックグラウンドでステータスの更新を行うか、この機能を無効にすることもできます。巨大なソース管理データベースを使用する場合、多くの CPU パワーならびにネットワークリソースが消費されることになります。バックグラウンドのステータス更新機能を無効にするか、インターバルの時間を長く取ることで、システムのスピードを図ることができます。
- プロジェクトを開くならびに閉じる時に、それぞれファイルのチェックアウトやチェックインを自動的に行うことができます。
- チェックアウトならびにチェックインを行う際に、ダイアログの表示を無効にすることができます。
- ダイアログボックスにて次回から表示しないオプションを選択した場合、リセットボタンが有効になります。リセットボタンをクリックすることで、ダイアログが再度表示されるようになります。

保存して完了

設定が完了したら、「OK」をクリックして完了します。

28.17.13.14 ネットワーク プロキシ

ネットワーク プロキシ セクションでは、カスタムのプロキシの設定を構成することができます（*下のスクリーンショット*）。デフォルトは、システムのプロキシの設定が使用されますが、設定はユーザーの介入無しで作動します。代替のネットワーク プロキシを設定する場合、設定を定義するオプションを使用してください。

(C) 2015-2021 Altova GmbH

システムプロキシの設定の使用

システムプロキシ設定を介して構成可能な インターネット エクスプローラー（IE） 設定を使用します。`netsh.exe winhttp` を介して構成される設定が必要とされます。

自動プロキシの構成

以下のオプションを使用することができます:

- *自動検知の設定:* DHCP または DNS を使用して W PAD スクリプト（`http://wpad.LOCALDOMAIN/wpad.dat`）を検索し、プロキシセットアップのためにこのスクリプトを使用します。
- *スクリプト URL:* プロキシセットアップのために使用されるプロキシ自動構成（`.pac`）スクリプトに対する HTTP URL を指定します。
- *再ロード:* 現在の自動プロキシ構成をリセットして再ロードします。このアクションには W indows 8 または以降が必要とされ、30 秒程の時間が必要です。

手動のプロキシの構成

ホスト名とポートを対応する製品のプロキシのために手動で指定します。サポートされるスキームはホスト名に含まれている場合があります（例：`http://hostname`）。プロキシがスキームをサポートする場合、対応するプロトコールと同じである必要はありません。

以下のオプションを使用することができます:

- *HTTP プロキシ:* HTTP プロトコルのための指定済みのホスト名とポートを使用します。*全てのプロトコルのためにこのプロキシサーバーを使用する* が選択されている場合、HTTP プロキシがすべてのプロトコルのために使用されます。
- *SSL プロキシ:* SSL プロトコルのために指定されているホスト名とポートが使用されます。
- *プロキシ無し:* セミコロン (`;`) により区別されているプロキシを使用しないホスト名、ドメイン名、または、ホストのための IP アドレスのリスト。IP アドレスは切り捨てられず、IPv6 アドレスは角かっこで囲まれる必要があります (例: `[2606:2800:220:1:248:1893:25c8:1946]`)。ドメイン名は、ドットと共に開始される必要があります (例: `.example.com`).
- *プロキシのサーバーをローカルのアドレスのために使用しない:* チェックされている場合、*プロキシ無し* リストのために `<local>` を追加します。このオプションが選択されている場合、次の場合、プロキシは使用されません: (i) `127.0.0.1`、(ii) [::1]、(iii) (`.`) ドット文字を含んでいないホスト名すべて。

現在のプロキシの設定

プロキシの検知の詳細なログを提供します。*URL のテスト* フィールドの右の「**更新**」ボタンを使用して更新することができます (例、URL のテスト を変更する場合、または、プロキシの設定が変更された場合)。

- *URL のテスト:* URL のテストを使用して、どのプロキシが特定の URL ために使用されているかを確認することができます。URL を使用して、I/O はされません。プロキシの自動構成が選択されている場合、このフィールドは空にしておく必要があります (*システムプロキシの設定の使用*、または、*自動プロキシの構成* を使用して)。

(C) 2015-2021 Altova GmbH

28.18 ウィンドウメニュー

ウィンドウメニューには、殆どのWindows アプリケーションに備わっているコマンドが含まれており、XMLSpy セッションで開かれた個々のドキュメントウィンドウを管理することができます。

開かれているドキュメントウィンドウを重ねて表示したり、上下や左右に並べて表示、最小化したドキュメントウィンドウへアクセスすることができます。各種入力ヘルパーを有効/無効にすることができるほか、ウィンドウダイアログから、目的のドキュメントウィンドウへ直接アクセスすることもできます。

28.18.1 重ねて表示

このコマンドにより開かれている全てのドキュメントウィンドウが重ね合わさって表示されるように配置されます。

28.18.2 上下に並べて表示

このコマンドにより、開かれている全てのドキュメントが上下に並べて配置され、複数のドキュメントを同時に閲覧することができます。

28.18.3 左右に並べて表示

このコマンドにより、開かれている全てのドキュメントが左右に並べて配置され、複数のドキュメントを同時に閲覧することができます。

28.18.4 プロジェクトウィンドウ

このコマンドによりプロジェクトウィンドウ[110] の表示/非表示を切り替えることができます。

プロジェクトウィンドウはドッキングすることのできるウィンドウで、タイトルバーをドラッグすることで、オリジナルの位置から切り離してフロートウィンドウにすることができます。タイトルバーを右クリックしてウィンドウをドッキングしたり隠したりすることができます。

28.18.5 情報ウィンドウ

このコマンドにより情報ウィンドウ[112]の表示/非表示を切り替えることができます。

情報ウィンドウはドッキングすることのできるウィンドウで、タイトルバーをドラッグすることで、オリジナルの位置から切り離してフロートウィンドウにすることができます。タイトルバーを右クリックしてウィンドウをドッキングしたり隠したりすることができます。

28.18.6 入力ヘルパー

このコマンドにより全ての入力ヘルパーの表示/非表示を切り替えることができます。

3つある入力ヘルパーウィンドウはドッキングすることのできるウィンドウで、タイトルバーをドラッグすることで、オリジナルの位置から切り離してフロートウィンドウにすることができます。タイトルバーを右クリックしてウィンドウをドッキングしたり隠したりすることができます。

28.18.7 出力ウィンドウ

出力ウィンドウは、(検証結果のようなメッセージを表示する)メッセージウィンドウや、(XPath 条件式評価の結果を表示する)XPath ウィンドウをタブ化したウィンドウです。初期状態ではメインウィンドウの下部に表示されます。「**出力ウィンドウ**」コマンドにより、出力ウィンドウの表示/非表示を切り替えることができます。

出力ウィンドウのウィンドウはドッキングすることのできるウィンドウで、タイトルバーをドラッグすることで、オリジナルの位置から切り離してフロートウィンドウにすることができます。タイトルバーを右クリックしてウィンドウをドッキングしたり隠したりすることができます。

出力ウィンドウに関する詳細は、テキストビューセクションの出力ウィンドウ[108]を参照ください。

28.18.8 プロジェクトおよび入力ヘルパー

このコマンドによりプロジェクトウィンドウならびに入力ヘルパーの表示/非表示を選択することができます。

28.18.9 全てオン/オフ

このコマンドにより、以下にあるドッキング可能な全てのウィンドウの表示/非表示を切り替えることができます:

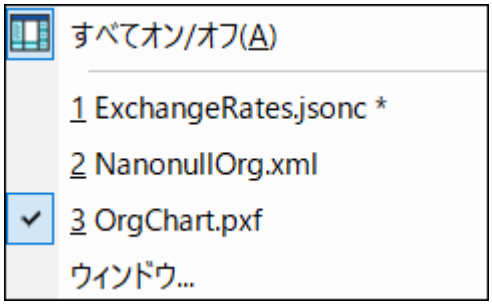
- プロジェクトウィンドウ[110]
- 情報ウィンドウ[112]

- 3つの入力ヘルパーウィンドウ
- 出力ウィンドウ

このコマンドによりドキュメント以外の全ウィンドウを素早く隠して、ドキュメントの表示を作業エリア全体に広げることができます。

28.18.10 現在開かれているウィンドウのリスト

このリストは現在開かれているすべてのウィンドウを表示し、ウィンドウ間で素早く切り替えることができます。



また**CTRL+F6** キーボードショートカットを使用して、開かれているウィンドウ間を循環することができます。

28.19 ヘルプメニュー

ヘルプメニューにはヘルプならびにXMLSpy に関する更に詳しい情報を得るために必要なコマンドが、ウェブサーバー上で利用できる情報へのリンクや、サポートページへのリンクとともに収められています。

ヘルプメニューからは登録ダイアログ[1498]を起動することもでき、購入した製品のライセンスキーを入力することができます。

28.19.1 目次、インデックス、検索

- 目次
  - *説明*

    ヘルプウィンドウの左側のペインに目次を表示した、XMLSpy の画面上のヘルプマニュアルを開きます。目次はヘルプドキュメント全体の概要を表示しています。目次のエントリをクリックしてトピックに移動することができます。

- インデックス
  - *説明*

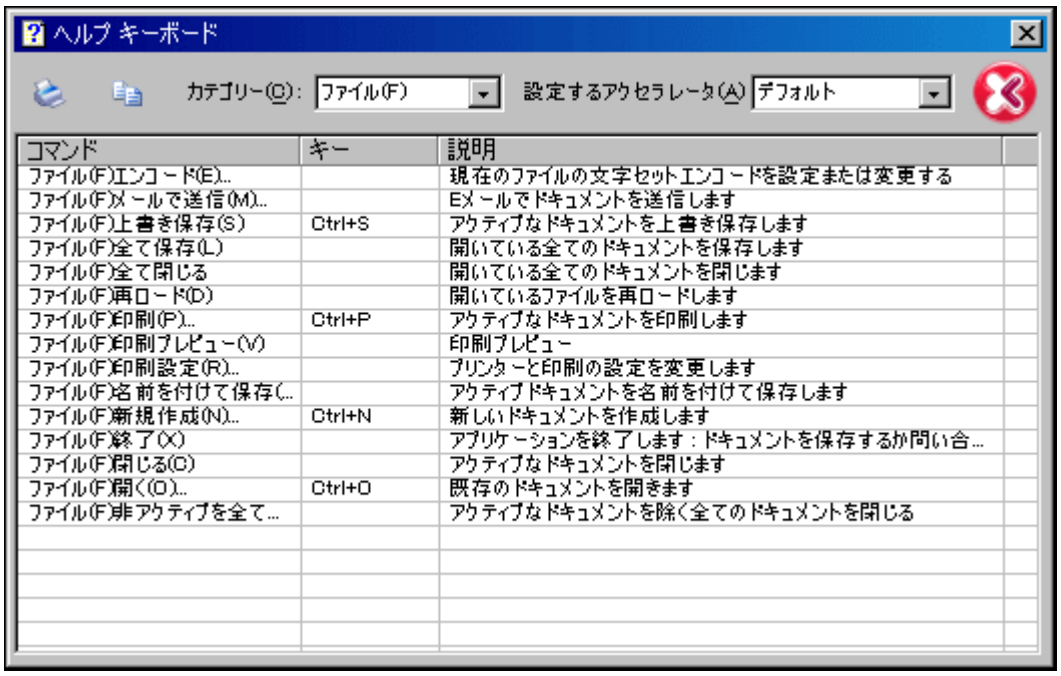
    ヘルプウィンドウの左側のペインにキーワード インデックスを表示したXMLSpy の画面上のヘルプマニュアルを開きます。目次はヘルプドキュメント全体の概要を表示しています。インデックスはキーワードをリストし、キーワードをダブルクリックすることでトピックへ移動することができます。キーワードが 1 つ以上のトピックにリンクされている場合は、トピックのリストが表示されます。

- 検索
  - *説明*

ヘルプウィンドウの左側のペインに検索ダイアログを表示したXMLSpy の画面上のヘルプマニュアルを開きます。単語を検索するには、入力フィールドに検索対象を入力して、(i)「Return」を押す、または(ii)「**トピックのリスト**」をクリックします。を押します。ヘルプシステムは、ヘルプドキュメント全体で全文検索を行いヒットしたリストを返します。アイテムを表示するためにはアイテムをダブルクリックします。

28.19.2 キーボードマップ

「ヘルプ | キーボードマップ」コマンドにより、XMLSpy 内にある全てのコマンドが、メニューごとに分けられて表示されます。各メニューコマンドには、その説明とキーボードショートカットが表示されます。



コマンド	キー	説明
ファイル(F)エンコード(E)...		現在のファイルの文字セットエンコードを設定または変更する
ファイル(F)メールで送信(M)...		Eメールでドキュメントを送信します
ファイル(F)上書き保存(S)	Ctrl+S	アクティブなドキュメントを上書き保存します
ファイル(F)全て保存(L)		開いている全てのドキュメントを保存します
ファイル(F)全て閉じる		開いている全てのドキュメントを閉じます
ファイル(F)再ロード(D)		開いているファイルを再ロードします
ファイル(F)印刷(P)...	Ctrl+P	アクティブなドキュメントを印刷します
ファイル(F)印刷プレビュー(V)		印刷プレビュー
ファイル(F)印刷設定(R)...		プリンターと印刷の設定を変更します
ファイル(F)名前を付けて保存(...		アクティブドキュメントを名前を付けて保存します
ファイル(F)新規作成(N)...	Ctrl+N	新しいドキュメントを作成します
ファイル(F)終了(X)		アプリケーションを終了します：ドキュメントを保存するか問い合...
ファイル(F)閉じる(C)		アクティブなドキュメントを閉じます
ファイル(F)開く(O)...	Ctrl+O	既存のドキュメントを開きます
ファイル(F)非アクティブを全て...		アクティブなドキュメントを除く全てのドキュメントを閉じる

特定のメニューに収められているコマンドを確認するには、カテゴリーコンボボックスからメニュー名を選択します。印刷アイコンをクリックすることで、コマンドを印刷することができます。

ショートカットに関するメモ:

- いくつかのコマンド（及びそのキーボードショートカット）は、特定のビューでしか有効になりません。例えば、XMLメニューにあるほとんどのコマンドは、グリッドビューでしか使用できません。その他のコマンド（例：「**ファイル | 保存**」や「**XML | 整形式のチェック**」）は複数のビューで使用することができます。
- 例えば、テキストビューやスキーマビューで「Shift + F10」を押下することでコンテキストメニューを表示したり、テキストビューにてカーソルが要素の開始または終了タグにあるときに「Ctrl + E」をう押下することで、対となるタグへ移動することができます。
- カスタマイズダイアログの キーボードタブ[1443] では、各種メニューコマンドに対して独自のショートカットをセットすることができます。

28.19.3 ライセンス登録、注文フォーム、登録、最新情報のチェック

▼ ソフトウェアのライセンス認証

説明

Altova 製品ソフトウェアをダウンロードすると、無料評価キーまたは購入されたライセンスキーを使用して、製品にライセンスを供与、または、ライセンスの認証を行うことができます。

- *無料 評価ライセンス*:初めて製品のダウンロードとインストールを行うと、ソフトウェアライセンス認証ダイアログが表示されます。ダイアログでは無料 評価 ライセンスをリクエストすることができます。ユーザーの名前、所属会社名、そして電子メールアドレスを表示されるダイアログに入力し **リクエスト** をクリックします。ライセンスファイルが入力された電子メールアドレスに送信されます。この手順には数分を要します。ライセンスファイルを適切な場所に保存します。**リクエスト** をクリックすると、リクエストダイアログの下に入力フィールドが表示されます。このフィールドはライセンスファイルの パスを取ります。ライセンスファイルを参照 またはライセンスファイルへの パスを入力し「OK」をクリックします。(「ソフトウェアのライセンス認証ダイアログ」内で「**新規のライセンスをアップロードする**」をクリックしてライセンスファイルへの パスを入力するダイアログにアクセスすることができます。)ソフトウェアは30日の間アンロックされます。
- *永続的なライセンスキー*:ソフトウェアライセンス認証ダイアログには永続的なライセンスキーを購入するためのボタンが含まれています。このボタンをクリックすると、製品の永続的なライセンスキーを購入することのできる Altova オンラインショップに移動することができます。受信する電子メールはライセンスデータを含むライセンスファイルの書式で送信されています。3つの種類の永続的なライセンスが存在します: *インストール済み*、同時使用ユーザー、名前を持つユーザー。*インストール済みのライセンス* は単一のコンピューター上のソフトウェアのロックを解除します。*インストール済みのライセンス* を N 台のコンピューターのために購入すると N 台までのコンピューター上でソフトウェアを使用することができます。*同時使用ユーザーライセンス* は N 人の同時使用ユーザーに N 人のユーザーに同時にソフトウェアを使用することを許可します。(10N 台のコンピューターにソフトウェアをインストールすることができます)。 *名前を持つユーザーライセンス* は*特定のユーザー*が5台の異なるコンピューター上でソフトウェアを使用することを許可します。ソフトウェアのライセンスを認証するには「**新規のライセンスをアップロードする**」をクリックして表示されるダイアログ内でライセンスファイルを参照、またはライセンスファイルの パスを入力し「OK」をクリックします。

メモ 複数のユーザーライセンスのために各ユーザーは各自の名前を入力するようにプロンプトされます。

ライセンスの電子メールと Altova 製品へのライセンス供与(有効化) の異なる方法
Altova から受信するライセンス電子メールにはライセンスファイルが添付ファイルとして含まれています。ライセンスファイルは `.altova_licenses` ファイル拡張子を有しています。

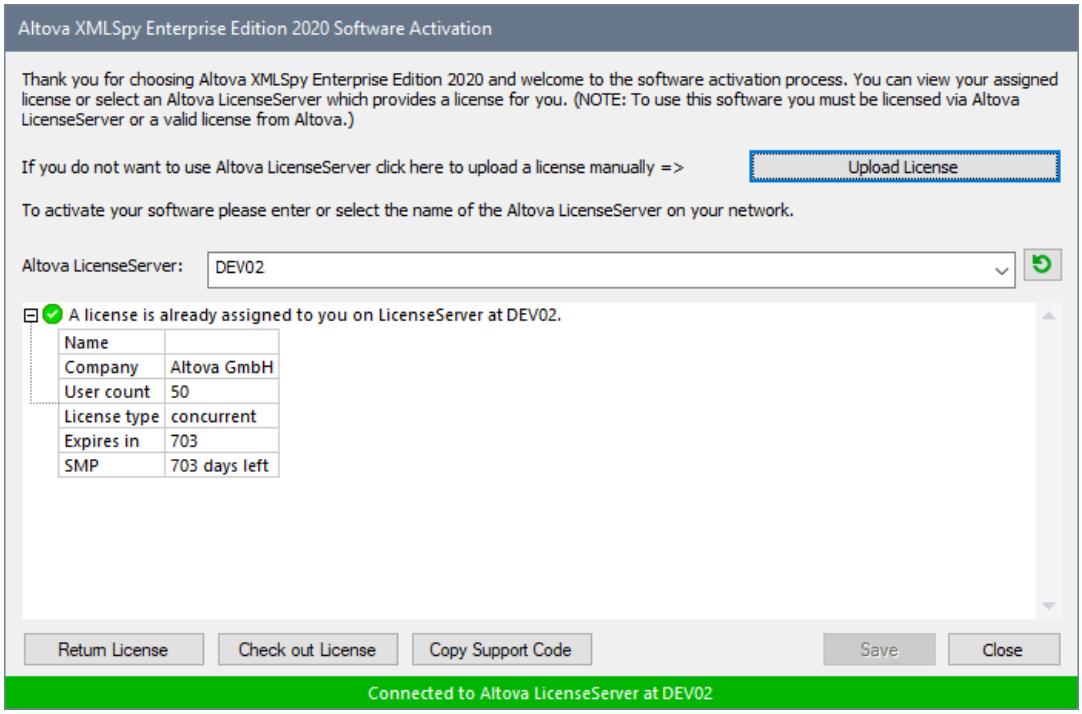
Altova 製品のライセンスを認証するには、以下のうち1つ行ってください:

- 適切な場所にライセンスファイル(`.altova_licenses`)を保存し、ライセンスファイルをダブルクリックし、表示されるダイアログに必要な情報を入力し、「**キーの適用**」をクリックして完了します。
- ライセンスファイル(`.altova_licenses`)を適切な場所に保存します。Altova 製品内では、「**ヘルプ | ライセンス登録**」メニューコマンドを選択し、**新規のライセンスをアップロード**します。ライセンスファイルへの パスを入力し「OK」をクリックします。
- 適切な場所にライセンスファイル(`.altova_licenses`)を保存し、Altova LicenseServer のライセンスプールにアップロードします。以下を行うことができます: (i) 製品のソフトウェアライセンス認証ダイアログから Altova 製品からライセンスを取得します。*(以下を参照)*または(ii) Altova LicenseServer から製品へのライセンスを割り当てます。*LicenseServer の使用の詳細に関しては、下記のトピックを参照してください*。

ソフトウェアライセンス認証ダイアログ(*下のスクリーンショット*) は、「**ヘルプ | ソフトウェア アクティベーション**」をクリックすることにより常にアクセスすることができます。

以下の方法によりソフトウェアをアクティブ化することができます:

- *ソフトウェアライセンス認証ダイアログでライセンスを登録する方法。*ダイアログ内で、「**新規のライセンスをアップロード**」をクリックして、ライセンスファイルを参照し選択します。「OK」をクリックしてライセンスファイルへのパスを確認します(複数のユーザーライセンスの場合は個人の名前です)。「**保存**」をクリックして完了します。
- *ネットワーク上のAltova LicenseServerを使用してライセンス供与する方法:* ネットワーク上のAltova LicenseServerを使用してライセンスを取得するには*ソフトウェアのライセンスの認証ダイアログの下にある***Altova LicenseServerを使用する**をクリックします。使用するLicenseServerがインストールされているマシンを選択します。License Serversの自動検知はLAN上で配信が送信されることを意味します。これらの配信がサブネットに制限されているため、License Serverは自動検知のためのクライアントマシンと同じサブネット上に存在する必要があります。自動検知が作動しない場合、サーバーの名前を入力します。Altova LicenseServerはライセンスプール内でAltova製品のためのライセンスを有している必要があります。LicenseServerプール内に存在する場合、ソフトウェアライセンス認証ダイアログ内に表示されます(*Altova XMLSpy内のダイアログで表示されている例を参照してください*)。「**保存**」をクリックしてライセンスを取得します。



マシン固有のライセンスがLicenseServerからインストールされると、7日間は、LicenseServerに戻すことができません。7日過ぎると、(「**ライセンスを戻す**」をクリックして) マシンのライセンスをLicenseServerに戻すことができ、このライセンスは、他のクライアントによりLicenseServerから取得することができます。LicenseServer管理者は、LicenseServerのWeb UIを使用して、取得されたライセンスの割り当てを解除することができます。ライセンスの返却は、マシン固有のライセンスのみに適用され、現在使用中のライセンスには適用されないことにご注意してください。

ライセンスのチェックアウト
ライセンスが製品マシン上に保管されるように、ライセンスをライセンスプールから30日間チェックアウトすることができます。これにより、オフラインで作業することが可能になります。この機能はとても役に立ちます。Altova LicenseServerにアクセスできない環境(例えば、旅行中にAltova製品がインストールされたラップトップコンピューターで作業する)場合などが挙げられます。ライセンスはチェックアウトされていますが、LicenseServerは、ライセンスが使用中と表示し、ライセンスは他のマシンで使用することができません。ライセンスはチェックアウトの期間が終わると自動的にチェックインされた状態を戻します。または、チェックアウトされたライセンスはソフトウェアのライセンスの認証ダイア

ログのボタンを使用して「**チェックイン**」することができます。

ライセンスをチェックアウトするには以下をおこないます: (i) ソフトウェアのライセンスの認証ダイアログで「**ライセンスのチェックアウト**」をクリックします (*上のスクリーンショット参照*)。(ii) ライセンスのチェックアウトダイアログ内から、チェックアウトの期間を選択し、「**チェックアウト**」をクリックします。ライセンスがチェックアウトされます。ライセンスのチェックアウト後2つの状態が発生します: (i) ソフトウェアのライセンス認証ダイアログは時刻およびチェックアウトの期限を含むチェックアウトに関する情報を表示します。(ii) ダイアログ内の「**ライセンスのチェックアウト**」ボタンは「**チェックイン**」ボタンに変更されます。「**チェックイン**」ボタンをクリックして、ライセンスをチェックインすることができます。チェックアウト期間の期限が切れると、ライセンスは自動的にチェックイン状態に戻されるため、選択したチェックアウトの期間がオフラインで作業する期間をカバーするように確認してください。

メモ ライセンスのチェックアウトを可能にするには、LicenseServer 上でチェックアウト機能が有効化されている必要があります。チェックアウトを試みる際この機能が有効化されていない場合、エラーメッセージが表示されます。この場合、LicenseServer 管理者に連絡してください。

サポートコードのコピー
「**サポートコードのコピー**」をクリックして、ライセンスの詳細をクリップボードにコピーしてください。これはオンラインサポートフォームを使用してサポートをリクエストする際に必要なデータです。

Altova LicenseServer を使用することにより、IT 管理者は、リアルタイムでネットワーク上の全てのライセンスの概要、および、クライアントの割り当てと、クライアントのライセンスの使用状況を確認することができます。LicenseServer を使用する利点は、ですから、多数の Altova ライセンスを管理することのできる管理機能です。Altova LicenseServer は、Altova Web サイトで無料で提供されています。Altova LicenseServer および Altova LicenseServer を使用したライセンスの供与に関する詳細は、Altova LicenseServer ドキュメントを参照してください。

注文フォーム

説明

ソフトウェア製品のライセンス許与バージョンを注文する準備が整っている場合、(*前のセクション参照*) ソフトウェアライセンス認証ダイアログ内の「**永久ライセンスの購入**」ボタン、または「**注文フォーム**」コマンドを使用して Altova オンラインショップに移動して注文することができます。

登録

説明

Altova 製品登録ページをブラウザーのタブに表示します。Altova ソフトウェアを登録することにより、最新の製品の情報が得られます。

更新のチェック

説明

Altova サーバーに接続して、より新しいバージョンの製品が利用可能かどうかチェックし、その結果を表示します。

28.19.4 他のコマンド

- サポートセンター
  - *説明*

    インターネット上にあるAltova サポートセンターへのリンクとなっています。サポートセンターにはFAQ やディスカッションフォーラムが含まれており、問題の解決方法を探り、Altova の技術サポートスタッフへアクセスすることができます(現在英語のみの提供となります)。

- WEB 上のFAQ
  - *説明*

    インターネット上にあるAltova のFAQ へのリンクとなっています。FAQ データベースはAltova のサポートスタッフにより常時更新されています。

- コンポーネントのダウンロード
  - *説明*

    インターネット上にあるAltova のコンポーネントダウンロードセンターへのリンクとなっています。このリンク先から様々なコンポーネントソフトウェアをダウンロードして、Altova 製品とともに使用することができます。ソフトウェアコンポーネントはXSLT やXSL-FO プロセッサーからアプリケーションサービスプラットフォームまで、幅広く提供されています。コンポーネントダウンロードセンターにてご利用になれるソフトウェアは、通常無料でご利用になれます。

- インターネット上のXMLSpy
  - *説明*

    インターネット上にあるAltova Web サイトへのリンクとなっています。Altova Web サイトでは、XMLSpy や関連するテクノロジーについて確認することができます。

- XMLSpy トレーニング
  - *説明*

    Altova Web サイトでオンライントレーニングページを見つけることができますAltova の専門家トレーナーによるオンラインコースを選択することができます。

- XMLSpy について
  - *説明*

    スプラッシュ画面と製品のバージョン番号が表示されます。XMLSpy の64 ビットバージョンを使用している場合、これはアプリケーション名の後のサフィックス(x64) により示されています。32 ビットバージョンにはサフィックスは存在しません。

28.20 コマンドライン

コマンドラインからXMLSpy に搭載されているいくつかのコマンドを実行することができます。以下のコマンドを利用することができます:

ファイルを開く

コマンド: `xmlspy.exe file.xml`
動作: XMLSpy にてファイルfile.xml を開く。

メモ: XML ファイルに SPS が割り当てられている場合、XML ファイルは Authentic ビューにて開かれます。そうでない場合、XML ファイルはテキストビューにて開かれます。SPS ファイルが割り当てられていない場合、/sps フラグにより割り当てを行うことができます（下記参照）。

複数のファイルを開く

コマンド: `xmlspy.exe file1.XML file2.xml`
動作: XMLSpy にてファイルfile1.xml ならびにfile2.xml を開く。

SPS ファイルを XML ファイルに割り当てて、Authentic ビューによる編集を行う

コマンド: `xmlspy.exe myxml.xml /sps mysps.sps`
動作: mysps.sps を SPS ファイルとして、ファイルmyxml.xml を Authentic ビューで開く。/sps フラグにより、あとに続く SPS ファイルが/sps フラグの前にある XML ファイルに（Authentic ビュー編集を行うために）割り当てられます。

SPS ファイルから新たな XML テンプレートを開く

コマンド: `xmlspy.exe mysps.sps`
動作: 新規 XML ファイルを Authentic ビューで開きます。表示はSPS をベースにしたもので、SPS スキーマをベースにした新規 XML ファイルが作成されます。XML ファイルの保存を行う際に、ファイルの名前を指定する必要があります。

テキストビューで SPS ファイルを XML ドキュメントとして開く

コマンド: `xmlspy.exe /raw mysps.sps`
動作： テキストビューで、ファイルmysps.sps を XML ドキュメントとして開きます。/raw フラグにより、それに続く SPS ファイルが XML ファイルとして編集されます。

(C) 2015-2021 Altova GmbH

29 プログラマーのレファレンス

XMLSpy は自動化サーバーです: 自動化クライアントは直接にオブジェクトおよび自動化サーバーにより使用することのできる機能にアクセスすることができます。自動化クライアントは直接にオブジェクトおよび自動化サーバーにより使用することのできる機能にアクセスすることができます。ですから、XMLSpy の自動化クライアントは、例えば、XMLSpyのXML 検証機能を使用することができます。。この結果、開発者は、XMLSpy の既存の機能を使用してアプリケーションを拡張することができます。

XMLSpy, のプログラムすることのできるオブジェクトは、XMLSpy のアプリケーション API を使用して、自動化クライアントで使用することができます。XMLSpy のアプリケーション API は、以降省略してアプリケーション API と呼ばれます。アプリケーション API のオブジェクトモデル、および使用可能なすべてのオブジェクトの詳細はこのドキュメントで説明されています(Application API[1544] のセクションを参照してください)。

実行環境

アプリケーション API は、以下の環境の内部からアクセスすることができます:

- スクリプトエディター[1505]
- IDE プラグイン[1530]
- 外部プログラム[1544]
- ActiveX 統合[1905]

これらの環境について下で簡単に説明がされています。

スクリプトエディター: XMLSpy 機能のカスタム化と変更
XMLSpy のインストールを機能を変更および追加することでカスタム化することができます。ユーザー入力のためのフォームを作成し、新しいメニューコマンド およびツールバーショートカットを含むユーザーインターフェイスを変更することもできます。これらすべての機能は、アプリケーション APIのオブジェクトとの作動するスクリプトを書くことにより達成することができます。これらのタスクを効率的に実行するために、XMLSpy は、ビルドインのスクリプトエディターを搭載しています。機能の完全な説明と使用方法はこのドキュメントのスクリプトエディター[1505] セクションで説明されています。サポートされるプログラミング言語は、**JScript** および **VBScript**. です。

IDE プラグイン: XMLSpy のためのプラグインの作成
XMLSpy により自身のプラグインを作成し、XMLSpy に統合することができます。これを行うには、**XMLSpy** のプラグインのための特別なインターフェイスを使用します。プラグイン作成に関する詳細は、XMLSpy IDE プラグイン[1530] のセクションで説明されています。

アプリケーションオブジェクトは、IDE プラグインにより実装され、アプリケーションにより呼び出される必要があるメソッドにパスされます。IDE プラグインの実装のために使用される 典型的な言語は、C# および C++ です。詳細に関しては、XMLSpy IDE プラグイン[1530] のセクションを参照してください。

外部プログラム

さらに、外部スクリプトともに**XMLSpy** を操作することができます。例えば、XMLSpy を指定した時刻に起動し、**XMLSpy** 内のファイルを開き、ファイルを検証し、印刷するスクリプトを作成することができます。外部スクリプトは、アプリケーション API を使用して、これらのタスクを実行します。アプリケーション API の詳細に関しては、アプリケーション API[1544] のセクションを参照してください。

XMLSpy から Application API を使用するには、XMLSpy のインスタンスが最初に開始される必要があります。そして、Authentic Desktop インスタンスと関連する Application オブジェクトを返します。個別の言語に関する情報は、プログラミング言語[1546] のセクションを参照してください。

基本的には、XMLSpy はCOM 登録を使用して起動されます。そして、XMLSpy インスタンスと関連する `Application` オブジェクトを返します。COM 設定により、既に実行中のXMLSpy に関連したオブジェクトが返されます。Access to COM オブジェクトへのアクセスは、自動的にラップされたC# を使用します。共通する項目は、下にリストされています。

- JScript[1567] およびVBScript[1555] スクリプト ファイルは、シンプルな構文を持ち、COM オブジェクトにアクセスするためにデザインされています。DOS コマンド ラインから直接、またはW indows Explorer をダブルクリックして実行することができます。これらのスクリプトはシンプルな 自動化 タスクのために使用されます。
- C#[1558] は、広範囲の既存の機能を搭載した完全なプログラミング言語です。Access to COM オブジェクトへのアクセスは、自動的にラップされたC# を使用します。
- C++ は、COM への直接的なコントロールを与えますが、他の言語よりも多量のコードが必要です。
- Java[1567]: Altova 製品には、アプリケーションAPI をラップして、Java の完全な外観と使用感を与える、ネイティブのJava クラスが搭載されています。
- 役に立つ代替の他のプログラミング言語は、次のとおりです: アプリケーション、Perl、およびPython のためのVisual Basic

ActiveX 統合

XMLSpy ActiveX コントロールを介してApplication API にアクセスする特別なケースです。この機能はXMLSpy 統合パッケージ[1905] がインストールされている場合のみ使用することができます。すべてのActiveX コントロールには、元になる機能のための対応するCOM オブジェクトを返すプロパティがあります。マネージャーコントロールは `Application` オブジェクト、ドキュメントコントロール、`Document` オブジェクト、およびプレースホルダオブジェクトを与え、プロジェクトツリーを含む箇所では、`Project` オブジェクトを返します。これらのオブジェクトによりサポートされるメソッドは、アプリケーションAPI のインターフェイスのセクション[1579]で説明されています。ActiveX コントロール統合 のコンテキストでは、このメソッドを使用する必要がなくこの点を考慮する必要があります。詳細に関してはActiveX 統合[1905]を参照してください。

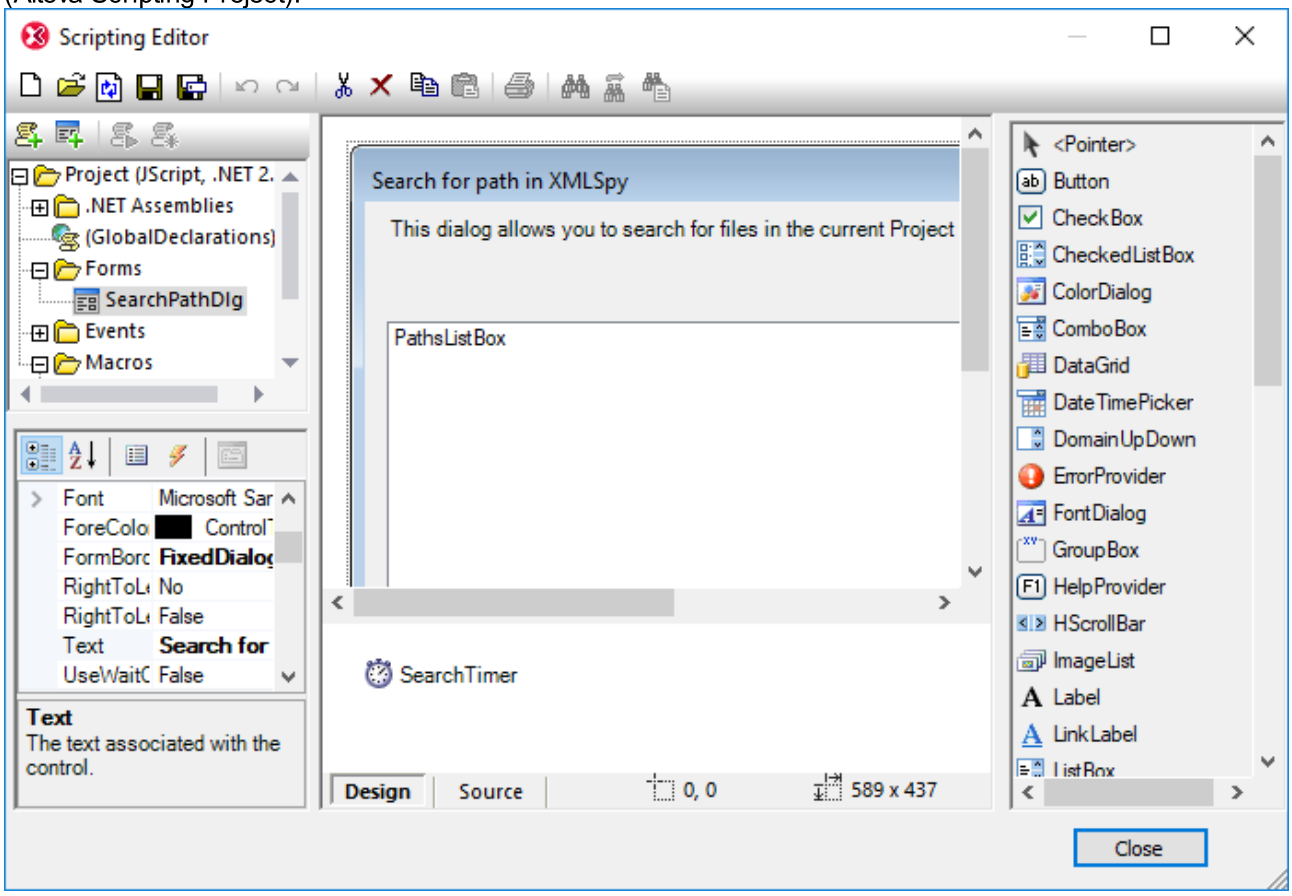
プログラマーのレファレンスについて

XMLSpy のプログラマーのレファレンスは、次のセクションから構成されています:

- スクリプトエディター[1505]:XMLSpy で使用することのできるスクリプト環境のためのユーザー参照。
- IDE プラグイン[1530]:XMLSpy のための プラグイン作成の説明
- Application API[1544]:アプリケーションAPI のためのレファレンス
- ActiveX 統合[1905]:ActiveX コントロールを使用した、XMLSpy GUI およびXMLSpy 機能の統合の方法の案内および参照。

29.1 Scripting Editor

Scripting Editor is a development environment built into XMLSpy from where you can customize the functionality of XMLSpy with the help of JScript or VBScript scripts. For example, you can add a new menu item to perform a custom project task, or you can have XMLSpy trigger some behavior each time when a document is opened or closed. To make this possible, you create scripting projects—files with .asprj extension (Altova Scripting Project).



Scripting Editor

Scripting projects typically include one or several macros—these are programs that perform miscellaneous custom tasks when invoked. You can run macros either explicitly from a menu item (or a toolbar button, if configured), or you can set up a macro to run automatically whenever XMLSpy starts. The scripting environment also integrates with the XMLSpy COM API. For example, your VBScript or JScript scripts can handle application or document events such as starting or shutting down XMLSpy, opening or closing a project, and so on. Scripting projects can include Windows Forms that you can design visually, in a way similar to Visual Studio. In addition, several built-in commands are available that help you instantiate and use .NET classes from VBScript or JScript code.

Once your scripting project is complete, you can enable it either globally in XMLSpy, or only for specific projects.

Scripting Editor requires .NET Framework 2.0 or later to be installed before XMLSpy is installed.

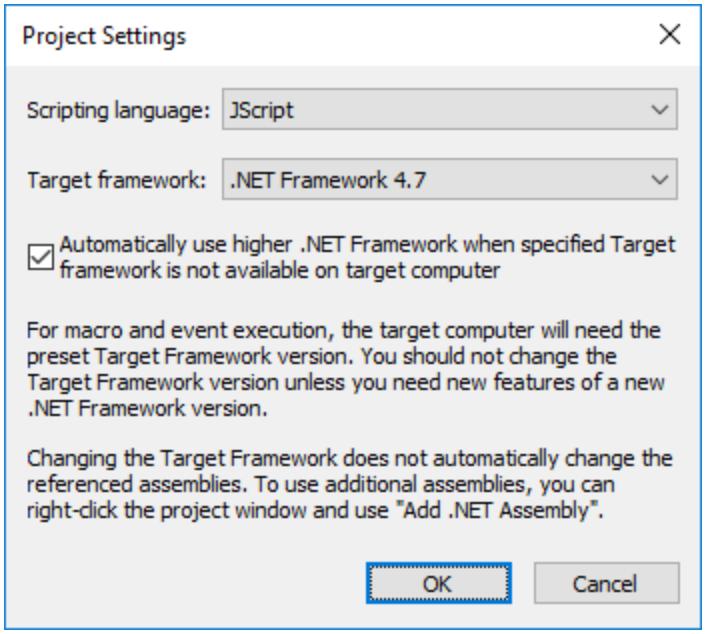
29.1.1 Creating a Scripting Project

All scripts and scripting information created in the Scripting Editor are stored in Altova Scripting Projects (.asprj files). A scripting project may contain macros, application event handlers, and forms (which can have their own event handlers). In addition, you can add global variables and functions to a "Global Declarations" script—this makes such variables and functions accessible across the entire project.

To start a new project, run the menu command **Tools | Scripting Editor**.

The languages supported for use in a scripting project are JScript and VBScript (not to be confused with Visual Basic, which is not supported). These scripting engines are available by default on Windows and have no special requirements to run. You can select a scripting language as follows:

1. Right-click the **Project** item in the upper-left pane, and select **Project settings** from the context menu.
2. Select a language (JScript or VBScript), and click **OK**.



From the Project settings dialog box above, you can also change the target .NET Framework version. This is typically necessary if your scripting project requires features available in a newer .NET Framework version. Note that any clients using your scripting project will need to have the same .NET Framework version installed (or a later compatible version).

By default, a scripting project references several .NET assemblies, like `System`, `System.Data`, `System.Windows.Forms`, and others. If necessary, you can import additional .NET assemblies, including assemblies from .NET Global Assembly Cache (GAC) or custom .dll files. You can import assemblies as follows:

1. Statically, by adding them manually to the project. Right-click **Project** in the top-left pane, and select **Add .NET Assembly** from the context menu.
2. Dynamically, at runtime, by calling the `CLR.LoadAssembly` [1521] command from the code.

You can create multiple scripting projects if necessary. You can save a scripting project to the disk, and then load it back into the Scripting Editor later. To do this, use the standard Windows buttons available in the toolbar: **New**, **Open**, **Save**, **Save As**. Once the scripting project has been tested and is ready for deployment,

you can load it into XMLSpy and run any of its macros or event handlers. For more information, see Enabling Scripts and Macros 1528.

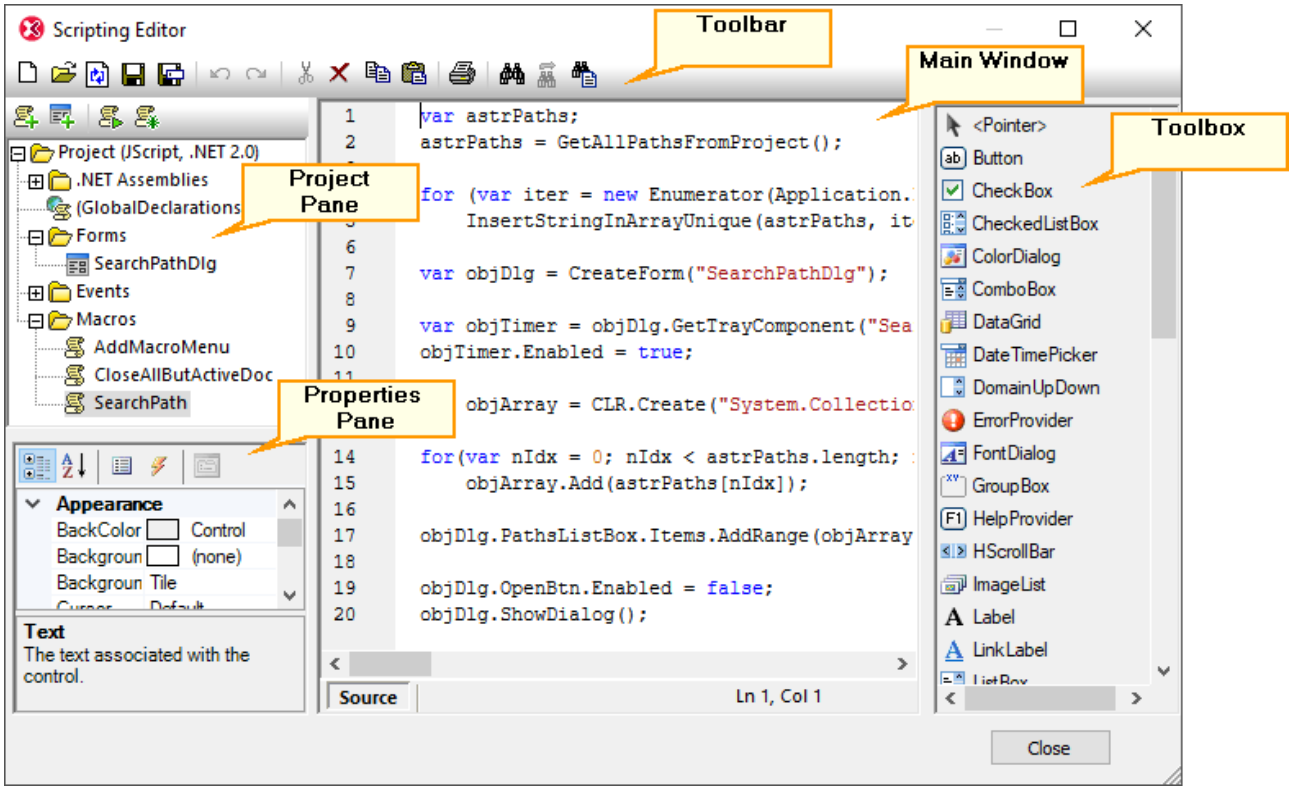
You can also find an example scripting project at the following path: **C:\Users\<user>\Documents\Altova\XMLSpy2021\Examples\SampleScripts.asprj**.

The next sections focus on the parts that your scripting project may need: global declarations, macros, forms, and events.

29.1.1.1 Overview of the Environment

The Scripting Editor consists of the following parts:

- Toolbar
- Project pane
- Properties pane
- Main window
- Toolbox



Toolbar

The toolbar includes standard Windows file management commands (**New**, **Open**, **Save**, **Save As**) commands and editor commands (**Copy**, **Cut**, **Delete**, **Paste**). When editing source code, the **Find** and **Replace** commands are additionally available, as well as the **Print** command.

Project pane

The project pane helps you view and manage the structure of the project. A scripting project consists of several components that can work together and may be created in any order:

- A *"Global Declarations"* script. As the name suggests, this script stores information available globally across the project. You can declare in this script any variables or functions that you need to be available in all forms, event handler scripts, and macros.
- *Forms.* Forms are typically necessary to collect user input, or provide some informative dialog boxes. A form is invoked by a call to it either within a function (in the Global Declarations script) or directly in a macro.
- *Events*. The "Events" folder displays XMLSpy application events provided by the COM API. To write a script that will be executed when an event occurs, double-click any event, and then type the handling code in the editor. The application events should not be confused with form events; the latter are handled at form level, as further detailed below.
- *Macros*. A macro is a script that can be invoked either on demand from a context menu or be executed automatically when XMLSpy starts. Macros do not have parameters or return values. A macro can access all variables and functions declared in the Global Declarations script and it can also display forms.

Right-click any of the components to see the available context menu commands and their shortcuts. Double-click any file (such as a form or a script) to open it in the main window.

The toolbar buttons provide the following quick commands:

	New macro	Adds a new macro to the project, in the **Macros** directory.
	New form	Adds a new form to the project, in the **Forms** directory.
	Run macro	Runs the selected macro.
	Debug macro	Runs the selected macro in debug mode.

Properties pane

The Properties pane is very similar to the one in Visual Studio. It displays the following:

- Form properties, when a form is selected
- Object properties, when an object in a form is selected
- Form events, when a form is selected
- Object events, when an object in a form is selected

To switch between the properties and events of the selected component, click the **Properties** or **Events** buttons, respectively.

The **Categorized** and **Alphabetical** icons display the properties or events either organized by category or organized in ascending alphabetical order.

When a property or event is selected, a short description of it is displayed at the bottom of the Properties pane.

(C) 2015-2021 Altova GmbH

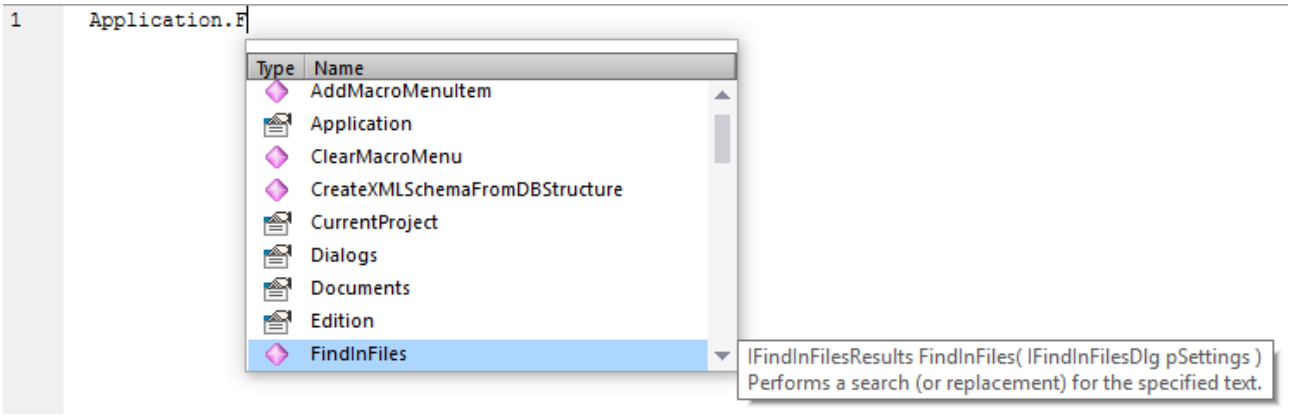
Main window

The main window is the working area where you can enter source code or modify the design of the form. When editing forms, you can work in two tabs: the **Design** tab and the **Source** tab. The **Design** tab shows the layout of the form, while the **Source** tab contains the source code such as handler methods for the form events.

The source code editor provides code editing aids such as syntax coloring, source code folding, highlighting of starting and ending braces, zooming, autocompletion suggestions, bookmarks.

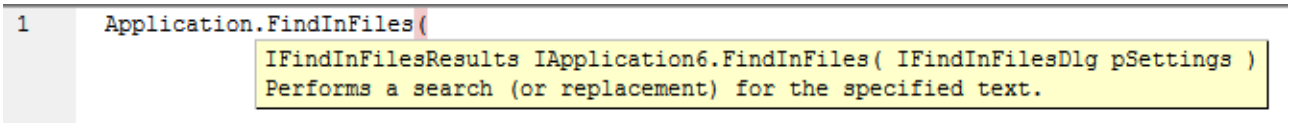
Autocompletion suggestions

JScript and VBScript are untyped languages, so autocompletion is limited to COM API names and XMLSpy built-in commands[1518]. The full method or property signature is shown next to the autocompletion entry helper.



If names start with `objDocument`, `objProject`, `objXMLData`, or `objAuthenticRange`, members of the corresponding interface will be shown.

Placing the mouse over a known method or property displays its signature (and documentation if available), for example:



The auto-completion entry helper is normally shown automatically during editing, but it can also be obtained on demand by pressing **Ctrl+Space**.

Bookmarks

- To set or remove a bookmark, click inside a line, and then press **Ctrl+F2**
- To navigate to the next bookmark, press **F2**
- To navigate to the previous bookmark, press **Shift+F2**
- To delete all bookmarks, press **Ctrl+Shift+F2**

Zooming in/out

- To zoom in or out, hold the **Ctrl** key pressed and then press the "+" or "-" keys or rotate the mouse wheel.

Text view settings
To trigger text settings, right-click inside the editor, and select **Text View Settings** from the context menu.

Font settings
To change the font, right-click inside the editor, and select **Text View Font** from the context menu.

Toolbox

The Toolbox contains all the objects that are available for designing forms, such as buttons, text boxes, combo boxes, and so on.

To add a Toolbox item to a form:

1. Create or open a form and make sure that the **Design** tab is selected.
2. Click the Toolbox object (for example, **Button**), and then click at the location in the form where you wish to insert it. Alternatively, drag the object directly onto the form.

Some objects such as `Timer` are not added to the Form but are created in a tray at the bottom of the main window. You can select the object in the tray and set properties and event handlers for the object from the Properties pane. For an example of handling tray components from the code, see Handling form events 1512.

You can also add registered ActiveX controls to the form. To do this, right-click the Toolbox area and select **Add ActiveX Control** from the context menu.

29.1.1.2 Global Declarations

The "Global Declarations" script is present by default in any scripting project; you do not need to create it explicitly. Any variables or functions that you add to this script are considered global across the entire project. Consequently, you can refer to such variables and functions from any of the project's macros and events. The following is an example of a global declarations script that imports the `System.Windows.Forms` namespace into the project. To achieve that, the code below invokes the `CLR.Import` command built into Scripting Editor.

```
// import System.Windows.Forms namespace for all macros, forms and events:
CLR.Import( "System.Windows.Forms" );
```

Note: Every time a macro is executed or an event handler is called, the global declarations are re-initialized.

29.1.1.3 Macros

Macros are scripts that contain JScript (or VBScript, depending on your project's language) statements, such as variable declarations and functions.

If your projects should use macros, you can add them as follows: right-click inside the Project pane, select **Add Macro** from the context menu, and then enter the macro's code in the main form. The code of a macro could be as simple as an alert, for example:

```
alert("Hello, I'm a macro!");
```


(C) 2015-2021 Altova GmbH

More advanced macros can contain variables and local functions. Macros can also contain code that invokes forms from the project. The listing below illustrates an example of a macro that shows a form. It is assumed that this form has already been created in the "Forms" folder and has the name "SampleForm", see also Forms 1511.

```
// display a form
ShowForm( "SampleForm" );
```

In the code listing above, `ShowForm` is a command built into Scripting Editor. For reference to other similar commands that you can use to work with forms and .NET objects, the Built-in Commands 1518.

You can add multiple macros to the same project, and you can designate any macro as "auto-macro". When a macro is designated as "auto-macro", it runs automatically when XMLSpy starts. To designate a macro as auto-macro, right-click it, and select **Set as Auto-Macro** from the context menu.

Only one macro can be run at a time. After a macro (or event) is executed, the script is closed and global variables lose their values.

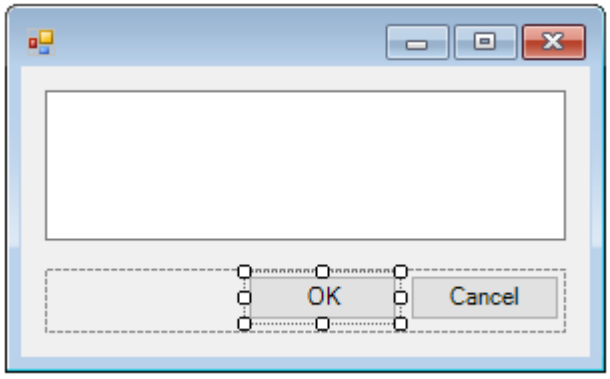
To run a macro directly in Script Editor, click **Run Macro**. To debug a macro using the Visual Studio debugger, click **Debug Macro**. For information about enabling and running macros in XMLSpy, see Enabling Scripts and Macros 1528.

29.1.1.4 Forms

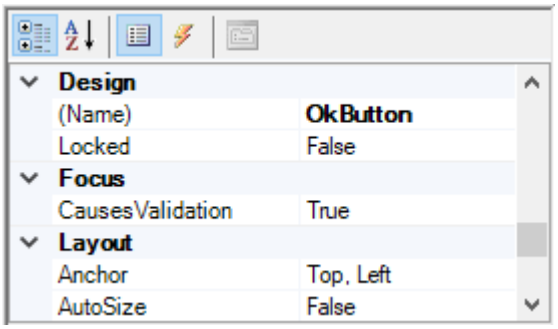
Forms are particularly useful if you need to collect input data from users or display data to users. A form can contain miscellaneous controls to facilitate this, such as buttons, check boxes, combo boxes, and so on.

To add a form, right-click inside the Project pane, and then select **Add Form** from the context menu. To add a control to a form, drag it from the Toolbox available to the right side of Scripting Editor and drop it onto the form.

You can change the position and size of the controls directly on the form, by using the handles that appear when you click any control, for example:



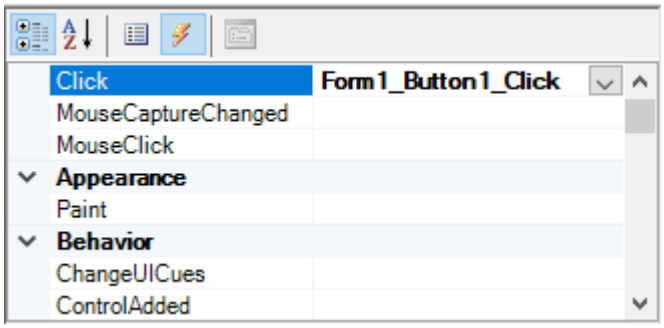
All form controls have properties that you can easily adjust in the Properties pane. To do this, first select the control on the form, and then edit the required properties in the Properties pane.



Handling form events

Each form control also exposes various events to which your scripting project can bind. For example, you might want to invoke some XMLSpy COM API method whenever a button is clicked. To create a function that binds to a form event, do the following:

1. In the Properties pane, click **Events** .
2. In the **Action** column, double-click the event where you need the method (for example, in the image below, the handled event is "Click").



You can also add handler methods by double-clicking a control on the form. For example, double-clicking a button in the form design generates a handler method for the "Click" event of that button.

Once the body of the handler method is generated, you can type code that handles this event, for example:

```
//Occurs when the component is clicked.
function MyForm_ButtonClick( objSender, e_EventArgs )
{
   alert("A button was clicked");
}
```

To display a work-in-progress form detached from the Scripting Editor, right-click the form, and select **Test Form** from the context menu. Note that the **Test Form** command just displays the form; the form's events (such as button clicks) are still disabled. To have the form react to events, call it from a macro, for example:

```
// Instantiate and display a form
ShowForm( "SampleForm" );
```


(C) 2015-2021 Altova GmbH

Accessing form controls

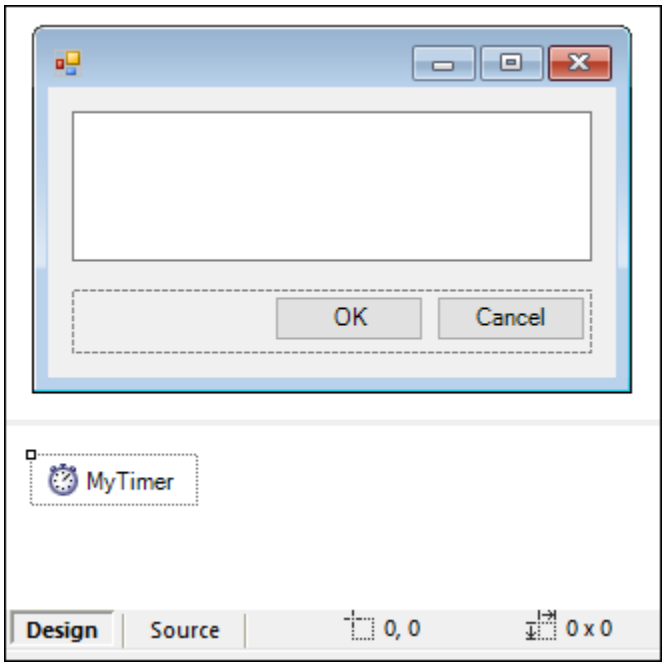
You can access any components on a form from your code by using field access syntax. For example, suppose there is a form designed as follows:

```
// MyForm
//   ButtonPanel
//      OkButton
//      CancelButton
//   TextEditor
//      AxMediaPlayer1
// TrayComponents
//   MyTimer
```

The code below shows how to instantiate the form, access some of its controls using field access syntax, and then display the form:

```
// Instantiate the form
var objForm = CreateForm("MyForm");
// Disable the OK button
objForm.ButtonPanel.OkButton.Enabled = false;
// Change the text of TextEditor
objForm.TextEditor.Text = "Hello";
// Show the form
objForm.ShowDialog();
```

When you add certain controls such as timers to the form, they are not displayed on the form; instead, they are shown as tray components at the base of the form design, for example:



To access controls from the tray, use the `GetTrayComponent` method on the form object, and supply the name of the control as argument. In this example, to get a reference to `MyTimer` and enable it, use the following code:

```
var objTimer = objForm.GetTrayComponent("MyTimer");
objTimer.Enabled = true;
```

For ActiveX Controls, you can access the underlying COM object via the OCX property:

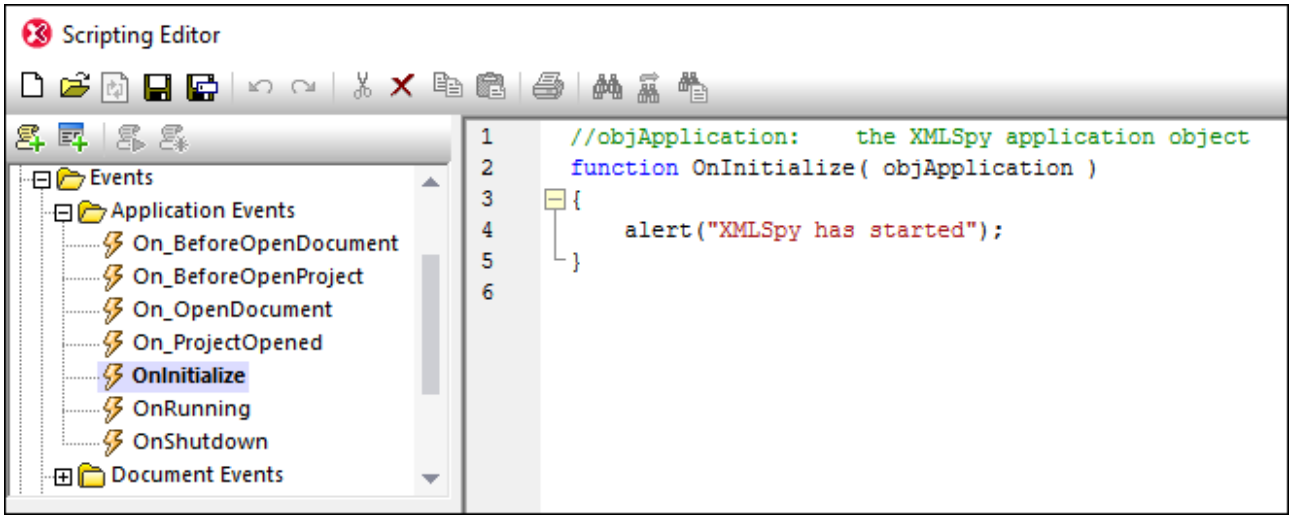
```
var ocx = lastform.AxMediaPlayer1.OCX; // get underlying COM object
ocx.enableContextMenu = true;
ocx.URL = "mms://apasf.apa.at/fm4_live_worldwide";
```

29.1.1.5 Events

Your scripting project may optionally include scripts that handle XMLSpy events such as opening, closing, or saving a document, starting or closing XMLSpy, adding an element to a diagram, and others. These events are provided by the XMLSpy COM API, and you can find them in the "Events" folder of your scripting project. Note that these events are XMLSpy-specific, as opposed to form events. Events are organized into folders as follows:

- Application Events
- Document Events
- AuthenticView Events
- GridView Events
- TextView Evets

To create an event handler script, right-click an event, and select **Open** from the context menu (or double-click the event). The event handler script is displayed in the main window, where you can start editing it. For example, the event handler illustrated below displays an alert each time XMLSpy starts:



Note the following:

- The `alert` command is applicable to JScript. The VBScript equivalent is `MsgBox`. See also alert[1519].
- The name of the event handler function must not be changed; otherwise, the event handler script will not be called.
- In order for events to be processed, the **Process Events** check box must be selected when you enable the scripting project in XMLSpy. For more information, see Enabling Scripts and Macros[1528].

You can optionally define local variables and helper functions within event handler scripts, for example:

```
var local;

function OnInitialize( objApplication )
{
   local = "OnInitialize";
   Helper();
}

function Helper()
{
   alert("I'm a helper function for " + local);
}
```

29.1.1.6 JScript Programming Tips

Below are a few JScript programming tips that you may find useful while developing a scripting project in XMLSpy Scripting Editor.

Out parameters

Out parameters from methods of the.NET Framework require special variables in JScript. For example:

```
var dictionary =
CLR.Create("System.Collections.Generic.Dictionary<System.String,System.String>");
dictionary.Add("1", "A");
dictionary.Add("2", "B");

// use JScript method to access out-parameters
var strOut = new Array(1);
if ( dictionary.TryGetValue("1", strOut) ) // TryGetValue will set the out parameter
  alert( strOut[0] ); // use out parameter
```

Integer arguments

.NET Methods that require integer arguments should not be called directly with JScript number objects which are floating point values. For example, instead of:

```
var objCustomColor = CLR.Static("System.Drawing.Color").FromArgb(128,128,128);
```

use:

```
var objCustomColor =
CLR.Static("System.Drawing.Color").FromArgb(Math.floor(128),Math.floor(128),Math.floor(
128));
```

Iterating .NET collections

To iterate .NET collections, the JScript Enumerator as well as the .NET iterator technologies can be used, for example:

```
// iterate using the JScript iterator
var itr = new Enumerator( coll );
for ( ; !itr.atEnd(); itr.moveNext() )
  alert( itr.item() );

// iterate using the .NET iterator
var itrNET = coll.GetEnumerator();
while( itrNET.MoveNext() )
  alert( itrNET.Current );
```

.NET templates

.NET templates can be instantiated as shown below:

```
var coll = CLR.Create( "System.Collections.Generic.List<System.String>" );
```

or

```
CLR.Import( "System" );
CLR.Import( "System.Collections.Generic" );
var dictionary = CLR.Create( "Dictionary<String,Dictionary<String,String>>" );
```

.NET enumeration values

.NET enumeration values are accessed as shown below:

```
var enumValStretch = CLR.Static( "System.Windows.Forms.ImageLayout" ).Stretch;
```

Enumeration literals

The enumeration literals from the XMLSpy API can be accessed as shown below (there is no need to know their numerical value).

```
objExportXMIFileDlg.XMIType = eXMI21ForUML23;
```

29.1.1.7 Example Scripting Project

A demo project that illustrates scripting with XMLSpy is available at the following path: **C:\Users\<user>\Documents\Altova\XMLSpy2021\Examples\SampleScripts.asprj**. This scripting project consists of a few macros and a Windows form.

(C) 2015-2021 Altova GmbH

To load the scripting project into Scripting Editor:

1. On the **Tools** menu, click **Scripting Editor**.
2. Click **Open** and browse for the **SampleScripts.asprj** file from the path above.

The project contains several macros in the "Macros" directory.

Macro	Description
AddMacroMenu	This macro adds a new menu item to XMLSpy, by invoking the `Application.AddMacroMenuItem` method of the COM API. The first argument of the `AddMacroMenuItem` method is the name of the macro to be added (in this example, "CloseAllButActiveDoc") and the second argument is the display text for the menu item. Whenever this macro is run, a new menu command called "CloseAllButActiveDoc") is added under the **Tools** menu. To clear macro menu items created previously, either restart XMLSpy or create a macro that calls the `Application.ClearMacroMenu` API method.
CloseAllButActiveDocument	When executed, the macro iterates though the currently open documents in XMLSpy and closes all of them, except for the active document.
SearchPath	This macro displays a form that lets users perform search for files within the current project. The form is available in the "Forms" directory, where you can view its design and the associated event handlers. The `GetAllPathsFromProject()` method returns all the file paths that belong to the currently opened project, as an array. The definition of this method is in the **GlobalDeclarations** script of the project. The `InsertStringInArrayUnique` method ensures that only unique paths are added to the array. Next, the form is initialized with CreateForm 1524. Finally, the array is converted to a .NET type with the help of the CLR.Create 1520 method and the form is populated with the resulting `ArrayList` collection. The **Open** button of the form has a handler that calls the `Application.Documents.OpenFile` API method to open the currently selected file.

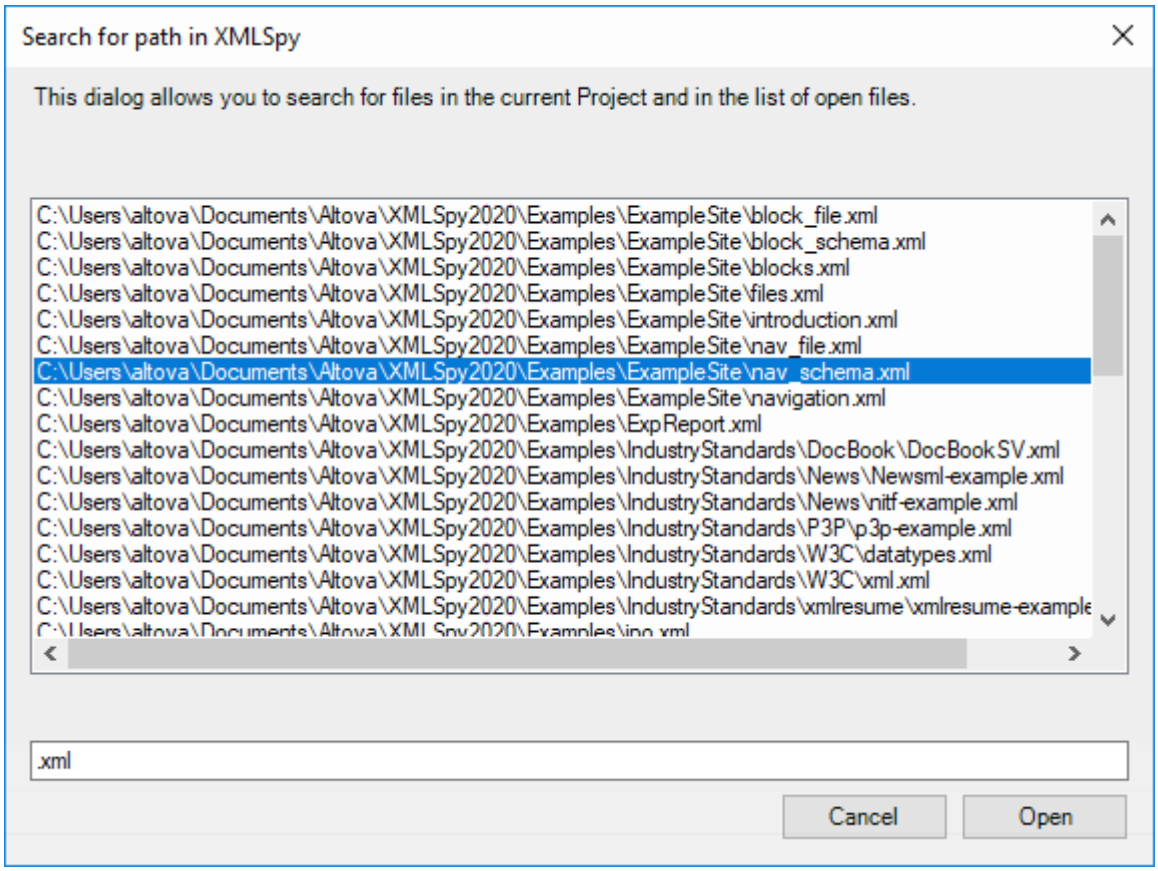
To enable the scripting project as global XMLSpy scripting project:

1. On the **Tools** menu, click **Options**.
2. Click the **Scripting** tab.
3. Under "Global scripting project file", click **Browse** and select the **SampleScripts.asprj** file from the path above.
4. This scripting project does not have auto-macros and application event handlers; therefore, you don't need to select either the **Run auto-macros...** or **Process events** check boxes.
5. Click **Apply**.

At this stage, several new menu items (one for each macro) become available under the **Tools | Macros** menu.

To run the "SearchPath" macro:

1. Open an XMLSpy project that contains several files (in this example, **C:\Users\<user>\Documents\Altova\XMLSpy2021\Examples\Examples.spp**).
2. On the **Tools** menu, click **Macros**, and then click **Search Path**.
3. Type the search term (in this example, ".xml").



As shown above, all file names that contain the search term are now listed. You can click any element in the list, and then click **Open** to display it in the main editor.

29.1.2 Built-in Commands

This section provides reference to all the commands you can use in the XMLSpy Scripting Editor.

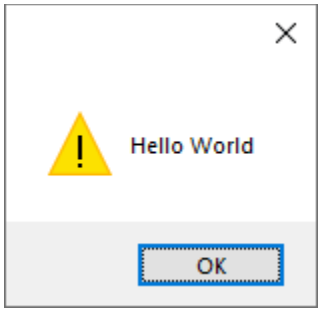
- alert 1519
- confirm 1519
- CLR.Create 1520
- CLR.Import 1521
- CLR.LoadAssembly 1521
- CLR.ShowImports 1522

(C) 2015-2021 Altova GmbH

- CLR.ShowLoadedAssemblies 1523
- CLR.Static 1523
- CreateForm 1524
- doevents 1525
- lastform 1525
- prompt 1526
- ShowForm 1526
- watchdog 1527

29.1.2.1 alert

Displays a message box that shows a given message and the "OK" button. To proceed, the user will have to click "OK".



Signature

For JScript, the signature is:

```
alert(strMessage : String) -> void
```

For VBScript, the signature is:

```
MsgBox(strMessage : String) -> void
```

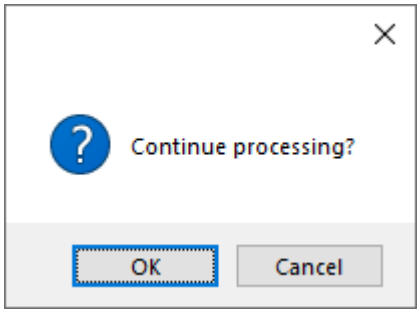
Example

The following JScript code displays a message box with the text "Hello World".

```
alert("Hello World");
```

29.1.2.2 confirm

Opens a dialog box that shows a given message, a confirmation button, and a cancel button. The user will have to click either "OK" or "Cancel" to proceed. Returns a Boolean that represents the user's answer. If the user clicked "OK", the function returns **true**; if the user clicked "Cancel", the function returns **false**.



Signature

```
confirm(strMessage : String) -> result : Boolean
```

Example (JScript)

```
if ( confirm( "Continue processing?" ) == false )
  alert("You have cancelled this action");
```

Example (VBScript)

```
If ( confirm( "Continue processing?" ) = false ) Then
  MsgBox("You have cancelled this action")
End If
```

29.1.2.3 CLR.Create

Creates a new .NET object instance of the type name supplied as argument. If more than one argument is passed, the successive arguments are interpreted as the arguments for the constructor of the .NET object. The return value is a reference to the created .NET object

Signature

```
CLR.Create(strTypeNameCLR : String, constructor arguments ... ) -> object
```

Example

The following JScript code illustrates how to create instances of various .NET classes.

```
// Create an ArrayList
var objArray = CLR.Create("System.Collections.ArrayList");
// Create a ListViewItem
var newItem = CLR.Create( "System.Windows.Forms.ListViewItem", "NewItemText" );
// Create a List<string>
var coll = CLR.Create( "System.Collections.Generic.List<System.String>" );
// Import required namespaces and create a Dictionary object
```


(C) 2015-2021 Altova GmbH

```
CLR.Import( "System" );
CLR.Import( "System.Collections.Generic" );
var dictionary = CLR.Create( "Dictionary< String, Dictionary< String, String > >" );
```

29.1.2.4 CLR.Import

Imports a namespace. This is the scripting equivalent of C# `using` and VB.Net `imports` keyword. Calling `CLR.Import` makes it possible to leave out the namespace part in subsequent calls like `CLR.Create()` and `CLR.Static()`.

Note: Importing a namespace does not add or load the corresponding assembly to the scripting project. You can add assemblies to the scripting project dynamically (at runtime) in the source code by calling CLR.LoadAssembly[1521].

Signature

```
CLR.Import(strNamespaceCLR : String) -> void
```

Example

Instead of having to use fully qualified namespaces like:

```
if ( ShowForm( "FormName" ) == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
{
    var sName = lastform.textboxFirstName.Text + " " + lastform.textboxLastName.Text;
    CLR.Static( "System.Windows.Forms.MessageBox" ).Show( "Hello " + sName );
}
```

One can import namespaces first and subsequently use the short form:

```
CLR.Import( "System.Windows.Forms" );

if ( ShowForm( "FormName" ) == CLR.Static( "DialogResult" ).OK )
{
    var sName = lastform.textboxFirstName.Text + " " + lastform.textboxLastName.Text;
    CLR.Static( "MessageBox" ).Show( "Hello " + sName );
}
```

29.1.2.5 CLR.LoadAssembly

Loads the .NET assembly with the given long assembly name or file path. Returns Boolean **true** if the assembly could be loaded; **false** otherwise.

Signature

```
CLR.LoadAssembly(strAssemblyNameCLR : String, showLoadErrors : Boolean) -> result :
Boolean
```

Example

The following JScript code attempts to set the clipboard text by loading the required assembly dynamically.

```
// set clipboard text (if possible)
// System.Windows.Clipboard is part of the PresentationCore assembly, so load this
assembly first:
if ( CLR.LoadAssembly( "PresentationCore, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35", true ) )
{
    var clipboard = CLR.Static( "System.Windows.Clipboard" );
    if ( clipboard != null )
        clipboard.SetText( "HelloClipboard" );
}
```

29.1.2.6 CLR.ShowImports

Opens a message box that shows the currently imported namespaces. The user will have to click "OK" to proceed.

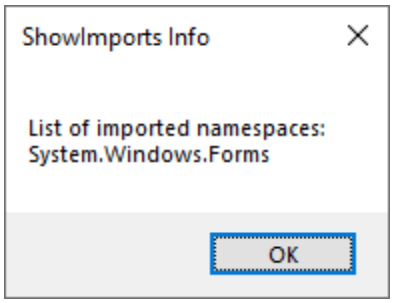
Signature

```
CLR.ShowImports() -> void
```

Example

The following JScript code first imports a namespace, and then displays the list of imported namespaces:

```
CLR.Import( "System.Windows.Forms");
CLR.ShowImports();
```



29.1.2.7 CLR.ShowLoadedAssemblies

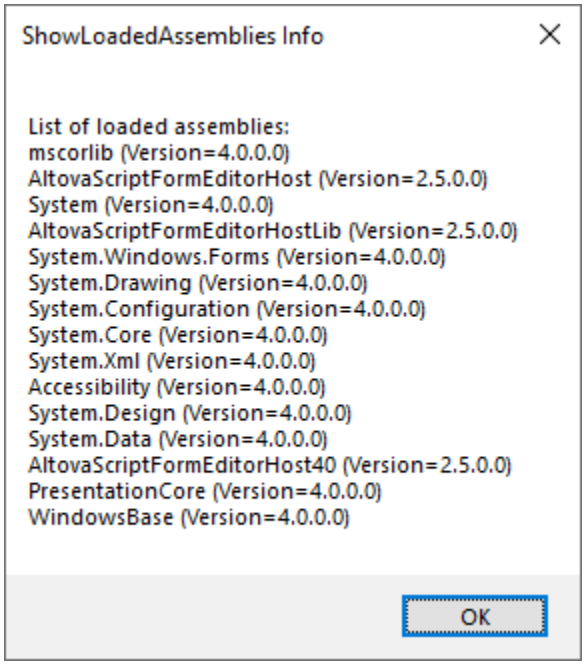
Opens a message box that shows the currently loaded assemblies. The user will have to click "OK" to proceed.

Signature

```
CLR.ShowLoadedAssemblies() -> void
```

Example

```
CLR.ShowLoadedAssemblies();
```



29.1.2.8 CLR.Static

Returns a reference to a static .NET object. You can use this function to get access to .NET types that have no instances and contain only static members.

Signature

```
CLR.Static(strTypeNameCLR : String) -> object
```

Example (JScript)

```
// Get the value of a .NET Enum into a variable
```

```
var enumValStretch = CLR.Static( "System.Windows.Forms.ImageLayout" ).Stretch

// Set the value of the Windows clipboard
var clipboard = CLR.Static( "System.Windows.Clipboard" );
clipboard.SetText( "HelloClipboard" );

// Check the buttons pressed by the user on a dialog box
if ( ShowForm( "FormName" ) == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
    alert( "ok" );
else
    alert( "cancel" );
```

29.1.2.9 CreateForm

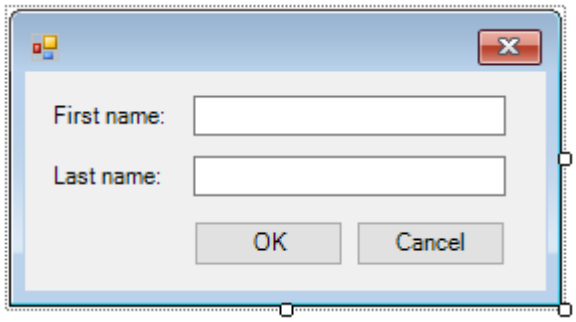
Instantiates the `Form` object identified by the name supplied as argument. The form must exist in the "Forms" folder of the scripting project. Returns the form object (`System.Windows.Forms.Form`) corresponding to the given name, or `null` if no form with such name exists.

Signature

```
CreateForm (strFormName : String) -> System.Windows.Forms.Form | null
```

Example

Let's assume that a form called "FormName" exists in the scripting project.



The following JScript code instantiates the form with some default values and displays it to the user.

```
var myForm = CreateForm( "FormName" );
if ( myForm != null )
{
   myForm.textboxFirstName.Text = "Daniela";
   myForm.textboxLastName.Text = "Heidegger";
   var dialogResult = myForm.ShowDialog();
}
```

The `dialogResult` can subsequently be evaluated as follows:

```
if ( dialogResult == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
```

```
    alert( "ok" );
else
    alert( "cancel" );
```

Note: The code above will work only if the **DialogResult** property of the "OK" and "Cancel" buttons is set correctly from the Properties pane (for example, it must be **OK** for the "OK" button).

29.1.2.10 doevents

Processes all Windows messages currently in the message queue.

Signature

```
doevents() -> void
```

Example (JScript)

```
for ( i=0; i < nLongLastingProcess; ++i )
{
    // do long lasting process

    doevents(); // process Windows messages; give UI a chance to update
}
```

29.1.2.11 lastform

This is a global field that returns a reference to the last form object that was created via `CreateForm()` or `ShowForm()`.

Signature

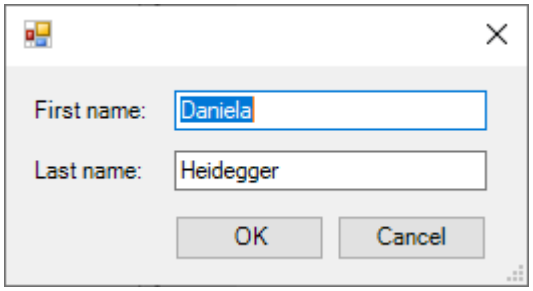
```
lastform -> formObj : System.Windows.Forms.Form
```

Example

The following JScript code shows the form "FormName" as a dialog box.

```
CreateForm( "FormName" );
if ( lastform != null )
{
    lastform.textboxFirstName.Text = "Daniela";
    lastform.textboxLastName.Text = "Heidegger";
    var dialogResult = lastform.ShowDialog();
}
```

The values of both textbox controls are initialized with the help of `lastform`.



29.1.2.12 prompt

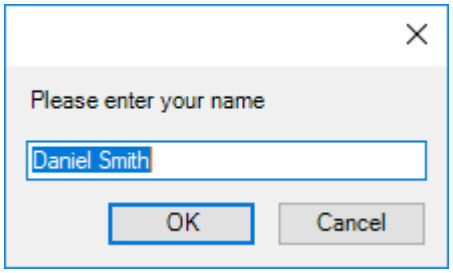
Opens a dialog box that shows a message and a textbox control with a default answer. This can be used to let the user input a simple string value. The return value is a string that contains the textbox value or null if the user selected "Cancel".

Signature

```
prompt(strMessage : String, strDefault : String) -> val : String
```

Example

```
var name = prompt( "Please enter your name", "Daniel Smith" );
if ( name != null )
    alert( "Hello " + name + "!" );
```



29.1.2.13 ShowForm

Instantiates a new form object from the given form name and immediately shows it as dialog box. The return value is an integer that represents the generated `DialogResult` (`System.Windows.Forms.DialogResult`). For the list of possible values, refer to the documentation of the `DialogResult` Enum (https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.dialogresult?view=netframework-4.8).

Signature

```
ShowForm(strFormName : String) -> result : Integer
```

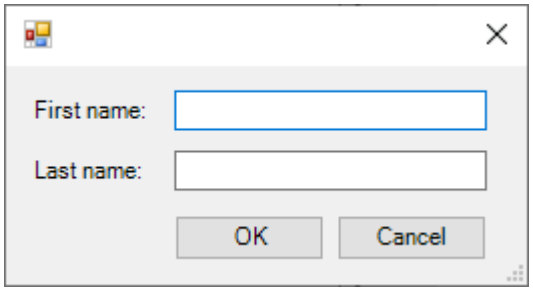

(C) 2015-2021 Altova GmbH

Example

The following JScript code

```
var dialogResult = ShowForm( "FormName" );
```

Shows the form "FormName" as a dialog box:



The `DialogResult` can subsequently be evaluated, for example:

```
if ( dialogResult == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
    alert( "ok" );
else
    alert( "cancel" );
```

Note: The code above will work only if the **DialogResult** property of the "OK" and "Cancel" buttons is set correctly from the Properties pane (for example, it must be **OK** for the "OK" button).

29.1.2.14 watchdog

Long running CPU-intensive scripts may ask the user if the script should be terminated. The `watchdog()` method is used to disable or enable this behavior. By default, the watchdog is enabled.

Calling `watchdog(true)` can also be used to reset the watchdog. This can be useful before executing long running CPU-intensive tasks to ensure they have the maximum allowed script processing quota.

Signature

```
watchdog(bEnable : boolean) -> void
```

Example

```
watchdog( false );  // disable watchdog - we know the next statement is CPU intensive
but it will terminate for sure
doCPUIntensiveScript();
watchdog( true );   // re-enable watchdog
```

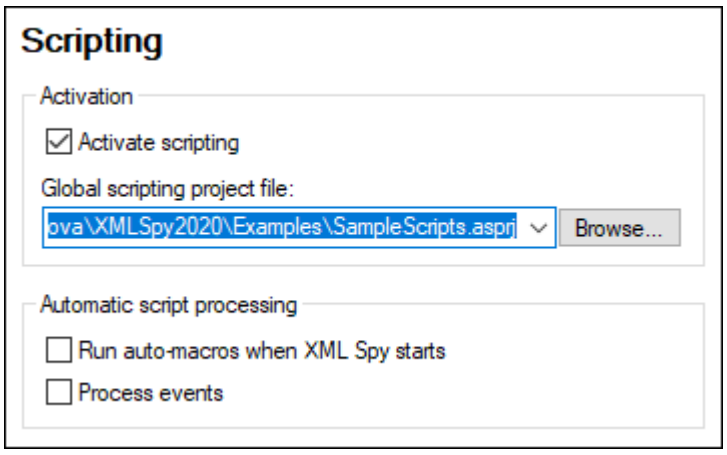
29.1.3 Enabling Scripts and Macros

Once a scripting project is complete and tested, you can use it in the following ways:

1. As the global scripting project for XMLSpy. This means that all the scripts and macros from the scripting project are available to XMLSpy.
2. At project level. This means that a reference to the .asprj file is saved together with the XMLSpy project. When the XMLSpy project is opened, its associated scripts and macros can be called.

To set a scripting project as global:

1. On the **Tools** menu, click **Options**.
2. Click the **Scripting** tab.
3. Select the **Activate scripting** check box and browse for the .asprj file to be used as global scripting project.



You can optionally enable the following additional script processing options:

Run auto-macros when XMLSpy starts	If you select this check box, any macros that were set as "Auto-macro" in the project will be triggerred automatically when XMLSpy starts.
Process events	Select this check box if your scripts bind to any application events. Clear the check box to prevent the scripts from reacting to events.

To enable a scripting project at project level:

1. Open the project.
2. On the **Project** menu, click **Script Settings**.
3. Select the **Activate project scripts** check box and browse for the .asprj file.

The **Run-auto macros...** check box has the same meaning as already described above.

(C) 2015-2021 Altova GmbH

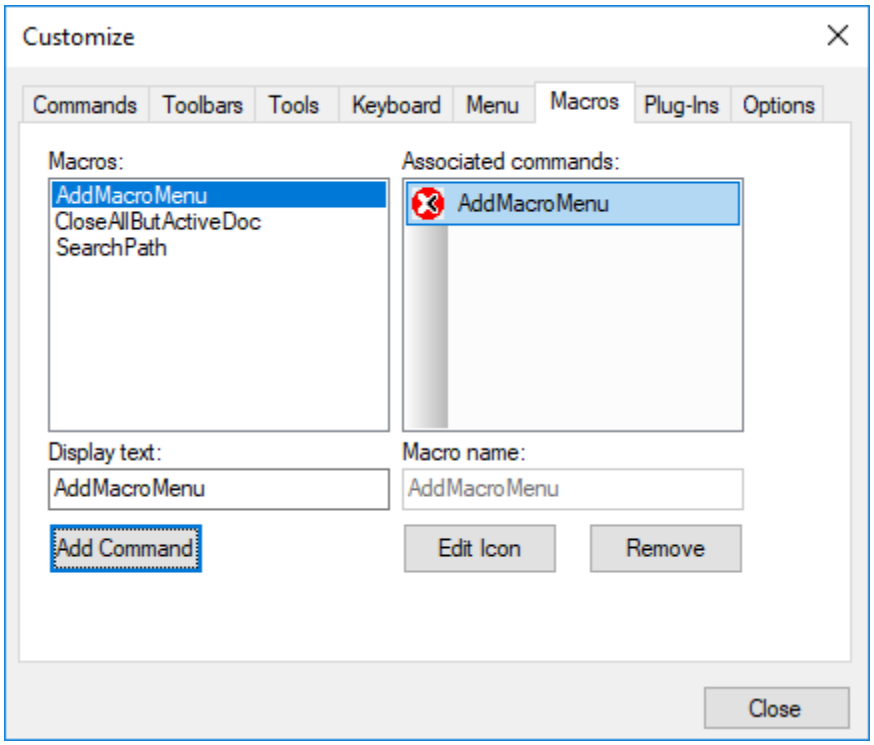
29.1.3.1 Running Macros

When a scripting project is active in XMLSpy, any macros available in that project are displayed in the **Tools | Macros** menu. Therefore, you can run a macro at any time, by triggering the respective menu command, for example **Tools | Macros | <SomeMacro>**.

Macros that were configured as auto-macros will run automatically whenever XMLSpy starts, provided that this behavior is enabled from options, as described in Enabling Scripts and Macros 1528.

For convenience, you can create toolbar buttons for macros, as follows:

1. On the **Tools** menu, click **Customize**.
2. Click the **Macros** tab. Any macros that are available at application level (in the *global* scripting project) are listed.
3. Click **Add Command**.



4. Optionally, click **Edit icon** and draw a new icon for the new macro. You can also assign a shortcut to the macro, from the **Keyboard** tab.
5. Drag the macro from the **Associated commands** pane onto the toolbar where you would like it to appear.

To remove a macro from a toolbar:

1. On the **Tools** menu, click **Customize**.
2. Click the **Macros** tab.
3. Drag the macro from the toolbar where it appears back into the **Associated commands** pane.

29.2 IDE Plugins

XMLSpy allows you to create your own IDE plug-ins and integrate them into XMLSpy.

Use plug-ins to:

- Configure your version of XMLSpy, add commands through menus, icons, buttons etc.
- React to events from XMLSpy.
- Run your specific code within XMLSpy with access to the complete XMLSpy API

XMLSpy expects your plug-in to implement the IXMLSpyPlugIn[1539] interface. C# and C++ are the currently supported languages, and examples using these languages are included with your installation package and are located in the `XMLSpy2021\Examples\IDEPlugin` folder of your XMLSpy installation.

Windows 7, 8, 10	`C:/Users/<username>/Documents`

See ATL sample files[1534] for an example using C++.

29.2.1 Registration of IDE PlugIns

XMLSpy maintains a specific key in the Registry where it stores all registered IDE plug-ins:

```
HKEY_CURRENT_USER\Software\Altova\XML Spy\PlugIns
```

All values of this key are treated as references to registered plug-ins and must conform to the following format:

Value name:	ProgID of the plug-in
Value type:	must be REG_SZ
Value data:	CLSID of the component

Each time the application starts the values of the **`PlugIns`** key is scanned, and the registered plug-ins are loaded.

Register plug-in manually

To register a plug-in manually, use the Customize dialog box of XMLSpy's **Tools** menu. Use the **Add Plug-In** button to specify the DLL that implements your plug-in. XMLSpy registers the DLL as a COM server and adds the corresponding entry in its **`PlugIns`** key.

If you experience problems with manual registration, check whether the CLSID of your plug-in is correctly registered in the **`PlugIns`** key. If the registration is incorrect, then the name of your plug-in DLL was probably not sufficiently unique. Use a different name or perform direct registration.

Register plug-in directly

A plug-in can be directly registered as an IDE plug-in by first registering the DLL and then adding the appropriate value to the **`PlugIns`** key of XMLSpy. (This can be done, for eaxmple, during plug-in setup.) The new plug-in will be activated the next time XMLSpy is launched.

(C) 2015-2021 Altova GmbH

Creating plug-ins

Source code for sample plug-ins has been provided in the application's (My) Documents folder[33]: `Examples\IDEPlugin` folder. To build a plug-in from such source code, do the following:

1. Open the solution you want to build as a plug-in in Visual Studio.
2. Build the plug-in with the command in the Build menu.
3. The plug-in's DLL file will be created in the `Bin` or `Debug` folder. This DLL file is the file that must be added as a plug-in (*see above*).

Note: C# and C++ are the currently supported languages.

29.2.2 ActiveX Controls

ActiveX controls are supported. Any IDE PlugIn which is also an ActiveX control will be displayed in a Dialog Control Bar. A sample PlugIn that is also an ActiveX control is included in the `IDEPlugin` folder in the `Examples` folder of your application folder.

29.2.3 Configuration XML

The IDE plug-in allows you to change the user interface (UI) of XMLSpy. This is done by describing each separate modification using an XML data stream. The XML configuration is passed to XMLSpy using the GetUIModifications[1542] method of the IXMLSpyPlugIn interface.

The XML file containing the UI modifications for the IDE PlugIn, must have the following structure:

```
<ConfigurationData>
   <ImageFile>Path to image file</ImageFile>
   <Modifications>
      <Modification>
         ...
      </Modification>
         ...
    </Modifications>
</ConfigurationData>
```

You can define icons or toolbar buttons for new menu items that are added to the UI of XMLSpy by the plug-in. The path to the file containing the images is set using the `ImageFile` element. Each image must be 16x16 pixels using maximum 256 colors. The image references must be arranged from left to right in a single **<ImageFile>** element. The rightmost image index value is zero.

The **Modifications** element can have any number of **Modification** child elements. Each **Modification** element defines a specific change to the standard UI of XMLSpy. The modifications you can carry out are described in the next section below.

Structure of Modification elements

A `Modification` element has two child elements:

```
<Modification>
```

```
   <Action>Type of action</Action>
   <UIElement Type="Type of UI element" />
</Modification>
```

Valid values for the **Action** element are:

Add: to add the following UI element to XMLSpy
Hide: to hide the following UI element in XMLSpy
Remove: to remove the UI element from the "Commands" list box, in the customize dialog

Multiple modifications can be combined in an **Action** element, like this: **"Add Hide"**

The **UIElement** element defines any new or existing UI element and may be one of the the following types: toolbars, buttons, menus, or menu items. The **type** attribute specifies which of these types the UI element belongs to. The structure of **UIElement** is described in the sections below.

Common UIElement children

The ID and Name elements are defined for all types of UI element. In the case of some types, however, one of these elements is ignored. For example, **Name** is ignored for a separator.

```
   <ID></ID>
   <Name></Name>
```

If **UIElement** describes an existing element of the UI, the value of the **ID** element is predefined by XMLSpy. Normally these ID values are not known to the public. If the XML fragment describes a new part of the UI, then the ID is arbitrary and the value should be less than **1000**. The **Name** element sets the textual value. Existing UI elements can be identified just by name; for example, menus and menu items that have sub menus. For new UI elements, the **Name** element sets the caption (for example, the title of a toolbar) or the text of a menu item.

Toolbars and Menus

To define a toolbar it is necessary to specify the ID and/or the name of the toolbar. An existing toolbar can be specified using only the name or ID (if the latter is known). To create a **new** toolbar, both values must be set. The **type** attribute must have a value of **ToolBar**.

```
<UIElement Type="ToolBar">
   <ID>1</ID>
   <Name>TestPlugIn</Name>
</UIElement>
```

To specify an XMLSpy menu you need two parameters:

- The ID of the menu bar which contains the menu. If no XML documents are open in the main window, the menu bar ID is 128. If one or more XML documents are open, the menu bar ID is 129.
- The menu name. Menus do not have an associated ID value. The following example defines the "Edit" menu of the menu bar which is active, when at least one XML document is open:

```
<UIElement Type="Menu">
   <ID>129</ID>
   <Name>Edit</Name>
</UIElement>
```

An additional element is used if you want to create a new menu. The **Place** element defines the position of the new menu in the menu bar:

```
<UIElement Type="Menu">
   <ID>129</ID>
   <Name>PlugIn Menu</Name>
   <Place>12</Place>
</UIElement>
```

A value of `-1` for the `Place` element sets the new button or menu item at the end of the menu or toolbar.

Commands

If you add a new command (through a toolbar button or a menu item), the **`UIElement`** fragment can contain any of these sub elements:

```
    <MacroName></MacroName>
    <Info></Info>
    <ImageID></ImageID>
```

If **`MacroName`** is specified, XMLSpy searches for a macro with the same name in the scripting environment and executes it each time this command is processed. The **`Info`** element contains a description string that is displayed in the status bar when the mouse pointer is over the associated command (button or menu item). **`ImageID`** defines the index of the icon in the image file. Note that all icons are stored in one image file.

To define a toolbar button, create an `UIElement` with this structure:

```
<UIElement Type="ToolBarItem">
   <!--don't reuse local IDs even the commands do the same-->
   <ID>5</ID>
   <Name>Open file from repository...</Name>
   <!--Set Place To -1 If this is the first button To be inserted-->
   <Place>-1</Place>
   <ImageID>0</ImageID>
   <ToolBarID>1</ToolBarID>
   <!--instead of the toolbar ID the toolbar name could be used-->
   <ToolBarName>TestPlugIn</ToolBarName>
</UIElement>
```

Additional elements to declare a toolbar button are **`Place`**, **`ToolBarID`** and **`ToolBarName`**. The `ToolBarID` and `ToolBarName` elements are used to identify the toolbar which contains the new or existing button. The textual value of **`ToolBarName`** is case-sensitive. The (UIElement) **`type`** attribute must be **`ToolBarItem`**.

To define a menu item, the elements **`MenuID`**, **`Place`** and **`Parent`** are available in addition to the standard elements used to declare a command. The content of the **`MenuID`** element can be either **`128`** or **`129`**. *See the section "Toolbars and Menus" above for more information.*

The **`Parent`** element is used to identify the menu where the new menu entry should be inserted. As sub menu items have no unique Windows ID, we need some other way to identify the parent of the menu item. We do this by setting the content of the **`Parent`** element to be the path to the menu item. The steps in the path are indicated by a colon. The pattern would be **`ParentMenu:SubMenu`**. If the menu has no parent (because it is not a submenu), add a colon to the beginning of the name (see example below). The **`type`** attribute must be set to **`MenuItem`**.

The example below defines a menu item, where the containing menu is not a sub menu:

```
<UIElement Type="MenuItem">
   <!--the following element is a Local command ID-->
   <ID>3</ID>
   <Name>Open file from repository...</Name>
```

```
   <Place>-1</Place>
   <MenuID>129</MenuID>
   <Parent>:PlugIn Menu</Parent>
   <ImageID>0</ImageID>
</UIElement>
```

You can add toolbar separators and menus if the value of the `ID` element is set to `0`.

29.2.4 ATL Sample Files

This section shows how to create a simple XMLSpy IDE plug-in DLL using ATL. You must know how to work with MS VisualStudio, ATL, and the wizards that generate new ATL objects. To access the API, the implementation imports the Type Library of XMLSpy. The code reads various properties and calls methods using the smart pointers provided by the `#import` statement of the code. In addition, the sample code uses the MFC class `CString` and ATL conversion macros such as W2T.

The broad steps to create an ATL DLL are as follows:

1. Open VisualStudio and select **File | New**.
2. Select the *Projects* tab.
3. Select *ATL COM AppWizard*, and type in a project name.
4. Select *Support for MFC* if you want to use MFC classes or if you want to create a project for the sample code.

Having created the project files you can add an ATL object to implement the `IXMLSpyPlugIn` interface:

1. Select **Insert | New ATL Object**.
2. Select *Simple Object* from the wizard. and click **Next**.
3. Type in a name for the object.
4. On the *Attributes* tab, select *Custom* for the type of interface and disable *Aggregation*.

These steps produce the skeleton code for the implementation of the IDE plug-in interface. See the following pages for information about how to modify the code and specify some basic functionality.

29.2.4.1 Interface description (IDL)

The IDL of the newly created ATL object contains a declaration for one COM interface.

- This interface declaration must be replaced by the declaration of `IXMLSpyPlugIn` as shown below.
- The IDL must also contain the definition of the `SPYUpdateAction` enumeration.
- Replace the generated default interface name (created by the wizard) with `IXMLSpyPlugIn` in the `coclass` declaration.

The IDL should then look something like the example code below. After creating the ATL object, you need to implement the IDE plug-in interface of XMLSpy.

```
import "oaidl.idl";
import "ocidl.idl";
```

```
// ----- please insert the following block into your IDL file -----
    typedef enum {
       spyEnable = 1,
       spyDisable = 2,
       spyCheck = 4,
       spyUncheck = 8
    } SPYUpdateAction;

// ----- end insert block ----


// ----- E.g. Interface entry automatically generated by the ATL wizard -----
//     [
//           object,
//           uuid(AB7CD86A-8145-429A-A1F3-270692EO8AFC),

//           helpstring("IXMLSpyPlugIn Interface")
//           pointer_default(unique)
//     ]
//     interface IXMLSpyPlugIn : IUnknown
//     {
//     };
// ----- end automatically generated Interface Entry


// ----- replace the Interface Entry (shown above) generated for you by the ATL wizard,
with the following block -----

       [
          odl,
          uuid(88F2A622-4B7E-42CD-8D04-3C0E5389DD85),
          helpstring("IXMLSpyPlugIn Interface")
       ]
       interface IXMLSpyPlugIn : IUnknown
          {
             HRESULT _stdcall OnCommand([in] long nID, [in] IDispatch* pXMLSpy);
             HRESULT _stdcall OnUpdateCommand([in] long nID, [in] IDispatch* pXMLSpy,
[out, retval] SPYUpdateAction* pAction);
             HRESULT _stdcall OnEvent([in] long nEventID, [in] SAFEARRAY(VARIANT)*
arrayParameters, [in] IDispatch* pXMLSpy, [out, retval] VARIANT* pReturnValue);
             HRESULT _stdcall GetUIModifications([out, retval] BSTR* pModificationsXML);
             HRESULT _stdcall GetDescription([out, retval] BSTR* pDescription);
          };

// ----- end replace block -----


// ----- The code below is automatically generated by the ATL wizard and will look
slightly different in your case -----

       [
          uuid(24FE0D1B-3FC0-494E-B36E-1D4CE412B014),
          version(1.0),
          helpstring("XMLSpyIDEPlugInDLL 1.0 Type Library")
       ]
       library XMLSPYIDEPLUGINDLLLib
          {
       importlib("stdole32.tlb");
       importlib("stdole2.tlb");

       [
          uuid(3800E791-7F6B-4ACD-9E32-2AC184444501),
```

```
         helpstring("XMLSpyIDEPlugIn Class")
      ]
      coclass XMLSpyIDEPlugIn
      {
         [default] interface IXMLSpyPlugIn;   // ----- define IXMLSpyPlugIn as the
default interface -----
      };
};
```

29.2.4.2 Class definition

In the class definition of the ATL object, the following changes must be made:

- The class has to derive from `IXMLSpyPlugIn`
- The "Interface Map" needs an entry for `IXMLSpyPlugIn`
- The methods of the IDE plug-in interface must be declared

These changes can be made as shown below:

```
#ifndef __XMLSPYIDEPLUGIN_H_
#define __XMLSPYIDEPLUGIN_H_

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CXMLSpyIDEPlugIn
class ATL_NO_VTABLE CXMLSpyIDEPlugIn :
   public CComObjectRootEx<CComSingleThreadModel>,
   public CComCoClass<CXMLSpyIDEPlugIn, &CLSID_XMLSpyIDEPlugIn>,
   public IXMLSpyPlugIn
{
public:
   CXMLSpyIDEPlugIn()
   {
   }

DECLARE_REGISTRY_RESOURCEID(IDR_XMLSPYIDEPLUGIN)
DECLARE_NOT_AGGREGATABLE(CXMLSpyIDEPlugIn)

DECLARE_PROTECT_FINAL_CONSTRUCT()

BEGIN_COM_MAP(CXMLSpyIDEPlugIn)
   COM_INTERFACE_ENTRY(IXMLSpyPlugIn)
END_COM_MAP()

// IXMLSpyIDEPlugIn
public:
   virtual HRESULT _stdcall OnCommand(long nID, IDispatch* pXMLSpy);
   virtual HRESULT _stdcall OnUpdateCommand(long nID, IDispatch* pXMLSpy,
SPYUpdateAction* pAction);
   virtual HRESULT _stdcall OnEvent(long nEventID, SAFEARRAY **arrayParameters,
IDispatch* pXMLSpy, VARIANT* pReturnValue);
   virtual HRESULT _stdcall GetUIModifications(BSTR* pModificationsXML);
   virtual HRESULT _stdcall GetDescription(BSTR* pDescription);
};
```


(C) 2015-2021 Altova GmbH

```
#endif //__XMLSPYIDEPLUGIN_H_
```

29.2.4.3 Implementation

The code below shows a simple implementation of an XMLSpy IDE plug-in. It adds a menu item and a separator (available with XMLSpy) to the Tools menu. Inside the OnUpdateCommand() method, the new command is only enabled when the active document is displayed using the Grid View. The command searches for the XML element which has the current focus, and opens any URL starting with "http://", from the textual value of the element.

```
////////////////////////////////////////////////////////////////////////
// CXMLSpyIDEPlugIn

#import "XMLSpy.tlb"
using namespace XMLSpyLib;

HRESULT CXMLSpyIDEPlugIn::OnCommand(long nID, IDispatch* pXMLSpy)
{
   USES_CONVERSION;

   if(nID == 1)     {
      IApplicationPtr      ipSpyApp;

      if(pXMLSpy)    {
         if(SUCCEEDED(pXMLSpy->QueryInterface(__uuidof(IApplication),(void **)
&ipSpyApp))) {
            IDocumentPtr  ipDocPtr = ipSpyApp->ActiveDocument;

            // we assume that grid view is active
            if(ipDocPtr) {
               IGridViewPtr         ipGridPtr = ipDocPtr->GridView;

               if(ipGridPtr)      {
                  IXMLDataPtr     ipXMLData = ipGridPtr->CurrentFocus;

                  CString  strValue = W2T(ipXMLData->TextValue);

                  if(!strValue.IsEmpty() && (strValue.Left(7) == _T("http://")))
                     ::ShellExecute(NULL,_T("open"),W2T(ipXMLData-
>TextValue),NULL,NULL,SW_SHOWNORMAL);
               }
            }
         }
      }
   }

   return S_OK;
}


HRESULT CXMLSpyIDEPlugIn::OnUpdateCommand(long nID, IDispatch* pXMLSpy, SPYUpdateAction*
pAction)
{
   *pAction = spyDisable;

   if(nID == 1)     {
      IApplicationPtr      ipSpyApp;
```

```
      if(pXMLSpy)   {
         if(SUCCEEDED(pXMLSpy->QueryInterface(__uuidof(IApplication),(void **)
&ipSpyApp)))  {
            IDocumentPtr   ipDocPtr = ipSpyApp->ActiveDocument;

            // only enable if grid view is active
            if((ipDocPtr != NULL) && (ipDocPtr->CurrentViewMode == spyViewGrid))
               *pAction = spyEnable;
         }
      }
   }

   return S_OK;
}


HRESULT CXMLSpyIDEPlugIn::OnEvent(long nEventID, SAFEARRAY **arrayParameters, IDispatch*
pXMLSpy, VARIANT* pReturnValue)
{
   return S_OK;
}

HRESULT CXMLSpyIDEPlugIn::GetUIModifications(BSTR* pModificationsXML)
{
   CComBSTR  bstrMods = _T(" \
            <ConfigurationData>   \
               <Modifications>    ");
   // add "Open URL..." to Tools menu
   bstrMods.Append (_T(" \
                  <Modification> \
                     <Action>Add</Action> \
                     <UIElement type=\"MenuItem\"> \
                        <ID>1</ID> \
                        <Name>Open URL...</Name> \
                        <Place>0</Place> \
                        <MenuID>129</MenuID> \
                        <Parent>:Tools</Parent> \
                     </UIElement> \
                  </Modification>"));
   // add Seperator to Tools menu
   bstrMods.Append (_T(" \
                  <Modification> \
                     <Action>Add</Action> \
                     <UIElement type=\"MenuItem\"> \
                        <ID>0</ID> \
                        <Place>1</Place> \
                        <MenuID>129</MenuID> \
                        <Parent>:Tools</Parent> \
                     </UIElement> \
                  </Modification>"));
   // finish modification description
   bstrMods.Append (_T(" \
               </Modifications> \
            </ConfigurationData>"));

   return bstrMods.CopyTo(pModificationsXML);
}


HRESULT CXMLSpyIDEPlugIn::GetDescription(BSTR* pDescription)
{
```

```
   CComBSTR  bstrDescr = _T("ATL C++ XMLSpy IDE PlugIn;This PlugIn demonstrates the
implementation of a simple ATL DLL as a IDE PlugIn for XMLSpy.");
   return bstrDescr.CopyTo(pDescription);
}
```

29.2.5 IXMLSpyPlugIn

Methods

OnCommand 1539
OnUpdateCommand 1540
OnEvent 1541
GetUIModifications 1542
GetDescription 1543

Description

If a DLL is added to XMLSpy as an IDE plug-in, it is necessary that it registers a COM component that answers to an `IXMLSpyPlugIn` interface with the reserved `uuid(88F2A622-4B7E-42CD-8D04-3C0E5389DD85)`. This is required for it to be recognized as a plug-in.

29.2.5.1 OnCommand

Declaration

```
OnCommand(nID as long, pXMLSpy as IDispatch)
```

Description

The `OnCommand()` method of the interface implementation is called each time a command added by the the IDE plug-in (menu item or toolbar button) is processed. `nID` stores the command ID defined by the `ID` element of the respective `UIElement`. `pXMLSpy` holds a reference to the dispatch interface of the `Application` object of XMLSpy.

Example

```
Public Sub IXMLSpyPlugIn_OnCommand(ByVal nID As Long, ByVal pXMLSpy As Object)
   If (Not (pXMLSpy Is Nothing)) Then
      Dim objDlg
      Dim objDoc As XMLSpyLib.Document
      Dim objSpy As XMLSpyLib.Application
      Set objSpy = pXMLSpy

      If nID = 3 Or nID = 5 Then
         Set objDlg = CreateObject("MSComDlg.CommonDialog")
         objDlg.Filter = "XML Files (*.xml)|*.xml|All Files (*.*)|*.*||"
         objDlg.FilterIndex = 1
         objDlg.ShowOpen

         If Len(objDlg.FileName) > 0 Then
            Set objDoc = objSpy.Documents.OpenFile(objDlg.FileName, False)
            Set objDoc = Nothing
         End If
      End If

      If nID = 4 Or nID = 6 Then
         Set objDlg = CreateObject("MSComDlg.CommonDialog")
         objDlg.Filter = "All Files (*.*)|*.*||"
```

```
         objDlg.Flags = cdlOFNPathMustExist
         objDlg.ShowSave

         If Len(objDlg.FileName) > 0 Then
            Set objDoc = objSpy.ActiveDocument

            If Not (objDoc Is Nothing) Then
               objDoc.SetPathName objDlg.FileName
               objDoc.Save
               Set objDoc = Nothing
            End If
         End If
      End If

      Set objSpy = Nothing
   End If
End Sub
```

29.2.5.2 OnUpdateCommand

Declaration

```
OnUpdateCommand(nID as long, pXMLSpy as IDispatch) as SPYUpdateAction
```

Description

The `OnUpdateCommand()` method is called each time the visible state of a button or menu item needs to be set. `nID` stores the command ID defined by the `ID` element of the respective `UIElement`. `pXMLSpy` holds a reference to the dispatch interface of the `Application` object.

Possible return values to set the update state are:

```
spyEnable    = 1
spyDisable   = 2
spyCheck     = 4
spyUncheck   = 8
```

Example

```
Public Function IXMLSpyPlugIn_OnUpdateCommand(ByVal nID As Long, ByVal pXMLSpy As Object)
As SPYUpdateAction
   IXMLSpyPlugIn_OnUpdateCommand = spyDisable

   If (Not (pXMLSpy Is Nothing)) Then
      Dim objSpy As XMLSpyLib.Application
      Set objSpy = pXMLSpy

      If nID = 3 Or nID = 5 Then
         IXMLSpyPlugIn_OnUpdateCommand = spyEnable
      End If
      If nID = 4 Or nID = 6 Then
         If objSpy.Documents.Count > 0 Then
            IXMLSpyPlugIn_OnUpdateCommand = spyEnable
         Else
            IXMLSpyPlugIn_OnUpdateCommand = spyDisable
         End If
      End If
   End If
End Function
```


(C) 2015-2021 Altova GmbH

29.2.5.3 OnEvent

Declaration

```
OnEvent(nEventID as long, arrayParameters as SAFEARRAY(VARIANT), pXMLSpy as IDispatch) as
VARIANT
```

Description

`OnEvent()` is called each time an event is raised from XMLSpy.

Possible values for `nEventID` are:

```
On_BeforeStartEditing              = 1
On_EditingFinished                 = 2
On_FocusChanged                    = 3
On_Beforedrag                      = 4
On_BeforeDrop                      = 5
On_OpenProject                     = 6
On_OpenDocument                    = 7
On_CloseDocument                   = 8
On_SaveDocument                    = 9
On_DocEditDragOver                 = 10
On_DocEditDrop                     = 11
On_DocEditKeyDown                  = 12
On_DocEditKeyUp                    = 13
On_DocEditKeyPressed               = 14
On_DocEditMouseMove                = 15
On_DocEditButtonUp                 = 16
On_DocEditButtonDown               = 17
On_DocEditContextMenu              = 18
On_DocEditPaste                    = 19
On_DocEditCut                      = 20
On_DocEditCopy                     = 21
On_DocEditClear                    = 22
On_DocEditSelectionChanged         = 23
On_DocEditDragOver                 = 10
On_BeforeOpenProject               = 25
On_BeforeOpenDocument              = 26
On_BeforeSaveDocument              = 27
On_BeforeCloseDocument             = 28
On_ViewActivation                  = 29
On_DocEditKeyboardEvent            = 30
```

```
On_DocEditMouseEvent                = 31
On_BeforeValidate                   = 32
On_BeforeShowSuggestions            = 33
On_ProjectOpened                    = 34
On_Char                             = 35
On_Initialize                       = 36
On_Running                          = 37
On_Shutdown                         = 38
On_AuthenticBeforeSave              = 39
On_AuthenticContextMenuActivated    = 40
On_AuthenticLoad                    = 41
On_AuthenticToolbarButtonClicked    = 42
On_AuthenticToolbarButtonExecuted   = 43
On_AuthenticUserAddedXMLNode        = 44
```

The names of the events are the same as they appear in the Scripting Environment of XMLSpy. For IDE plug-ins the names used are immaterial. The events are identified using the ID value.

`arrayParameters` is an array which is filled with the parameters of the currently raised event. Order, type, and meaning of the single parameters are available through the scripting environment of XMLSpy. The events module of a scripting project contains predefined functions for all events prior to version 4.4. The parameters passed to the predefined functions are identical to the array elements of the `arrayParameters` parameter.

Events raised from the Authentic View of XMLSpy do not pass any parameters directly. An "event" object is used instead. The event object can be accessed through the `Document` object of the active document.

`pXMLSpy` holds a reference to the dispatch interface of the `Application` object of XMLSpy.

If the return value of `OnEvent()` is set, then neither the IDE plug-in nor an event handler inside of the scripting environment will get this event afterwards. Please note that all IDE plug-ins get/process the event before the Scripting Environment does.

29.2.5.4 GetUIModifications

Declaration

```
GetUIModifications() as String
```

Description

The `GetUIModifications()` method is called during initialization of the plug-in, to get the configuration XML data that defines the changes to the UI of XMLSpy. The method is called when the plug-in is loaded for the first time, and at every start of XMLSpy. *See also Configuration XML* 1531 *for a detailed description how to change the UI.*

Example

```
Public Function IXMLSpyPlugIn_GetUIModifications() As String
   ' GetUIModifications() gets the XML file with the specified modifications of
   ' the UI from the config.xml file in the plug-in folder
```

```
   Dim strPath As String
   strPath = App.Path

   If Len(strPath) > 0 Then
      Dim fso As New FileSystemObject
      Dim file As file
      Set file = fso.GetFile(strPath & "\config.xml")

      If (Not (file Is Nothing)) Then
         Dim stream As TextStream
         Set stream = file.OpenAsTextStream(ForReading)

         ' this replaces the token '**path**' from the XML file with
         ' the actual installation path of the plug-in to get the image file
         Dim strMods As String
         strMods = stream.ReadAll
         strMods = Replace(strMods, "**path**", strPath)

         IXMLSpyPlugIn_GetUIModifications = strMods
      Else
         IXMLSpyPlugIn_GetUIModifications = ""
      End If
   End If
End Function
```

29.2.5.5 GetDescription

Declaration

```
GetDescription() as String
```

Description

`GetDescription()` is used to define the description string for the plug-in entries visible in the Customize dialog box.

Example

```
Public Function IXMLSpyPlugIn_GetDescription() As String
    IXMLSpyPlugIn_GetDescription = "Sample Plug-in for XMLSpy;This Plug-in demonstrates
the implementation of a simple VisualBasic DLL as a Plug-in for XMLSpy."
End Function
```

29.3 Application API

The COM-based API of XMLSpy (also called the Application API from now on) enables other applications to use the functionality of XMLSpy. As a result, it is possible to automate a wide range of tasks, from validating an XML file to modifying complex XML content (with the XMLData[1795] interface).

XMLSpy and its Application API follow the common specifications for automation servers set out by Microsoft. It is possible to access the methods and properties of the Application API from common development environments, such as those using C#, C++, VisualBasic, and Delphi, and with scripting languages like JScript and VBScript.

Execution environments for the Application API

The Application API can be accessed from the following execution environments:

- External programs (described below[1544] and in the Overview[1545] part of this section)
- From within the built-in Scripting Editor of XMLSpy. For a description of the scripting environment, see the section, Scripting Editor[1505].
- XMLSpy allows you to create and integrate your own plug-ins into the application using a special interface for plug-ins. A description of how to create plug-ins is given in the section IDE Plug-ins[1530].
- Via an ActiveX Control, which is available if the integration package[1905] is installed. For more information, see the section ActiveX Integration[1905].

External programs

In the Overview[1545] part of this section, we describe how the functionality of XMLSpy can be accessed and automated from external programs.

Using the Application API from outside XMLSpy requires an instance of XMLSpy to be started first. How this is done depends on the programming language used. See the section, Programming Languages[1546], for information about individual languages.

Essentially, XMLSpy will be started via its COM registration. Then the `Application` object associated with the XMLSpy instance is returned. Depending on the COM settings, an object associated with an already running XMLSpy can be returned. Any programming language that supports creation and invocation of COM objects can be used. The most common of these are listed below.

- JScript and VBScript[1555] script files have a simple syntax and are designed to access COM objects. They can be run directly from a DOS command line or with a double click on Windows Explorer. They are best used for simple automation tasks.
- C#[1558] is a full-fledged programming language that has a wide range of existing functionality. Access to COM objects can be automatically wrapped using C#..
- C++ provides direct control over COM access but requires relatively larger amounts of code than the other languages.
- Java[1567]: Altova products come with native Java classes that wrap the Application API and provide a full Java look-and-feel.
- Other programming languages that make useful alternatives are: Visual Basic for Applications, Perl, and Python.

Programming points

The following limitations must be considered in your client code:

(C) 2015-2021 Altova GmbH

- Be aware that if your client code crashes, instances of XMLSpy may still remain in the system.
- Don't hold references to objects in memory longer than you need them, especially those from the `XMLData` interface. If the user interacts between two calls of your client, then there is no guarantee that these references are still valid.
- Don't forget to disable dialogs if the user interface is not visible.
- See Error handling in JScript[1550] (and in C#[1565] and Java[1574]) for details of how to avoid annoying error messages.
- Free references explicitly if you are using C# or C++.

This documentation

This documentation section about the Application API is broadly divided into two parts.

- The first part consists of an Overview[1545], which describes the object model for the API and explains how the API is accessed via various programming languages[1546].
- The second part is a reference section (Interfaces[1579] and Enumerations[1843]) that contains descriptions of the interface objects of the Application API.

29.3.1 Overview

This overview of the Application API is organized as follows:

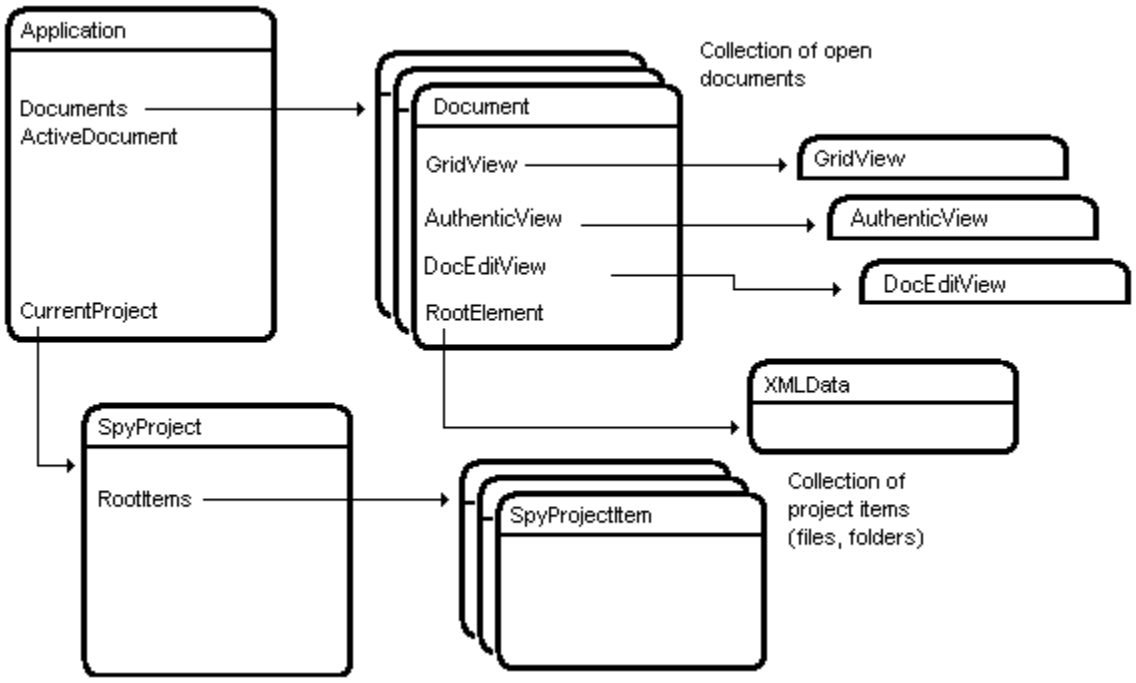
- The Object Model[1545] describes the relationships between the objects of the Application API.
- Programming Languages[1546] explains how the most commonly used programming languages (JScript, VBScript, C#, and Java) can be used to access the functionality of the Application API. Code listings from the example files supplied with your application package are used to describe basic mechanisms.
- The DOM and XMLData[1575] explains the relationship between the Application API's XMLData interface and the DOM.
- Obsolete: Authentic View Row Operations[1577] supplies information about obsolete objects for Authentic View table row operations.
- Obsolete: Authentic View Editing Operations[1578] supplies information about obsolete objects for Authentic View editing operations.

29.3.1.1 Object Model

The starting point for every application which uses the Application API is the `Application`[1580] object. This object contains general methods like import/export support and references to the open documents and any open project.

The `Application` object is created differently in various programming languages. In scripting languages such as JScript or VBScript, this involves calling a function which initializes the application's COM object. For examples, see the Programming Languages[1546] section.

The picture below shows the links between the main objects of the Application API:



The application object consists of the following parts:

1. Document collection and reference to the active document.
2. Reference to current project and methods for creating and opening projects.
3. Methods to support the export to and import from databases, text files, and Word documents.
4. URL management.
5. Methods for macro menu items.

Once you have created an Application object you can start using the functionality of XMLSpy. In most cases, you either open a project and access the documents from there or you directly open a document via the Documents[1698] interface.

29.3.1.2 Programming Languages

Programming languages differ in the way they support COM access. A few examples for the most frequently used languages (*links below*) will help you get started. The code listings in this section show how basic functionality can be accessed. The files in the `API` subfolder of the `Examples` folder can be used to test this functionality:

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

JScript

The JScript listings demonstrate the following basic functionality:

- Start application or attach to a running instance[1548]

- Simple document access 1549
- Iteration 1550
- Error handling 1550
- Events 1551
- Import and export of data 1552

VBScript

VBScript is different than JScript only syntactically; otherwise it works in the same way. The link below goes to an example of how VBScript can be used. For more information, refer to the JScript examples 1547.

- Events 1556: Shows how events are handled using VBScript.

C#

C# can be used to access the Application API functionality. The code listings show how to access the API for certain basic functionality.

- Start XMLSpy 1563: Starts XMLSpy, which is registered as an automation server, or activates the program if XMLSpy is already running.
- Open OrgChart.pxf 1564: Locates one of the example documents installed with XMLSpy and opens it. If this document is already open it becomes the active document.
- OnDocumentOpened Event On/Off 1566: Shows how to listen to XMLSpy events. When turned on, a message box will pop up after a document has been opened.
- Open ExpReport.xml 1564: Opens another example document.
- Toggle View Mode 1565: Changes the view of all open documents between Text View and Authentic View. The code shows how to iterate through open documents.
- Validate 1565: Validates the active document and shows the result in a message box. The code shows how to handle errors and COM output parameters.
- Shutdown XMLSpy 1563: Stops XMLSpy.

Java

The XMLSpy API can be accessed from Java code. The Java sub-section of this section 1567 explains how some basic XMLSpy functionality can be accessed from Java code. It is organized into the following sub-sections:

- Mapping Rules for the Java Wrapper 1567
- Example Java Project 1569
- Application Startup and Shutdown 1572
- Simple Document Access 1573
- Iterations 1573
- Use of Out-Parameters 1574
- Event Handlers 1574

29.3.1.2.1 JScript

This section contains listings of JScript code that demonstrate the following basic functionality:

- Start application or attach to a running instance 1548

- Simple document access 1549
- Iteration 1550
- Error handling 1550
- Events 1551
- Import and export of data 1552

Example files

The code listings in this section are available in example files that you can test as is or modify to suit your needs. The JScript example files are located in the `JScript` subfolder of the API Examples folder:

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

The example files can be run in one of two ways:

- *From the command line:* Open a command prompt window, change the directory to the path above, and type the name of one of the example scripts (for example, `Start.js`).
- *From Windows Explorer:* In Windows Explorer, browse for the JScript file and double-click it.

The script is executed by Windows Script Host that is packaged with Windows operating system. For more information about Windows Script Host, refer to MSDN documentation (https://msdn.microsoft.com).

29.3.1.2.1.1 Start Application

The JScript below starts the application and shuts it down. If the COM object of the 32-bit XMLSpy cannot be found, the code attempts to get the COM object of the 64-bit application; otherwise, an error is thrown. If an instance of the application is already running, the running instance will be returned.

メモ 32-ビット XMLSpy では、登録された名前、またはCOM オブジェクトのプログラム識別子（ProgId）は以下の通りです: `MapForce.ApplicationXMLSpy.Application`。64-ビット XMLSpy では、名前は以下の通りです: `MapForce_x64.ApplicationXMLSpy_x64.Application`。プログラムの呼び出しは自身のレジストリハイブ、またはグループ（32-bit または64-bit）内のCLASSES レジストリエントリにアクセスすることにご注意してください。ですから、標準のコマンドプロンプトと上 64-bit Windows のWindows Explorerを使用してスクリプトを実行すると、64-ビット XMLSpy. を指す64-ビットレジストリエントリにアクセスされます。この理由のため、XMLSpy 32-ビットと64-ビットの両方がインストールされている場合、32-ビット XMLSpy. を呼び出すために特別な処理が必要になります。例えば、スクリプトホストがプログラムを呼び出す場合以下を行います:

1. ディレクトリを **C:\Windows\SysWOW64** に変更します。
2. コマンドライン上で **wscript.exe** を入力し実行するスクリプトパスを以下のように入力します:

```
wscript.exe "C:\Users\...
\Documents\Altova\StyleVision2021\StyleVisionExamples\API\JScript\start.js"wscript.exe
"C:\Users\...\Documents\Altova\XMLSpy2021\Examples\API\JScript\start.js"
```

```
// Initialize application's COM object. This will start a new instance of the
application and
```

```
// return its main COM object. Depending on COM settings, the main COM object of an
already
// running application might be returned.

try {   objSpy = WScript.GetObject("", "XMLSpy.Application");   }
catch(err) {}

if( typeof( objSpy ) == "undefined" )
{
   try   {   objSpy = WScript.GetObject("", "XMLSpy_x64.Application")   }
   catch(err)
   {
      WScript.Echo( "Can't access or create XMLSpy.Application" );
      WScript.Quit();
   }
}

// if newly started, the application will start without its UI visible. Set it to
visible.
objSpy.Visible = true;

WScript.Echo(objSpy.Edition + " has successfully started. ");

objSpy.Visible = false;      // will shutdown application if it has no more COM
connections
//objSpy.Visible = true;   // will keep application running with UI visible
```

The JScript code listed above is available in the sample file `Start.js` (see Example Files 1548).

29.3.1.2.1.2 Simple Document Access

After you have started the application as shown in Start Application 1548, you will most likely want to programmatically open a document in order to work with it. The JScript code listing below illustrates how to open two documents from the XMLSpy Examples folder and set one of them as the active document.

```
// Locate examples via USERPROFILE shell variable. The path needs to be adapted to
major release versions.
objWshShell = WScript.CreateObject("WScript.Shell");
majorVersionYear = objSpy.MajorVersion + 1998
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\My
Documents\\Altova\\XMLSpy" + majorVersionYear + "\\Examples\\";

// Tell XMLSpy to open two documents. No dialogs
objDoc1 = objSpy.Documents.OpenFile(strExampleFolder + "OrgChart.pxf", false);
objSpy.Documents.OpenFile(strExampleFolder + "ExpReport.xml", false);

// The document currently active can be easily located.
objDoc2 = objSpy.ActiveDocument;

// Let us make sure that the document is shown in grid view.
objDoc2.SwitchViewMode(0);   // SPYViewModes.spyViewGrid = 0

// Now switch back to the document opened first
objDoc1.SetActiveDocument();
```

The JScript code listed above is available in the sample file `DocumentAccess.js` (see Example Files[1548]).

29.3.1.2.1.3 Iteration

The JScript listing below shows how to iterate through the open documents. It is assumed that you have already started the application and opened some documents as shown in the previous sections.

```
// go through all open documents using a JScript Enumerator
bRequiresSaving = false;
for (var iterDocs = new Enumerator(objSpy.Documents); !iterDocs.atEnd();
iterDocs.moveNext())
{
   if (iterDocs.item().IsModified)
      bRequiresSaving = true;

   var strErrorText = new Array(1);
   var nErrorNumber = new Array(1);
   var errorData = new Array(1);

   if (!iterDocs.item().IsValid(strErrorText, nErrorNumber, errorData))
   {
       var text = strErrorText;
      // access that XMLData object only if filled in
       if (errorData[0] != null)
         text += "(" + errorData[0].Name + "/" + errorData[0].TextValue + ")";

      WScript.Echo("Document \"" + iterDocs.item().Name +"\" validation error[" +
nErrorNumber + "]: " + text);
   }
   else
   {
      // The COM call succeeded and the document is valid.
      WScript.Echo("Document \"" + iterDocs.item().Name + "\" is valid.");
   }
}

// go through all open documents using index-based access to the document collection
for (i = objSpy.Documents.Count; i > 0; i--)
   objSpy.Documents.Item(i).Close(false);
```

The JScript code listed above is available in the sample file `DocumentAccess.js` (see Example Files[1548]).

29.3.1.2.1.4 Error Handling

The Application API returns errors in two different ways:

- The `HRESULT` returned by every API method
- The `IErrorInfo` interface of the Application API

Every API method returns an HRESULT. This return value gives the caller information about errors during execution of the method. If the call was successful, the return value is S_OK. The HRESULT option is commonly used in C/C++ programs.

However, programming languages such as VisualBasic and scripting languages (and other high-level development environments) don't give the programmer access to the HRESULT return of a COM call. Such languages use the IErrorInfo interface, which is also supported by the Application API. If an error occurs, the Application API creates a new object that implements the IErrorInfo interface. The information provided by the IErrorInfo interface is imported by the development environment into its own error-handling mechanism.

For example, the JScript code listing below causes an error to be thrown by incorrectly declaring an array. Additional information about the error object is provided by its properties number and description.

```
try {
   var arr = new Array(-1);
}
catch (err) {
   WScript.Echo("Error : (" + (err.number & 0xffff) + ")" + err.description);
}
```

29.3.1.2.1.5 Events

COM specifies that a client must register itself at a server for callbacks using the connection point mechanism. The automation interface for XMLSpy defines the necessary event interfaces. The way to connect to those events depends on the programming language you use in your client. The following code listing shows how this is done using JScript.

The method WScript.ConnectObject is used to receive events.

```
// The event-handler function
function DocEvent_OnBeforeCloseDocument(objDocument)
{
   WScript.Echo("Received event - before closing document");
}

// Create or connect to XMLSpy (or Authentic Desktop)
try
{
   // Create the environment and XMLSpy (or Authentic Desktop)
   objWshShell = WScript.CreateObject("WScript.Shell");
   objFSO = WScript.CreateObject("Scripting.FileSystemObject");
   objSpy = WScript.GetObject("", "XMLSpy.Application");

// If only Authentic Desktop is installed (and XMLSpy is not installed) use:
// objSpy = WScript.GetObject("", "AuthenticDesktop.Application")

}
catch(err)
   { WScript.Echo ("Can't create WScript.Shell object or XMLSpy"); }

// Create document object and connect to its events
objSpy.Visible = true;
majorVersionYear = objSpy.MajorVersion + 1998
```

```
docPath = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\Documents\\Altova\
\XMLSpy" + majorVersionYear + "\\Examples\\ExpReport.xml";
objDoc = objSpy.Documents.OpenFile (docPath, false);
WScript.ConnectObject(objDoc, "DocEvent_");

// Keep running while waiting for the event
// In the meanwhile close this document in XMLSpy (or Authentic Desktop) manually
WScript.Echo ("Sleeping for 10 seconds ...");
WScript.Sleep (10000);

objDoc = null;
WScript.Echo ("Stopped listening for event");
objSpy.Quit();
```

29.3.1.2.1.6 Import and Export of Data

Before you implement your import and export tasks with the Application API, it is good practice to test the connections, parameters, SQL queries and so on in XMLSpy. In this way you are able to verify the results and make quick adjustments to import or export parameters. Most of the methods for importing and exporting data are placed in the `Application` 1580 object; the remaining functions are accessible via the `Document` 1668 interface.

There is some preparatory work necessary before the actual import or export can be started. Every import/export job consists of two parts. You need to define a connection to your data and the specific behavior for the import/export process. In case of an import, the connection is either a database, a text-file, or a Word document. The main issue is which data (columns) to imported into XMLSpy. In case of an export, the connection is either a database or a text file. Specify which data (elements of the XML file) and additional parameters (for example, automatic key generation or number of sub-levels) to use from the XML-structure.

The properties in the `DatabaseConnection`, `TextImportExportSettings` and `ExportSettings` interfaces have default values. See the corresponding descriptions in the `Interfaces` 1579 chapter for further information.

The sub-sections of this section describe each of these operations in detail.

- Import from Database 1552
- Export to Database 1553
- Import from Text 1554
- Export to Text 1555

Given below are the steps to establish a connection to an existing database for import:

1. Use a `DatabaseConnection` 1658 object and set the following:
   - The method `Application.GetDatabaseSettings` 1587 returns a new object for a database connection: `objImpSettings = objSpy.GetDatabaseSettings();`
   - You have to set either an ADO connection string, `objImpSettings.ADOConnection = strADOConnection` or the path to an existing database file: `objImpSettings.File = strExampleFolder + "Tutorial\\Company.mdb";`
   - To complete the settings you create an SQL SELECT statement to define the data to be queried: `objImpSettings.SQLSelect = "SELECT * FROM Address";`
2. Call `Application.GetDatabaseImportElementList` 1586 to get a collection of the resulting columns of the SQL query: `objElementList = objSpy.GetDatabaseImportElementList(objImpSettings);` This collection gives you the

opportunity to control which columns should be imported and specify the datatype of the new elements. Each item of the collection represents one column to import. If you remove an item, the corresponding column will not be imported. You can additionally modify the `ElementListItem.ElementKind` [1707] property to set the datatype of the XML elements for each column. Please consider that `GetDatabaseImportElementList()` executes the SQL query and could initiate a time-consuming call. To avoid this, it is possible to pass a null-pointer as the second parameter to `ImportFromDatabase();` this imports all columns as plain XML elements.

3. Start the import with `Application.ImportFromDatabase` [1590]: `objImpDocFromDB = objSpy.ImportFromDatabase(objImpSettings,objElementList);`

```
// Locate examples via USERPROFILE shell variable.
objWshShell = WScript.CreateObject("WScript.Shell");
majorVersionYear = objSpy.MajorVersion + 1998
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\My
Documents\\Altova\\XMLSpy" + majorVersionYear + "\\Examples\\";

try
{
   // specify the source of data import
   objImpSettings = objSpy.GetDatabaseSettings();
   objImpSettings.File = strExampleFolder + "Tutorial\\Company.mdb";
   objImpSettings.SQLSelect = "SELECT * FROM Address";

   // column filter
   objElementList = objSpy.GetDatabaseImportElementList(objImpSettings);

   // import into a new XML file
   objImpDocFromDB = objSpy.ImportFromDatabase(objImpSettings,objElementList);
}
catch(err)
{
   WScript.Echo("Error importing from database.\n\n" +
                     "Error: " + (err.number & 0xffff) + "\n" +
                     "Description: " + err.description);
}
```

The JScript code listed above is available in the sample file `ImportExport.js` (see Example Files [1548]).

To export data to a database, carry out the steps below:

1. Use a `DatabaseConnection` [1658] object and set the necessary properties. All properties except `SQLSelect` are important for the export. `ADOConnection or File` defines the target for the output. You need to set only one of them.
2. Fill an `ExportSettings` [1708] object with the required values. These properties are the same options as those available in the export dialog of XMLSpy. Select the menu option **Convert | Export to Text files/Database** to see the options and try a combination of export settings. After that it is easy to transfer these settings to the properties of the interface. Call `Application.GetExportSettings` [1588] to get an `ExportSettings` object: `objExpSettings = objSpy.GetExportSettings()`
3. Build an element list with `Document.GetExportElementList` [1686]. The element list enables you to eliminate XML elements from the export process. It also gives you information about the record and field count in the `RecordCount` and `FieldCount` properties. Set the `ExportSettings.ElementList` [1709] property to this collection. It is possible to set the element list to `null/Nothing` (default) to export all elements.
4. Call `Document.ExportToDatabase` [1680] to execute the export. The description of the `ExportToDatabase` method contains also a code example for a database export.

```
   // set the behaviour of the export with ExportSettings
   objExpSettings = objSpy.GetExportSettings()

   //set the destination with DatabaseConnection
   objDB = objSpy.GetDatabaseSettings();
   objDB.CreateMissingTables = true;
   objDB.CreateNew = true;
   objDB.File = "C:\\Temp\\Export.mdb";

try
{
   objImpDocFromDB.ExportToDatabase(objImpDocFromDB.RootElement, objExpSettings,
objDB);
}
catch(err)
{
   WScript.Echo("Error exporting to database.\n\n" +
                    "Error: " + (err.number & 0xffff) + "\n" +
                    "Description: " + err.description);
}
```

The JScript code listed above is available in the sample file `ImportExport.js` (see Example Files[1548]).

Importing data from a text file is similar to the import from a database. You must use other interfaces (described in steps 1 to 3 below) with different methods and properties:

1. Use a `TextImportExportSettings`[1753] object and set the properties: The method `Application.GetTextImportExportSettings`[1589] returns a new object to specify a text file for import: `objImpSettings = objSpy.GetTextImportExportSettings();` You have to set at least the `ImportFile` property to the path of the file for the import. Another important property is `HeaderRow`. Set it to `False` if the text file does not contain a leading line as a header row: `objImpSettings.ImportFile = strExampleFolder + "Tutorial\\Shapes.txt";`
2. Call `Application.GetTextImportElementList`[1588] to get a collection of all columns inside the text file: `objElementList = objSpy.GetTextImportElementList(objImpSettings);`
3. Start the import with `Application.ImportFromText`[1592]: `objImpDocFromText = objSpy.ImportFromText(objImpSettings,objElementList);`

```
try
{
   // specify the source of data import
   objImpSettings = objSpy.GetTextImportExportSettings();
   objImpSettings.ImportFile = strExampleFolder + "Tutorial\\Shapes.txt";
   objImpSettings.HeaderRow = false;

   // column filter
   objElementList = objSpy.GetTextImportElementList(objImpSettings);

   // import into a new XML file
   objImpDocFromText = objSpy.ImportFromText(objImpSettings,objElementList);
}
catch(err)
{
   WScript.Echo("Error importing from text file.\n\n" +
                    "Error: " + (err.number & 0xffff) + "\n" +
```

```
                    "Description: " + err.description);
}
```

The JScript code listed above is available in the sample file `ImportExport.js` (see Example Files[1548]).

To export data to text, carry out the steps below:

1. Use a `TextImportExportSettings`[1753] object and set the necessary properties.
2. Fill an `ExportSettings`[1708] object with the required values. See Item 2 in Export to database[1553].
3. Build an element list with `Document.GetExportElementList`[1686]. See Item 3 in Export to database[1553].
4. Call `Document.ExportToText`[1681] to execute the export.

```
objExpSettings = objSpy.GetExportSettings();
objExpSettings.ElementList =
objImpDocFromText.GetExportElementList(objImpDocFromText.RootElement, objExpSettings);

objTextExp = objSpy.GetTextImportExportSettings();
objTextExp.HeaderRow = true;
objTextExp.DestinationFolder = "C:\\Temp";

try
{

       objImpDocFromText.ExportToText(objImpDocFromText.RootElement, objExpSettings,
objTextExp);
}
catch(err)
{
   WScript.Echo("Error exporting to text.\n\n" +
                    "Error: " + (err.number & 0xffff) + "\n" +
                    "Description: " + err.description);
}
```

The JScript code listed above is available in the sample file `ImportExport.js` (see Example Files[1548]).

29.3.1.2.2 VBScript

VBScript is syntactically different than JScript but works in the same way. This section contains a listing showing how events are used with VBScript[1556] and an example[1557].

For information about other functionality, refer to the JScript examples listed below:

- Start application or attach to a running instance[1548]
- Simple document access[1549]
- Iteration[1550]
- Error handling[1550]
- Import and export of data[1552]

29.3.1.2.2.1 Events

COM specifies that a client must register itself at a server for callbacks using the connection point mechanism. The automation interface for XMLSpy defines the necessary event interfaces. The way to connect to those events depends on the programming language you use in your client. The following code listing shows how this is done using VBScript.

The method `WScript.ConnectObject` is used to receive events.

To run this code, paste it into a file with .vbs extension, and either double-click in Windows Explorer, or run it from a command prompt.

```
' the event handler function
Function DocEvent_OnBeforeCloseDocument(objDocument)
   Call WScript.Echo("received event - before closing document")
End Function

' create or connect to XmlSpy
Set objWshShell = WScript.CreateObject("WScript.Shell")
Set objFSO = WScript.CreateObject("Scripting.FileSystemObject")
Set objSpy = WScript.GetObject("", "XMLSpy.Application")
' If only Authentic is installed (and XMLSpy is not installed) use:
' Set objSpy = WScript.GetObject("", "AuthenticDesktop.Application")
' If only XMLSpy 64-bit is intalled, use:
' Set objSpy = WScript.GetObject("", "XMLSpy_x64.Application")

' create document object and connect to its events
objSpy.Visible = True

' Find out user's personal folder and locate one of the installed examples.
personalFolder = objWshShell.ExpandEnvironmentStrings("%UserProfile%")
majorVersionYear = objSpy.MajorVersion + 1998
xmlspyExamplesFolder = personalFolder & "\Documents\Altova\XMLSpy" & majorVersionYear
& "\Examples\"
docPath = xmlspyExamplesFolder & "ExpReport.xml"

' open a document
Set objDoc = objSpy.Documents.OpenFile (docPath, False)
Call WScript.ConnectObject(objDoc, "DocEvent_")

' keep running while waiting on the event
' in the meantime close the document in XMLSPY manually
Call WScript.Echo ("sleeping for 10 seconds ...")
Call WScript.Sleep (10000)

Set objDoc = Nothing
Call WScript.Echo ("stopped listening for event")
Call objSpy.Quit
```

メモ 32-ビット XMLSpy では、登録された名前、またはCOM オブジェクトのプログラム識別子（ProgId）は以下の通りです: `MapForce.ApplicationXMLSpy.Application`。64-ビット XMLSpy では、名前は以下の通りです: `MapForce_x64.ApplicationXMLSpy_x64.Application`。プログラムの呼び出しは自身のレジストリハイブ、またはグループ（32-bit または64-bit）内のCLASSES レジストリエントリにアクセスすることに注意してください。ですから、標準のコマンドプロンプトと上 64-bit W indows のW indows Explorerを使用してスクリプトを実行すると、64-ビット XMLSpy. を指

す 64-ビットレジストリエントリにアクセスされます。この理由のため、XMLSpy 32-ビットと64-ビットの両方がインストールされている場合、32-ビット XMLSpy. を呼び出すために特別な処理が必要になります。例えば、スクリプトホストがプログラムを呼び出す場合以下を行います:

1. ディレクトリを **C:\Windows\SysWOW64** に変更します。
2. コマンドライン上で **wscript.exe** を入力し実行するスクリプトパスを以下のように入力します:

```
wscript.exe "C:\Users\...
\Documents\Altova\StyleVision2021\StyleVisionExamples\API\JScript\start.js"wscript.exe
"C:\Users\...\Documents\Altova\XMLSpy2021\Examples\API\JScript\start.js"
```

29.3.1.2.2.2 Example: Using Events

Authentic View supports event connection on a per-object basis. Implementation of this feature is based on COM connection points and is available in environments that support this mechanism.

The following example is a VBScript code example that shows how to use events from within a VBScript project.

```
' ------------------------------------------------------------------------
' VBScript example that demonstrates how to use events.
' ------------------------------------------------------------------------

' Event handler for OnSelectionChanged event of AuthenticView
Function AuthenticViewEvent_OnSelectionChanged(objAuthenticRange)
    If objAuthenticRange.FirstTextPosition <> objAuthenticRange.LastTextPosition Then
        Call WScript.Echo("Selection: " & objAuthenticRange.Text & vbNewLine &
vbNewLine & "Close this dialog.")
    Else
        Call WScript.Echo("Cursor position: " & objAuthenticRange.FirstTextPosition &
vbNewLine & vbNewLine & "Close this dialog.")
    End If
End Function

' Start/access XMLSpy and connect to its automation interface.
Set WshShell = WScript.CreateObject("WScript.Shell")
Set objSpy = GetObject("", "XMLSpy.Application")
' Make the UI of XMLSpy visible.
objSpy.Visible = True

' Find out user's personal folder and locate one of the installed XMLSpy examples.
personalFolder = WshShell.ExpandEnvironmentStrings("%UserProfile%")
majorVersionYear = objSpy.MajorVersion + 1998
xmlspyExamplesFolder = personalFolder & "\Documents\Altova\XMLSpy" & majorVersionYear
& "\Examples\"
docPath = xmlspyExamplesFolder & "ExpReport.xml"

' Create object to access windows file system and test if the our document exists.
Set fso = CreateObject("Scripting.FileSystemObject")
If fso.FileExists(docPath) Then
    ' open the document
    Call objSpy.Documents.OpenFile(docPath, False)
    set objDoc = objSpy.ActiveDocument

    ' switch active document to authentic view
```

```
    objDoc.SwitchViewMode 4 ' spyViewAuthentic

    ' Register for connection point events on the authentic view of the active
document.
    ' Any function with a valid event name prefixed with "AuthenticViewEvent_" will
    ' be called when the corresponding event gets triggered on the specified object.
    set objView = objDoc.AuthenticView
    Call WScript.ConnectObject(objView, "AuthenticViewEvent_")
    Call WScript.Echo("Events are connected." & vbNewLine & vbNewLine & "Now set or
move the cursor in XMLSpy." & vbNewLine & vbNewLine & "Close this dialog to shut down
XMLSpy.")

    ' To disconnect from the events delete the reference to the object.
    set objView = Nothing
Else
    Call WScript.Echo("The file " & docPath & " does not exist.")
End If

' shut down XMLSpy when this script ends
objSpy.Visible = False
```

29.3.1.2.3 C#

The C# programming language can be used to access the Application API functionality. You could use Visual Studio 2010/2012/2013/2015/2017/2019 to create the C# code, saving it in a Visual Studio project. Create the project as follows:

1. In Microsoft Visual Studio, add a new project using **File | New | Project**.
2. Add a reference to the XMLSpy Type Library by clicking **Project | Add Reference**. The Add Reference dialog appears. Browse for the XMLSpy Type Library component, which is located in the XMLSpy application folder, and add it.
3. Enter the code you want.
4. Compile the code and run it.

Example C# project

Your XMLSpy package contains an example C# project, which is located in the `API\C#` subfolder of the `Examples` folder :

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019. The code listing below shows how basic application functionality can be used. This code is similar to the example C# project in the API Examples folder of your application package, but might differ slightly.

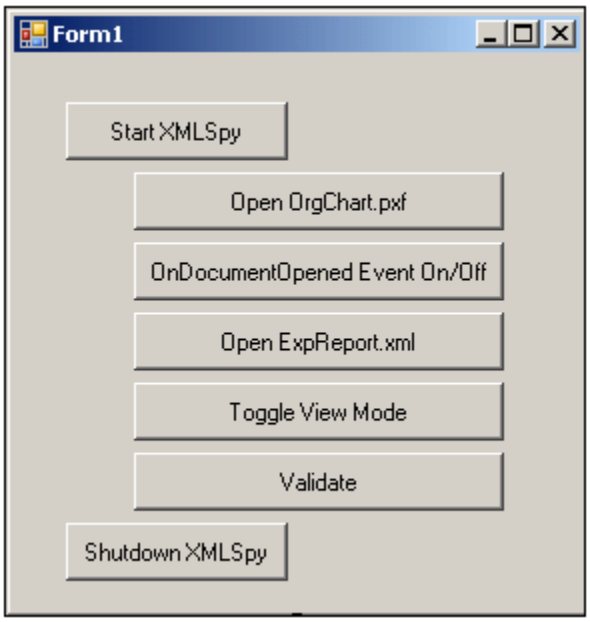
Platform configuration

If you have a 64-bit operating system and are using a 32-bit installation of XMLSpy, you must add the x86 platform in the solution's Configuration Manager and build the sample using this configuration. A new x86 platform (for the active solution in Visual Studio) can be created in the New Solution Platform dialog (**Build | Configuration Manager | Active solution platform | <New...>**).

(C) 2015-2021 Altova GmbH

What the code listing below does

The example code listing below creates a simple user interface (*screenshot below*) with buttons that invoke basic XMLSpy operations:



- Start XMLSpy 1563: Starts XMLSpy, which is registered as an automation server, or activates the application if it is already running.
- Open OrgChart.pxf 1564: Locates one of the example documents installed with XMLSpy and opens it. If this document is already open it becomes the active document.
- OnDocumentOpened Event On/Off 1566: Shows how to listen to XMLSpy events. When turned on, a message box will pop up after a document has been opened.
- Open ExpReport.xml 1564: Opens another example document.
- Toggle View Mode 1565: Changes the view of all open documents between Text View and Authentic View. The code shows how to iterate through open documents.
- Validate 1565: Validates the active document and shows the result in a message box. The code shows how to handle errors and COM output parameters.
- Shut down XMLSpy 1563: Stops XMLSpy.

You can modify the code (of the code listing below or of the example C# project in the API Examples folder) in any way you like and run it.

Compiling and running the example

In the API Examples folder, double-click the file `AutomateXMLSpy_VS2008.sln` or the file `AutomateXMLSpy_VS2010.sln` (to open in Visual Studio 2010/2012/2013/2015/2017/2019). Alternatively the file can be opened from within Visual Studio (with **File | Open | Project/Solution**). To compile and run the example, select **Debug | Start Debugging** or **Debug | Start Without Debugging**.

Code listing of the example

Given below is the C# code listing of the basic functionality of the form (`Form1.cs`) created in the `AutomateXMLSpy` example. Note that the code listed below might differ slightly from the code in the API

Examples form.The listing below is commented for ease of understanding. Parts of the code are also presented separately in the sub-sections of this section, according to the Application API functionality they access.

The code essentially consists of a series of handlers for the buttons in the user interface shown in the screenshot above.

```
namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        // An instance of XMLSpy accessed via its automation interface
        XMLSpyLib.Application XMLSpy;

        // Location of examples installed with XMLSpy
        String strExamplesFolder;

        private void Form1_Load(object sender, EventArgs e)
        {
            // Locate examples installed with XMLSpy
            // REMARK: You might need to adapt this if you have a different major version
of the product
            strExamplesFolder = Environment.GetEnvironmentVariable("USERPROFILE") + "\\My
Documents\\Altova\\XMLSpy2012\\Examples\\";
        }

        // Handler for the "Start XMLSpy" button
        private void StartXMLSpy_Click(object sender, EventArgs e)
        {
            if (XMLSpy == null)
            {
                Cursor.Current = Cursors.WaitCursor;

                // If no XMLSpy instance is open, create one and make it visible
                XMLSpy = new XMLSpyLib.Application();
                XMLSpy.Visible = true;

                Cursor.Current = Cursors.Default;
            }
            else
            {
                // If an instance of XMLSpy is already running, make sure it's visible
                if (!XMLSpy.Visible)
                    XMLSpy.Visible = true;
            }
        }

        // Handler for the "Open OrgChart.pxf" button
        private void openOrgChart_Click(object sender, EventArgs e)
        {
            // Make sure there's a running XMLSpy instance, and that it's visible
            StartXMLSpy_Click(null, null);

            // Open one of the example files installed with the product
            XMLSpy.Documents.OpenFile(strExamplesFolder + "OrgChart.pxf", false);
```

```
        }

        // Handler for the "Open ExpReport.xml" button
        private void openExpReport_Click(object sender, EventArgs e)
        {
            // Make sure there's a running XMLSpy instance, and that it's visible
            StartXMLSpy_Click(null, null);

            // Open one of the sample files installed with the product.
            XMLSpy.Documents.OpenFile(strExamplesFolder + "ExpReport.xml", false);
        }

        // Handler for the "Toggle View Mode" button
        private void toggleView_Click(object sender, EventArgs e)
        {
            // Make sure there's a running XMLSpy instance, and that it's visible
            StartXMLSpy_Click(null, null);

            // Iterate through all open documents and toggle view between Text View and
Authentic View
            foreach (XMLSpyLib.Document doc in XMLSpy.Documents)
                if (doc.CurrentViewMode == XMLSpyLib.SPYViewModes.spyViewText)
                    doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewAuthentic);
                else
                    doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewText);
        }

        // Handler for the "Shutdown XMLSpy" button
        // Shut down the application instance by explicitly releasing the COM object
        private void shutdownXMLSpy_Click(object sender, EventArgs e)
        {
            if (XMLSpy != null)
            {
                // Allow shutdown of XMLSpy by releasing the UI
                XMLSpy.Visible = false;

                // Explicitly release the COM object
                try
                {
                    while
(System.Runtime.InteropServices.Marshal.ReleaseComObject(XMLSpy) > 0) ;
                }
                finally
                {
                    // Disallow subsequent access to this object
                    XMLSpy = null;
                }
            }
        }

        // Handler for button "Validate"
        private void validate_Click(object sender, EventArgs e)
        {
            // COM errors are returned to C# as exceptions. We use a try/catch block to
handle them.
            try
            {
                // Method 'IsValid' is one of the few functions that uses output
parameters
                // Use 'object' type for these parameters
                object strErrorText = "";
                object nErrorNumber = 0;
```

```
                object errorData = null;

                if (!XMLSpy.ActiveDocument.IsValid(ref strErrorText, ref nErrorNumber,
ref errorData))
                {
                    // The COM call succeeded but the document is not valid
                    // A detailed description of the problem is returned in strErrorText,
nErrorNumber and errorData
                    listBoxMessages.Items.Add("Document " + XMLSpy.ActiveDocument.Name +
" is not valid.");
                    listBoxMessages.Items.Add("\tErrorText  : " + strErrorText);
                    listBoxMessages.Items.Add("\tErrorNumber: " + nErrorNumber);
                    listBoxMessages.Items.Add("\tElement    : " + (errorData != null ?
((XMLSpyLib.XMLData)errorData).TextValue : "null"));
                }
                else
                {
                    // The COM call succeeded and the document is valid
                    listBoxMessages.Items.Add("Document " + XMLSpy.ActiveDocument.Name +
" is valid.");
                }
            }
            catch (Exception ex)
            {
                // The COM call was not successful
                // Probably no application instance has been started or no document is
open.
                listBoxMessages.Items.Add("Error validating active document: " +
ex.Message);
            }
        }



        // Event handler for OnDocumentOpened event
        private void handleOnDocumentOpened(XMLSpyLib.Document i_ipDocument)
        {
            MessageBox.Show("Document " + i_ipDocument.Name + " was opened!");
        }

        // Remember if the event handler is currently registered.
        private bool bEventHandlerIsRegistered = false;

        // Handler for button 'OnDocuemntOpened Event On/Off
        private void toggleOnDocumentOpenedEvent_Click(object sender, EventArgs e)
        {
            if (XMLSpy != null)
            {
                if (bEventHandlerIsRegistered)
                    XMLSpy.OnDocumentOpened -= new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
                else
                    XMLSpy.OnDocumentOpened += new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);

                bEventHandlerIsRegistered = !bEventHandlerIsRegistered;
            }
        }
    }
}
```

29.3.1.2.3.1 Add Reference to XMLSpy API

Add the application's type library as a reference in a .NET project as follows: With the .NET project open, click **Project | Add Reference**. Then browse for the type library, which is called `XMLSpy.tlb`, and is located in the XMLSpy application folder.

Then declare a variable to access the XMLSpy API:

```
// An instance of XMLSpy is accessed via its automation interface.
   XMLSpyLib.Application XMLSpy;
```

29.3.1.2.3.2 Application Startup and Shutdown

In the code snippets below, the methods `StartXMLSpy_Click` and `ShutdownXMLSpy_Click` are those assigned to buttons in the AutomateXMLSpy example[1558] that, respectively, start up and shut down the application. This example is located in the C# subfolder of the API Examples folder (*see the file Form1.cs*):

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

Starting XMLSpy

The following code snippet from the AutomateXMLSpy example[1558] shows how to start up the application.

```
// Handler for the "Start XMLSpy" button
private void StartXMLSpy_Click(object sender, EventArgs e)
{
    if (XMLSpy == null)
    {
        Cursor.Current = Cursors.WaitCursor;

        // If no XMLSpy instance is running, we create one and make it visible
        XMLSpy = new XMLSpyLib.Application();
        XMLSpy.Visible = true;

        Cursor.Current = Cursors.Default;
    }
    else
    {
        // If an instance of XMLSpy is already running, make sure it's visible
        if (!XMLSpy.Visible)
            XMLSpy.Visible = true;
    }
}
```

Shutting down XMLSpy

The following code snippet from the AutomateXMLSpy example[1558] shows how to shut down the application.

```
// Handler for the "Shutdown XMLSpy" button
```

```
        // Shut down the application instance by explicitly releasing the COM object
         private void shutdownXMLSpy_Click(object sender, EventArgs e)
         {
             if (XMLSpy != null)
             {
                 // Allow shutdown of XMLSpy by releasing the UI
                 XMLSpy.Visible = false;

                 // Explicitly release COM object
                 try
                 {
                     while
(System.Runtime.InteropServices.Marshal.ReleaseComObject(XMLSpy) > 0) ;
                 }
                 finally
                 {
                     // Disallow subsequent access to this object
                     XMLSpy = null;
                 }
             }
         }
```

29.3.1.2.3.3 Opening Documents

The code snippets below (from the AutomateXMLSpy example[1558]) show how two files are opened via two separate methods assigned to two buttons in the user interface. Both methods use the same Application API access mechanism: XMLSpy.Documents.OpenFile(string, boolean)[1700].

The AutomateXMLSpy example[1558] (*see the file Form1.cs*) is located in the C# subfolder of the API Examples folder:

Windows 7、Windows 8、Windows 10	C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

Code snippet

```
        // Handler for the "Open OrgChart.pxf" button
         private void openOrgChart_Click(object sender, EventArgs e)
         {
             // Make sure there's a running XMLSpy instance, and that it's visible
             StartXMLSpy_Click(null, null);

             // Open a file from the Examples folder installed with the product
             XMLSpy.Documents.OpenFile(strExamplesFolder + "OrgChart.pxf", false);
         }

         // Handler for the "Open ExpReport.xml" button
         private void openExpReport_Click(object sender, EventArgs e)
         {
             // Make sure there's a running XMLSpy instance, and that it's visible
             StartXMLSpy_Click(null, null);
```


(C) 2015-2021 Altova GmbH

```
            // Open a file from the Examples folder installed with the product
            XMLSpy.Documents.OpenFile(strExamplesFolder + "ExpReport.xml", false);
        }
```

The file opened last will be the active file.

29.3.1.2.3.4 Iterating through Open Documents

The code snippet below (from the AutomateXMLSpy example[1558]; *see the file Form1.cs*) shows how to iterate through open documents. A condition is then tested within the iteration loop, and the document view is switched between Text View and Authentic View.

```
       // Handler for the "Toggle view mode" button
        private void toggleView_Click(object sender, EventArgs e)
        {
            // Make sure there's a running XMLSpy instance, and that it's visible
            StartXMLSpy_Click(null, null);

            // Iterate through open documents and toggle current view between text and
authentic view.
            foreach (XMLSpyLib.Document doc in XMLSpy.Documents)
                if (doc.CurrentViewMode == XMLSpyLib.SPYViewModes.spyViewText)
                    doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewAuthentic);
                else
                    doc.SwitchViewMode(XMLSpyLib.SPYViewModes.spyViewText);
        }
```

The AutomateXMLSpy example[1558] example is located in the C# subfolder of the API Examples folder:

Windows 7、Windows 8、Windows 10	C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

29.3.1.2.3.5 Errors and COM Output Parameters

The code snippet below (from the AutomateXMLSpy example[1558]) shows how to handle errors and COM output parameters. The method XMLSpy.ActiveDocument.IsValid(ref strErrorText, ref nErrorNumber, ref errorData)[1688] uses output parameters that are used, in the code snippet below, to generate an error-message text.

The AutomateXMLSpy example[1558] (*see the file Form1.cs*) is located in the C# subfolder of the API Examples folder:

Windows 7、Windows 8、Windows 10	C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

Code snippet

```
        // Handler for button "Validate"
        private void validate_Click(object sender, EventArgs e)
        {
            // COM errors are returned to C# as exceptions. We use a try/catch block to
handle them.
            try
            {
                // Method 'IsValid' is one of the few functions that uses output
parameters
                // Use 'object' type for these parameters
                object strErrorText = "";
                object nErrorNumber = 0;
                object errorData = null;

                if (!XMLSpy.ActiveDocument.IsValid(ref strErrorText, ref nErrorNumber,
ref errorData))
                {
                    // The COM call succeeded but the document is not valid
                    // A detailed description of the problem is returned in strErrorText,
nErrorNumber and errorData
                    listBoxMessages.Items.Add("Document " + XMLSpy.ActiveDocument.Name +
" is not valid.");
                    listBoxMessages.Items.Add("\tErrorText  : " + strErrorText);
                    listBoxMessages.Items.Add("\tErrorNumber: " + nErrorNumber);
                    listBoxMessages.Items.Add("\tElement    : " + (errorData != null ?
((XMLSpyLib.XMLData)errorData).TextValue : "null"));
                }
                else
                {
                    // The COM call succeeded and the document is valid
                    listBoxMessages.Items.Add("Document " + XMLSpy.ActiveDocument.Name +
" is valid.");
                }
            }
            catch (Exception ex)
            {
                // The COM call was not successful
                // Probably no application instance has been started or no document is
open.
                listBoxMessages.Items.Add("Error validating active document: " +
ex.Message);
            }
        }
```

29.3.1.2.3.6 Events

The code snippet below (from the AutomateXMLSpy example[1558]) lists the code for two event handlers. The AutomateXMLSpy example[1558] (*see the file Form1.cs*) is located in the C# subfolder of the API Examples folder:

Windows 7、Windows 8、	C:\Users\<username>\Documents\

Windows 10	Altova\XMLSpy\2021\Examples

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

Code snippet

```
        // Event handler for OnDocumentOpened event
        private void handleOnDocumentOpened(XMLSpyLib.Document i_ipDocument)
        {
            MessageBox.Show("Document " + i_ipDocument.Name + " was opened!");
        }

        // Remember if the event handler is currently registered.
        private bool bEventHandlerIsRegistered = false;

        // Handler for button 'OnDocuemntOpened Event On/Off
        private void toggleOnDocumentOpenedEvent_Click(object sender, EventArgs e)
        {
            if (XMLSpy != null)
            {
                if (bEventHandlerIsRegistered)
                    XMLSpy.OnDocumentOpened -= new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
                else
                    XMLSpy.OnDocumentOpened += new
XMLSpyLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);

                bEventHandlerIsRegistered = !bEventHandlerIsRegistered;
            }
        }
```

29.3.1.2.4 Java

The Application API can be accessed from Java code. To allow accessing the XMLSpy automation server directly from Java code, the libraries listed below must reside in the `classpath`. They are installed in the folder: `JavaAPI` in the XMLSpy application folder.

- `AltovaAutomation.dll`: a JNI wrapper for Altova automation servers (`AltovaAutomation_x64.dll` in the case of 64-bit versions)
- `AltovaAutomation.jar`: Java classes to access Altova automation servers
- `XMLSpyAPI.jar`: Java classes that wrap the XMLSpy automation interface
- `XMLSpyAPI_JavaDoc.zip`: a `Javadoc` file containing help documentation for the Java API

Note: In order to use the Java API, the DLL and Jar files must be on the Java Classpath.

Example Java project

An example Java project is supplied with your product installation. You can test the Java project and modify and use it as you like. For more details of the example Java project, see the section, Example Java Project 1569.

Rules for mapping the Application API names to Java

The rules for mapping between the Application API and the Java wrapper are as follows:

- ***Classes and class names***
  For every interface of the XMLSpy automation interface a Java class exists with the name of the interface.

- ***Method names***
  Method names on the Java interface are the same as used on the COM interfaces but start with a small letter to conform to Java naming conventions. To access COM properties, Java methods that prefix the property name with `get` and `set` can be used. If a property does not support write-access, no setter method is available. Example: For the `Name` property of the `Document` interface, the Java methods `getName` and `setName` are available.

- ***Enumerations***
  For every enumeration defined in the automation interface, a Java enumeration is defined with the same name and values.

- ***Events and event handlers***
  For every interface in the automation interface that supports events, a Java interface with the same name plus '`Event`' is available. To simplify the overloading of single events, a Java class with default implementations for all events is provided. The name of this Java class is the name of the event interface plus '`DefaultHandler`'. For example:
  `Application`: Java class to access the application
  `ApplicationEvents`: Events interface for the Application
  `ApplicationEventsDefaultHandler`: Default handler for `ApplicationEvents`

Exceptions to mapping rules

There are some exceptions to the rules listed above. These are listed below:

Interface	Java name
`Document`, method `SetEncoding`	`setFileEncoding`
`AuthenticView`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Clone`	`cloneRange`

This section

This section explains how some basic XMLSpy functionality can be accessed from Java code. It is organized into the following sub-sections:

- Example Java Project 1569
- Application Startup and Shutdown 1572
- Simple Document Access 1573
- Iterations 1573
- Use of Out-Parameters 1574
- Event Handlers 1574

29.3.1.2.4.1 Example Java Project

The XMLSpy installation package contains an example Java project, located in the the `API\Java` subfolder of the `Examples` folder :

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\XMLSpy\2021\Examples`

This folder contains Java examples for the XMLSpy API. You can test it directly from the command line using the batch file `BuildAndRun.bat`, or you can compile and run the example project from within Eclipse. See below for instructions on how to use these procedures.

File list

The Java examples folder contains all the files required to run the example project. These files are listed below. If you are using a 64-bit version of the application, some filenames contain `_x64` in the name. These filenames are indicated with `(_x64)`.

`AltovaAutomation(_x64).dll`	Java-COM bridge: DLL part
`AltovaAutomation.jar`	Java-COM bridge: Java library part
`XMLSpyAPI.jar`	Java classes of the XMLSpy API
`RunXMLSpy.java`	Java example source code
`BuildAndRun.bat`	Batch file to compile and run example code from the command line prompt. Expects folder where Java Virtual Machine resides as parameter.
`.classpath`	Eclipse project helper file
`.project`	Eclipse project file
`XMLSpyAPI_JavaDoc.zip`	`Javadoc` file containing help documentation for the Java API

What the example does

The example starts up XMLSpy and performs a few operations, including opening and closing documents. When done, XMLSpy stays open. You must close it manually.

- Start XML Spy 1572: Starts XMLSpy, which is registered as an automation server, or activates XMLSpy if it is already running.
- Open OrgChart.pxf 1573: Locates one of the example documents installed with XMLSpy and opens it.
- Iteration and Changing the View Mode 1573: Changes the view of all open documents to Text View. The code also shows how to iterate through open documents.
- Iteration, validation, output parameters 1574: Validates the active document and shows the result in a message box. The code shows how to use output parameters.
- Event Handling 1574: Shows how to handle XMLSpy events.
- Shut down XMLSpy 1572: Shuts down XMLSpy.

You can modify the example in any way you like and run it.

Running the example from the command line

To run the example from the command line, open a command prompt window, go to the Java folder of the API Examples folder (*see above for location*), and then type:

```
buildAndRun.bat "<Path-to-the-Java-bin-folder>"
```

The Java binary folder must be that of a JDK 1.5 or later installation on your computer. Press the **Return** key. The Java source in `RunXMLSpy.java` will be compiled and then executed.

Loading the example in Eclipse

Open Eclipse and use the **Import | Existing Projects into Workspace** command to add the Eclipse project file (`.project`) located in the Java folder of the API Examples folder (*see above for location*). The project `RunXMLSpy` will then appear in your Package Explorer or Navigator. Select the project and then the command **Run as | Java Application** to execute the example.

Note: You can select a class name or method of the Java API and press F1 to get help for that class or method.

Java source code listing

The Java source code in the example file `RunXMLSpy.java` is listed below with comments.

```
001 // Access general JAVA-COM bridge classes
002 import com.altova.automation.libs.*;
003
004 // Access XMLSpy Java-COM bridge
005 import com.altova.automation.XMLSpy.*;
006 import com.altova.automation.XMLSpy.Enums.SPYViewModes;
007
008 /**
009  * An example that starts XMLSpy COM server and performs view operations on it
010  * Feel free to extend
011  */
012 public class RunXMLSpy
013 {
014   public static void main(String[] args)
015   {
016     // An instance of the application.
017     Application xmlSpy = null;
018
019     // Instead of COM error handling, use Java exception mechanism
020     try
021       {
022       // Start XMLSpy as COM server
023       xmlSpy = new Application();
024
025       // COM servers start up invisible, so make it visible
026       xmlSpy.setVisible(true);
027
028       // Locate samples installed with the product
029       String strExamplesFolder =
030       System.getenv("USERPROFILE") + "\\My Documents\\Altova\\XMLSpy2012\\Examples\
\";
031
032       // Open two example files
033       xmlSpy.getDocuments().openFile(strExamplesFolder + "OrgChart.pxf", false);
034       xmlSpy.getDocuments().openFile(strExamplesFolder + "ExpReport.xml", false);
035
036       // Iterate through open documents and set view mode to 'Text'.
037       for (Document doc:xmlSpy.getDocuments())
```


(C) 2015-2021 Altova GmbH

```
038       if ( doc.getCurrentViewMode() != SPYViewModes.spyViewText)
039           doc.switchViewMode(SPYViewModes.spyViewText);
040
041       // An alternative iteration mode is index-based
042       // COM indices are typically zero-based
043       Documents documents = xmlSpy.getDocuments();
044       for (int i = 1; i <= documents.getCount();
045           i++)
046           {
047             Document doc = documents.getItem(i);
048
049       // Validation is one of the few methods to have output parameters.
050       // The class JVariant is the correct type for parameters in these cases.
051       // To get values back mark them with the by-reference flag.
052       JVariant validationErrorText = new
053
054       JVariant.JStringVariant("");
055
056       validationErrorText.setByRefFlag();
057       JVariant validationErrorCount = new
058
059       JVariant.JIntVariant(0);
060
061       validationErrorCount.setByRefFlag();
062       JVariant validationErrorXMLData = new
063
064       JVariant.JIDispatchVariant(0);
065
066       validationErrorXMLData.setByRefFlag();
067       if (!doc.isValid(validationErrorText, validationErrorCount,
validationErrorXMLData))
068           System.out.println("Document" + doc.getName() + " is not wellformed - " +
validationErrorText.getStringValue());
069         else
070           System.out.println("Document" + doc.getName() + " is wellformed.");
071       }
072
073       // The following lines attach to the document events using a default
implementation
074       // for the events and override one of its methods.
075       // If you want to override all document events it is better to derive your
listener class
076       // from DocumentEvents and implement all methods of this interface.
077       Document doc = xmlSpy.getActiveDocument();
078       doc.addListener(new
079
080 DocumentEventsDefaultHandler()
081       {
082         @Override
083         public boolean
084
085 onBeforeCloseDocument(Document i_ipDoc) throws AutomationException
086         {
087           System.out.println("Document
088
089 " + i_ipDoc.getName() + " requested closing.");
090
091           // Allow closing of document
092           return true;
093         }
094       });
095       doc.close(true);
```

```
096       doc = null;
097
098       System.out.println("Watch XMLSpy!");
099     }
100     catch (AutomationException e)
101     {
102       // e.printStackTrace();
103     }
104     finally
105     {
106       // Make sure that XMLSpy can shut down properly.
107       if (xmlSpy != null)
108           xmlSpy.dispose();
109
110       // Since the COM server was made visible and still is visible,
111       // it will keep running, and needs to be closed manually.
112       System.out.println("Now close XMLSpy!");
113     }
114   }
115 }
```

29.3.1.2.4.2 Application Startup and Shutdown

The code listings below show how the application can be started up and shut down.

Application startup

Before starting up the application, the appropriate classes must be imported (*see below*).

```
01 // Access general JAVA-COM bridge classes
02 import com.altova.automation.libs.*;
03
04 // Access XMLSpy Java-COM bridge
05 import com.altova.automation.XMLSpy.*;
06 import com.altova.automation.XMLSpy.Enums.SPYViewModes;
07
08 /**
09  * An example that starts XMLSpy COM server and performs view operations on it
10  * Feel free to extend
11  */
12 public class RunXMLSpy
13 {
14   public static void main(String[] args)
15   {
16     // An instance of the application.
17     Application xmlSpy = null;
18
19     // Instead of COM error handling, use Java exception mechanism
20     try
21     {
22         // Start XMLSpy as COM server
23           xmlSpy = new Application();
24         // COM servers start up invisible, so make it visible
25           xmlSpy.setVisible(true);
26
27 ...
28     }
29   }
```

```
30 }
```

Application shutdown

The application can be shut down as shown below.

```
01 {
02       // Allow shutdown of XMLSpy by releasing the UI.
03       xmlSpy.setVisible(true);
04
05       // Make sure that XMLSpy can shut down properly.
06       if (xmlSpy != null)
07         xmlSpy.dispose();
08
09       // Since the COM server was made visible and still is visible,
10       // it will keep running, and needs to be closed manually.
11       System.out.println("Now close XMLSpy!");
12 }
```

29.3.1.2.4.3 Simple Document Access

The code listing below shows how to open a document.

```
1  // Locate samples installed with the product
2  String strExamplesFolder =
3  System.getenv("USERPROFILE") + "\\My Documents\\Altova\\XMLSpy2012\\Examples\\";
4
5
6  // Open file
7  xmlSpy.getDocuments().openFile(strExamplesFolder + "OrgChart.pxf", false);
```

29.3.1.2.4.4 Iterations

The listing below shows how to iterate through open documents.

```
01  // Iterate through open documents and set view mode to 'Text'.
02  for (Document doc:xmlSpy.getDocuments())
03    if ( doc.getCurrentViewMode() != SPYViewModes.spyViewText)
04              doc.switchViewMode(SPYViewModes.spyViewText);
05
06  // An alternative iteration mode is index-based
07  // COM indices are typically zero-based
08  Documents documents = xmlSpy.getDocuments();
09     for (int i = 1; i <= documents.getCount();
10         i++)
11       {
12         Document doc = documents.getItem(i);
13               ...
14       }
```

29.3.1.2.4.5 Use of Out-Parameters

The code listing below iterates through open documents and validates each of them. For each validation, a message is generated using the output parameters of the Validation method.

```
01 // An alternative iteration mode is index-based
02 // COM indices are typically zero-based
03 Documents documents = xmlSpy.getDocuments();
04 for (int i = 1; i <= documents.getCount();
05       i++)
06     {
07         Document doc = documents.getItem(i);
08
09 // Validation is one of the few methods to have output parameters.
10 // The class JVariant is the correct type for parameters in these cases.
11 // To get values back mark them with the by-reference flag.
12 JVariant validationErrorText = new
13
14 JVariant.JStringVariant("");
15
16 validationErrorText.setByRefFlag();
17       JVariant validationErrorCount = new
18
19 JVariant.JIntVariant(0);
20
21 validationErrorCount.setByRefFlag();
22       JVariant validationErrorXMLData = new
23
24 JVariant.JIDispatchVariant(0);
25
26 validationErrorXMLData.setByRefFlag();
27       if (!doc.isValid(validationErrorText,
28
29         validationErrorCount, validationErrorXMLData))
30         System.out.println("Document
31
32         " + doc.getName() + " is not wellformed - " +
33
34         validationErrorText.getStringValue());
35       else
36         System.out.println("Document
37
38         " + doc.getName() + " is wellformed.");
39     }
```

29.3.1.2.4.6 Event Handlers

The listing below shows how to listen for and use events.

```
01 // The following lines attach to the document events using a default implementation
02 // for the events and override one of its methods.
03 // If you want to override all document events it is better to derive your listener
class
04 // from DocumentEvents and implement all methods of this interface.
```

```
05
06 Document doc = xmlSpy.getActiveDocument();
07 doc.addListener(new DocumentEventsDefaultHandler()
08     {
09       @Override
10       public boolean
11       onBeforeCloseDocument(Document i_ipDoc) throws AutomationException
12       {
13         System.out.println("Document " + i_ipDoc.getName() + " requested closing.");
14
15         // Allow closing of document
16         return true;
17       }
18     });
19 doc.close(true);
20 doc = null;
```

29.3.1.3 The DOM and XMLData

The `XMLData` interface gives you full access to the XML structure behind the current document with less methods than DOM and is much simpler. The `XMLData` interface is a minimalist approach to reading and modifying existing, or newly created XML data. You might however, want to use a DOM tree because you can access one from an external source or you just prefer the MSXML DOM implementation.

The `ProcessDOMNode()` and `ProcessXMLDataNode()` functions provided below convert any segments of an XML structure between XMLData and DOM.

To use the `ProcessDOMNode()` function:

- pass the root element of the DOM segment you want to convert in objNode and
- pass the plugin object with the CreateChild() method in objCreator

To use the `ProcessXMLDataNode()` function:

- pass the root element of the XMLData segment in objXMLData and
- pass the DOMDocument object created with MSXML in xmlDoc

```
////////////////////////////////////////////////////////////
// DOM To XMLData conversion
Function ProcessDOMNode(objNode,objCreator)
{
       var objRoot;
       objRoot = CreateXMLDataFromDOMNode(objNode,objCreator);

       If(objRoot)   {
              If((objNode.nodeValue != Null) && (objNode.nodeValue.length > 0))
                     objRoot.TextValue = objNode.nodeValue;
                     // add attributes
              If(objNode.attributes) {
                     var Attribute;
                     var oNodeList = objNode.attributes;

                     For(var i = 0;i < oNodeList.length; i++) {
                            Attribute = oNodeList.item(i);

                            var newNode;
                            newNode = ProcessDOMNode(Attribute,objCreator);
```

```
                        objRoot.AppendChild(newNode);
                }
        }
        If(objNode.hasChildNodes)  {
                try {
                        // add children
                        var Item;
                        oNodeList = objNode.childNodes;

                        For(var i = 0;i < oNodeList.length; i++) {
                                Item = oNodeList.item(i);

                                var newNode;
                                newNode = ProcessDOMNode(Item,objCreator);

                                objRoot.AppendChild(newNode);
                        }
                }
                catch(err) {
                }
        }
    }
    Return objRoot;
}

Function CreateXMLDataFromDOMNode(objNode,objCreator)
{
    var bSetName = True;
    var bSetValue = True;

    var nKind = 4;

    switch(objNode.nodeType)   {
        Case 2:nKind = 5;break;
        Case 3:nKind = 6;bSetName = False;break;
        Case 4:nKind = 7;bSetName = False;break;
        Case 8:nKind = 8;bSetName = False;break;
        Case 7:nKind = 9;break;
    }
    var objNew = Null;
    objNew = objCreator.CreateChild(nKind);

    If(bSetName)
        objNew.Name = objNode.nodeName;

    If(bSetValue && (objNode.nodeValue != Null))
        objNew.TextValue = objNode.nodeValue;

    Return objNew;
}
////////////////////////////////////////////////////////////
// XMLData To DOM conversion

Function ProcessXMLDataNode(objXMLData,xmlDoc)
{
    var objRoot;
    objRoot = CreateDOMNodeFromXMLData(objXMLData,xmlDoc);

    If(objRoot)   {
        If(IsTextNodeEnabled(objRoot) && (objXMLData.TextValue.length > 0))
            objRoot.appendChild(xmlDoc.createTextNode(objXMLData.TextValue));
```

```
		If(objXMLData.HasChildren) {
			try {
				var objChild;
				objChild = objXMLData.GetFirstChild(-1);

				While(True)	{
					If(objChild)	{
						var newNode;
						newNode = ProcessXMLDataNode(objChild,xmlDoc);

						If(newNode.nodeType == 2)	{
							// child node is an attribute
							objRoot.attributes.setNamedItem(newNode);
						}
						Else
							objRoot.appendChild(newNode);
					}
					objChild = objXMLData.GetNextChild();
				}
			}
			catch(err) {
			}
		}
	}
	Return objRoot;
}

Function CreateDOMNodeFromXMLData(objXMLData,xmlDoc)
{
	switch(objXMLData.Kind)	{
		Case 4:Return xmlDoc.createElement(objXMLData.Name);
		Case 5:Return xmlDoc.createAttribute(objXMLData.Name);
		Case 6:Return xmlDoc.createTextNode(objXMLData.TextValue);
		Case 7:Return xmlDoc.createCDATASection(objXMLData.TextValue);
		Case 8:Return xmlDoc.createComment(objXMLData.TextValue);
		Case 9:Return
xmlDoc.createProcessingInstruction(objXMLData.Name,objXMLData.TextValue);
	}

	Return xmlDoc.createElement(objXMLData.Name);
}
Function IsTextNodeEnabled(objNode)
{
	switch(objNode.nodeType) {
		Case 1:
		Case 2:
		Case 5:
		Case 6:
		Case 11:Return True;
	}
	Return False;
}
```

29.3.1.4 Obsolete: Authentic View Row operations

If the schema on which an XML document is based specifies that an element is repeatable, such a structure can be represented in Authentic View as a table. When represented as a table, rows and their contents can be

manipulated individually, thereby allowing you to manipulate each of the repeatable elements individually. Such row operations would be performed by an external script.

If an external script is to perform row operations then two steps must occur:

- The first step checks whether the cursor is currently in a row using a property. Such a check could be, for example, `IsRowInsertEnabled`, which returns a value of either `TRUE` or `FALSE`.
- If the return value is `TRUE` then a row method, such as `RowAppend`, can be called. (`RowAppend` has no parameters and returns no value.)

The following is a list of properties and methods available for table operations. Each property returns a `BOOL`, and the methods have no parameter.

Property	Method	Table operations
IsRowInsertEnabled	**RowInsert** 1840, superseded by `AuthenticRange.InsertRow` 1621	Insert row operation
IsRowAppendEnabled	**RowAppend** 1838, superseded by `AuthenticRange.AppendRow` 1607	Append row operation
IsRowDeleteEnabled	**RowDelete** 1839, superseded by `AuthenticRange.DeleteRow` 1611	Delete row operation
IsRowMoveUpEnabled	**RowMoveUp** 1841, superseded by `AuthenticRange.MoveRowUp` 1628	Move XML data up one row
IsRowMoveDownEnabled	**RowMoveDown** 1840, superseded by `AuthenticRange.MoveRowDown` 1628	Move XML data down one row
IsRowDuplicateEnabled	**RowDuplicate** 1839, superseded by `AuthenticRange.DuplicateRow` 1611	Duplicate currently selected row

29.3.1.5 Obsolete: Authentic View Editing operations

When XML data is displayed as data in Authentic View, it is possible to manipulate individual elements using standard editing operations such as cut, copy, and paste. However, not all XML data nodes can be edited. So, in order to carry out an editing operation, first a property is used to test whether editing is possible, and then a method is called to perform the editing operation.

The only method that does not have a test is the method `EditSelectAll`, which automatically selects all elements displayed in the document.

The following is a list of properties and methods that perform editing operations. Each property returns a `BOOL`, and the methods have no parameter.

Property	Method	Editing operation
IsEditUndoEnabled	**EditUndo** 1826, superseded by `AuthenticView.Undo` 1651	Undo an editing operation
IsEditRedoEnabled	**EditRedo** 1826, superseded by `AuthenticView.Redo` 1650	Redo an editing operation

(C) 2015-2021 Altova GmbH

IsEditCopyEnabled	**EditCopy**[1824], superseded by AuthenticRange.Copy[1610]	Copy selected text to the clipboard
IsEditCutEnabled	**EditCut**[1825], superseded by AuthenticRange.Cut[1610]	Cut selected text to the clipboard
IsEditPasteEnabled	**EditPaste**[1825], superseded by AuthenticRange.Paste[1628]	Paste from clipboard to current cursor position
IsEditClearEnabled	**EditClear**[1824], superseded by AuthenticRange.Delete[1610]	Clear selected text from XML document

29.3.2 Interfaces

Object Hierarchy

Application[1580]
- SpyProject[1747]
  - SpyProjectItems[1751]
    - SpyProjectItem[1748]
  - Documents[1698]
    - Document[1668]
      - GridView[1730]
      - AuthenticView[1634]
        - AuthenticRange[1606]
        - AuthenticDataTransfer[1601] (previously DocEditDataTransfer)
      - OldAuthenticView[1821] (previously DocEditView, **now obsolete**, superseded by AuthenticView[1634] and AuthenticRange[1606])
        - AuthenticSelection[1819] (previously DocEditSelection, **now obsolete**, superseded by AuthenticRange[1606])
        - AuthenticEvent[1806] (previously DocEditEvent, **now obsolete**)
        - AuthenticDataTransfer[1601] (previously DocEditDataTransfer)
      - TextView[1755]
      - XMLData[1795]
  - Dialogs[1585]
    - CodeGeneratorDlg[1652]
    - FileSelectionDlg[1711]
    - SchemaDocumentationDlg[1733]
    - GenerateSampleXMLDlg[1724]
    - DTDSchemaGeneratorDlg[1702]
    - FindInFilesDlg[1712]
    - WSDLDocumentationDlg[1761]
    - WSDL20DocumentationDlg[1772]
    - XBRLDocumentationDlg[1784]
  - DatabaseConnection[1658]
  - ExportSettings[1708]
  - TextImportExportSettings[1753]
  - ElementList[1706]
    - ElementListItem[1707]

Enumerations[1843]

Description
This chapter contains the reference of the XMLSpy 1.5 Type Library.

Most of the given examples are written in VisualBasic. These code snippets assume that there is a variable defined and set, called **objSpy of type Application**. There are also some code samples written in JavaScript.

29.3.2.1 Application

Methods
GetDatabaseImportElementList 1586
GetDatabaseSettings 1587
GetDatabaseTables 1587
ImportFromDatabase 1590
CreateXMLSchemaFromDBStructure 1585

GetTextImportElementList 1588
GetTextImportExportSettings 1589
ImportFromText 1592

ImportFromWord 1593

ImportFromSchema 1591

GetExportSettings 1588

NewProject 1594
OpenProject 1594

AddMacroMenuItem 1583
ClearMacroMenu 1584

ShowForm 1597

ShowApplication 1597

URLDelete 1598
URLMakeDirectory 1598

AddXSLT_XQParameter 1584
GetXSLT_XQParameterCount 1589
GetXSLT_XQParameterName 1590
GetXSLT_XQParameterXPath 1590
RemoveXSLT_XQParameter 1595

FindInFiles 1586

Quit 1595

Properties
Application 1584
Parent 1594

ActiveDocument 1583

(C) 2015-2021 Altova GmbH

Documents 1585

CurrentProject 1585

Dialogs 1585

WarningNumber 1599
WarningText 1599

Status 1598
MajorVersion 1593
MinorVersion 1593
Edition 1586
IsAPISupported 1593
ServicePackVersion 1596

Description
Application is the root for all other objects. It is the only object you can create by CreateObject (VisualBasic) or other similar COM related functions.

Example

```
Dim objSpy As Application
Set objSpy = CreateObject("XMLSpy.Application")
```

29.3.2.1.1 Events

29.3.2.1.1.1 OnBeforeOpenDocument

Event: OnBeforeOpenDocument(*objDialog* as FileSelectionDlg 1711)

Description
This event gets fired whenever a document gets opened via the OpenFile or OpenURL menu command. It is sent after a document file has been selected but before the document gets opened. The file selection dialog object is initialized with the name of the selected document file. You can modify this selection. To continue the opening of the document leave the FileSelectionDlg.DialogAction 1711 property of *io_objDialog* at its default value spyDialogOK 1846. To abort the opening of the document set this property to spyDialogCancel 1846.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:

```
Function On_BeforeOpenDocument(objDialog)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeOpenDocument(objDialog)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (26, ...)// nEventId = 26
```

29.3.2.1.1.2 OnBeforeOpenProject

Event: OnBeforeOpenProject(*objDialog* as FileSelectionDlg[1711])

Description
This event gets fired after a project file has been selected but before the project gets opened. The file selection dialog object is initialized with the name of the selected project file. You can modify this selection. To continue the opening of the project leave the FileSelectionDlg.DialogAction[1711] property of *io_objDialog* at its default value spyDialogOK[1846]. To abort the opening of the project set this property to spyDialogCancel[1846].

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_BeforeOpenProject(objDialog)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeOpenProject(objDialog)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (25, ...)// nEventId = 25
```

29.3.2.1.1.3 OnDocumentOpened

Event: OnDocumentOpened(*objDocument* as Document[1668])

Description
This event gets fired whenever a document opens in XMLSpy. This can happen due to opening a file with the OpenFile or OpenURL dialog, creating a new file or dropping a file onto XMLSpy. The new document gets passed as parameter. The operation cannot be canceled.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_OpenDocument(objDocument)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_OpenDocument(objDocument)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (7, ...)  // nEventId = 7
```


(C) 2015-2021 Altova GmbH

29.3.2.1.1.4 OnProjectOpened

Event: OnProjectOpened(*objProject* as SpyProject[1747])

Description
This event gets fired whenever a project gets opened in XMLSpy. The new project gets passed as parameter.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_OpenProject(objProject)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_OpenProject(objProject)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (6, ...)  // nEventId = 6
```

29.3.2.1.2 ActiveDocument

Property: ActiveDocument as Document[1668]

Description
Reference to the active document. If no document is open, ActiveDocument is null (nothing).

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.3 AddMacroMenuItem

Method: AddMacroMenuItem(*strMacro* as String,*strDisplayText* as String)

Description
Adds a menu item to the **Tools** menu. This new menu item invokes the macro defined by strMacro. See also Example Scripting Project[1516].

Errors
1111 The application object is no longer valid.

1100 Invalid parameter or invalid address for the return parameter was specified.
1108 Number of macro items is limited to 16 items.

29.3.2.1.4 AddXSLT_XQParameter

Method: AddXSLT_XQParameter(name as String, XPath as String)

Description
Adds an XSLT or XQuery parameter. The parameter's name and value are the two arguments of the method.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.
1124 The XPath expression is not set.
1125 Not a QName.
1126 The specified XPath is not valid. Reason for invalidity appended.
1127 A parameter with the submitted name already exists.

29.3.2.1.5 Application

Property: Application as Application[1580] (read-only)

Description
Accesses the XMLSpy application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.6 ClearMacroMenu

Method: ClearMacroMenu()

Return Value
None

Description
Removes from the **Tools** menu those menu items that were added by calling AddMacroMenuItem[1583]. See also Example Scripting Project[1516].

Errors
1111 The application object is no longer valid.

(C) 2015-2021 Altova GmbH

29.3.2.1.7 CreateXMLSchemaFromDBStructure

Method: CreateXMLSchemaFromDBStructure(pImportSettings as DatabaseConnection[1658], pTables as ElementList[1706])

Description
CreateXMLSchemaFromDBStructure creates from a database specified in pImportSettings for the defined tables in pTables new XML Schema document(s) describing the database tables structure.

The parameter pTables specifies which table structures the XML Schema document should contain. This parameter can be NULL, specifying that all table structures will be exported.

See also GetDataBaseTables[1587].

Errors
1112 Invalid database specified.
1120 Database import failed.

29.3.2.1.8 CurrentProject

Property: CurrentProject as SpyProject[1747]

Description
Reference to the active document. If no project is open, CurrentProject is null (nothing).

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.9 Dialogs

Property: Dialogs as Dialogs[1664] (read-only)

Description
Access the built-in dialogs of XMLSpy.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.10 Documents

Property: Documents as Documents[1698]

Description

Collection of all open documents.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.11 Edition

Property: Edition as String

Description
Returns the edition of the application, for example `Altova XMLSpy Enterprise Edition` for the Enterprise edition.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.12 FindInFiles

Method: FindInFiles(*pSettings* as FindInFilesDlg[1712]) as FindInFilesResults[1723]

Description
Returns a FindInFilesResults[1723] object containing information about the files that matched the specified settings.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.13 GetDatabaseImportElementList

Method: GetDatabaseImportElementList(*pImportSettings* as DatabaseConnection[1658]) as ElementList[1706]

Description
The function returns a collection of ElementListItems where the properties ElementListItem.Name[1708] contain the names of the fields that can be selected for import and the properties ElementListItem.ElementKind[1707] are initialized either to *spyXMLDataAttr* or *spyXMLDataElement*, depending on the value passed in DatabaseConnection.AsAttributes[1659]. This list serves as a filter to what finally gets imported by a future call to ImportFromDatabase[1590]. Use ElementList.RemoveElement[1707] to exclude fields from import.

Properties mandatory to be filled out for the database connection are one of DatabaseConnection.File[1661], DatabaseConnection.ADOConnection[1659] and DatabaseConnection.ODBCConnection[1662], as well as DatabaseConnection.SQLSelect[1663]. Use the property DatabaseConnection.AsAttributes[1659] to initialize ElementListItem.ElementKind[1707] of the resulting element list to either *spyXMLDataAttr* or *spyXMLDataElement*, respectively.

(C) 2015-2021 Altova GmbH

Example
See example at ImportFromDatabase[1590].

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from database failed.
1112 Invalid database specified.
1114 Select statement is missing.
1119 database element list import failed.

29.3.2.1.14 GetDatabaseSettings

Method: GetDatabaseSettings() as DatabaseConnection[1658]

Description
GetDatabaseSettings creates a new object of database settings. The object is used to specify database connection parameters for the methods GetDatabaseTables[1587], GetDatabaseImportElementList[1586], ImportFromDatabase[1590], ImportFromSchema[1591] and ExportToDatabase[1680].

Example
See example of ImportFromDatabase[1590].

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.15 GetDatabaseTables

Method: GetDatabaseTables(*pImportSettings* as DatabaseConnection[1580]) as ElementList[1706]

Description
GetDatabaseTables reads the table names from the database specified in *pImportSettings*. Properties mandatory to be filled out for the database connection are one of DatabaseConnection.File[1661], DatabaseConnection.ADOConnection[1659] and DatabaseConnection.ODBCConnection[1662]. All other properties are ignored.
The function returns a collection of ElementListItems where the properties ElementListItem.Name[1708] contain the names of tables stored in the specified database. The remaining properties of ElementListItem[1707] are unused.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1112 Invalid database specified.
1113 Error while reading database table information.
1118 Database table query failed.

Example

```
        Dim objImpSettings As DatabaseConnection
Set objImpSettings = objSpy.GetDatabaseSettings
objImpSettings.ADOConnection = TxtADO.Text

'store table names in list box
ListTables.Clear

Dim objList As ElementList
Dim objItem As ElementListItem
On Error GoTo ErrorHandler
Set objList = objSpy.GetDatabaseTables(objImpSettings)

        For Each objItem In objList
                ListTables.AddItem objItem.Name
        Next
```

29.3.2.1.16 GetExportSettings

Method: GetExportSettings()as ExportSettings[1708] (read-only)

Description
GetExportSettings creates a new object of common export settings. This object is used to pass the parameters to the export functions and defines the behaviour of the export calls. See also the export functions from Document[1668] and the examples at Import and Export[1552].

Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

29.3.2.1.17 GetTextImportElementList

Method: GetTextImportElementList(pImportSettings as TextImportExportSettings[1753]) as ElementList[1706]

Description
GetTextImportElementList retrieves importing information about the text-file as specified in pImportSettings. The function returns a collection of ElementListItems where the properties ElementListItem.Name[1708] contain the names of the fields found in the file. The values of remaining properties are undefined.

If the text-file does not contain a column header, set `pImportSettings.HeaderRow`[1754] to `false`. The resulting element list will contain general column names like 'Field1' and so on.

See also Import and export of data[1552].

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1107	Import from database failed.

(C) 2015-2021 Altova GmbH

1115 Error during text element list import. Cannot create parser for import file.
1116 Error during text element list import.

Example

```
' ---------------------------------------------------------
' VBA client code fragment - import selected fields from text file
' ---------------------------------------------------------
        Dim objImpSettings As TextImportExportSettings
        Set objImpSettings = objSpy.GetTextImportExportSettings

        objImpSettings.ImportFile = "C:\ImportMe.txt"
        objImpSettings.HeaderRow = False

        Dim objList As ElementList
        Set objList = objSpy.GetTextImportElementList(objImpSettings)

        'exclude first column
        objList.RemoveItem 1

        Dim objImpDoc As Document
        On Error Resume Next
        Set objImpDoc = objSpy.ImportFromText(objImpSettings, objList)
        CheckForError
```

29.3.2.1.18 GetTextImportExportSettings

Method: GetTextImportExportSettings() as TextImportExportSettings[1753] (read-only)

Description
GetTextImportExportSettings creates a new object of common import and export settings for text files. See also the example for Application.GetTextImportElementList[1588] and Import and Export[1552].

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.19 GetXSLT_XQParameterCount

Method: GetXSLT_XQParameterCount() as Long

Description
Returns the number of XSLT and XQuery parameters.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.20 GetXSLT_XQParameterName

Method: GetXSLT_XQParameterName(index as Long) as String

Description
Returns the name of the XSLT or XQuery parameter identified by the supplied index.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.21 GetXSLT_XQParameterXPath

Method: GetXSLT_XQParameterXPath(index as Long) as String

Description
Returns the XPath expression of the XSLT or XQuery parameter identified by the supplied index.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.22 ImportFromDatabase

Method: ImportFromDatabase(pImportSettings as DatabaseConnection[1658],pElementList as ElementList[1706]) as Document[1668]

Return Value
Creates a new document containing the data imported from the database.

Description
ImportFromDatabase imports data from a database as specified in pImportSettings and creates a new document containing the data imported from the database. Properties mandatory to be filled out are one of DatabaseConnection.File[1661], DatabaseConnection.ADOConnection[1659] or DatabaseConnection.ODBCConnection[1662] and DatabaseConnection.SQLSelect[1663]. Additionally, you can use DatabaseConnection.AsAttributes[1659], DatabaseConnection.ExcludeKeys[1661], DatabaseConnection.IncludeEmptyElements[1662] and NumberDateTimeFormat[1662] to further parameterize import.

The parameter pElementList specifies which fields of the selected data gets written into the newly created document, and which are created as elements and which as attributes. This parameter can be NULL, specifying that all selected fields will be imported as XML elements.

See GetDatabaseSettings[1587] and GetDatabaseImportElementList[1586] for necessary steps preceding any import of data from a database.

(C) 2015-2021 Altova GmbH

Errors

1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from database failed.
1112 Invalid database specified.
1114 Select statement is missing.
1117 Transformation to XML failed.
1120 Database import failed.

Example

```
        Dim objImpSettings As DatabaseConnection
        Set objImpSettings = objSpy.GetDatabaseSettings

        objImpSettings.ADOConnection = strADOConnection
        objImpSettings.SQLSelect = "SELECT * FROM MyTable"

        Dim objDoc As Document
        On Error Resume Next
        Set objDoc = objSpy.ImportFromDatabase(objImpSettings,
objSpy.GetDatabaseImportElementList(objImpSettings))
        ' CheckForError here
```

29.3.2.1.23 ImportFromSchema

Method: ImportFromSchema(*pImportSettings* as DatabaseConnection[1658],*strTable* as String,*pSchemaDoc* as Document[1668]) as Document[1668]

Return Value

Creates a new document filled with data from the specified database as specified by the schema definition in *pSchemaDoc*.

Description

ImportFromSchema imports data from a database specified in pImportSettings. Properties mandatory to be filled out are one of DatabaseConnection.File[1661], DatabaseConnection.ADOConnection[1659] or DatabaseConnection.ODBCConnection[1662]. Additionally, you can use DatabaseConnection.AsAttributes[1659], DatabaseConnection.ExcludeKeys[1661] and NumberDateTimeFormat[1662] to further parameterize import. All other properties get ignored.

ImportFromSchema does not use an explicit SQL statement to select the data. Instead, it expects a structure definition of the document to create in form of an XML schema document in *pSchemaDoc*. From this definition the database select statement is automatically deduced. Specify in *strTable* the table name of the import root that will become the root node in the new document.

See GetDatabaseSettings[1587] and GetDatabaseTables[1587] for necessary steps preceding an import from a database based on a schema definition. To create the schema definition file use command 'create database schema' from the 'convert' menu of XMLSpy.

Errors

1111 The application object is no longer valid.

1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from database failed.
1112 Invalid database specified.
1120 Database import failed.
1121 Could not create validator for the specified schema.
1122 Failed parsing schema for database import.

29.3.2.1.24 ImportFromText

Method: ImportFromText(*pImportSettings* as TextImportExportSettings[1753],*pElementList* as ElementList[1706]) as Document[1668]

Description
ImportFromText imports the text file as specified in pImportSettings. The parameter pElementList can be used as import filter. Either pass the list returned by a previous call to GetTextImportElementList[1588] or null to import all columns. To avoid import of unnecessary columns use ElementList.RemoveElement[1707] to remove the corresponding field names from `pElementList` before calling `ImportFromText`.
The method returns the newly created document containing the imported data. This document is the same as the active document of XMLSpy.

See also Import and export of data[1552].

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1107 Import from text file failed.
1117 Transformation to XML failed.

Example

```
' ---------------------------------------------------------
' VBA client code fragment - import from text file
' ---------------------------------------------------------
        Dim objImpSettings As TextImportExportSettings
        Set objImpSettings = objSpy.GetTextImportExportSettings

        objImpSettings.ImportFile = strFileName
        objImpSettings.HeaderRow = False

        Dim objImpDoc As Document
        On Error Resume Next
        Set objImpDoc = objSpy.ImportFromText(objImpSettings,
objSpy.GetTextImportElementList(objImpSettings))

        CheckForError
```

29.3.2.1.25 ImportFromWord

Method: ImportFromWord(*strFile* as String) as Document[1668]

Description
ImportFromWord imports the MS-Word Document strFile into a new XML document.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
Import from document failed.

29.3.2.1.26 IsAPISupported

Property: IsAPISupported as Boolean

Description
Returns whether the API is supported in this version or not.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.27 MajorVersion

Property: MajorVersion as Integer

Description
Returns the application version's major number, for example `15` for 2013 versions, and `16` for 2014 versions..

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.28 MinorVersion

Property: MinorVersion as Integer

Description
Returns the application version's minor number.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.29 NewProject

Method: NewProject(*strPath* as String,*bDiscardCurrent* as Boolean)

Description
NewProject creates a new project.

If there is already a project open that has been modified and bDiscardCurrent is false, then NewProject() fails.

Errors
1111 The application object is no longer valid.
1102 A project is already open but *bDiscardCurrent* is *true*.
1103 Creation of new project failed.

29.3.2.1.30 OpenProject

Method: OpenProject(*strPath* as String,*bDiscardCurrent* as Boolean,*bDialog* as Boolean)

Parameters
strPath
Path and file name of the project to open. Can be empty if bDialog is true.

bDiscardCurrent
Discard currently open project and possibly lose changes.

bDialog
Show dialogs for user input.

Return Value
None

Description
OpenProject opens an existing project. If there is already a project open that has been modified and bDiscardCurrent is false, then OpenProject() fails.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.
1101 Cannot open specified project.
1102 A project is already open but *bDiscardCurrent* is *true*.

29.3.2.1.31 Parent

Property: Parent as Application 1580 (read-only)

Description

Accesses the XMLSpy application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.32 Quit

Method: Quit()

Return Value
None

Description
This method terminates XMLSpy. All modified documents will be closed without saving the changes. This is also true for an open project.

If XMLSpy was automatically started as an automation server by a client program, the application will not shut down automatically when your client program shuts down if a project or any document is still open. Use the Quit method to ensure automatic shut-down.

Errors
1111 The application object is no longer valid.

29.3.2.1.33 ReloadSettings

Method: ReloadSettings

Return Value

Description
The application settings are reloaded from the registry.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.

29.3.2.1.34 RemoveXSLT_XQParameter

Method: RemoveXSLT_XQParameter(index as Long)

Description
Removes the XSLT or XQuery parameter identified by the supplied index.

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.35 RunMacro

Method: RunMacro(*strMacro* as String)

Return Value

Description
Calls the specified macro either from the project scripts (if present) or from the global scripts.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.

29.3.2.1.36 ScriptingEnvironment

Property: ScriptingEnvironment as IUnknown (read-only)

Description
Reference to any active scripting environment. This property makes it possible to access the TypeLibrary of the XMLSpyFormEditor.exe application which is used as the current scripting environment.

Available with TypeLibrary version 1.5

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.37 ServicePackVersion

Property: ServicePackVersion as Long

Description
Returns the Service Pack version number of the application. Eg: 1 for 2010 R2 SP1

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.1.38 ShowApplication

Method: ShowApplication(*bShow* as Boolean)

Return Value
None

Description
The method shows (bShow = True) or hides (bShow = False) XMLSpy.

Errors
1110 The application object is no longer valid.

29.3.2.1.39 ShowFindInFiles

Method: ShowFindInFiles(pSettings as FindInFilesDlg[1712]) as Boolean

Return Value
Returns false if the user pressed the Cancel button, true otherwise.

Description
Displays the FindInFiles dialog preset with the given settings. The user modifications of the settings are stored in the passed dialog object.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.

29.3.2.1.40 ShowForm

Method: ShowForm(*strFormName* as String) as Long

Return Value
Returns zero if the user pressed a Cancel button or the form calls TheView.Cancel().

Description
Displays the form strFormName.

Forms, event handlers and macros can be created with the Scripting Environment. Select "Switch to scripting environment" from the **Tools** menu to invoke the Scripting Environment.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.

29.3.2.1.41 Status

Property: Status as ENUMApplicationStatus 1844

Description
Returns the current status of the running application.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.42 URLDelete

Method: URLDelete(*strURL* as String,*strUser* as String,*strPassword* as String)

Return Value
None

Description
The method deletes the file at the URL strURL.

Errors
1111 The application object is no longer valid.
1109 Error deleting file at specified URL.

29.3.2.1.43 URLMakeDirectory

Method: URLMakeDirectory(*strURL* as String,*strUser* as String,*strPassword* as String)

Return Value
None

Description
The method creates a new directory at the URL strURL.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter specified.

29.3.2.1.44 Visible

Property: Visible as VARIANT_BOOL

Description
Sets or gets the visibility attribute of XMLSpy. This standard automation property makes usage of ShowApplication 1597 obsolete.

(C) 2015-2021 Altova GmbH

Errors
1110 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.45 WarningNumber

Property: WarningNumber as integer

Description
Some methods fill the property WarningNumber with additional information if an error occurs.

Currently just Documents.OpenFile[1700] fills this property.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.1.46 WarningText

Property: WarningText as String

Description
Some methods fill the property WarningText with additional information if an error occurs.

Currently just Documents.OpenFile[1700] fills this property.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

29.3.2.2 AuthenticContextMenu

The context menu interface provides the means for the user to customize the context menus shown in Authentic. The interface has the methods listed in this section.

29.3.2.2.1 CountItems

Method: CountItems() nItems as long

Return Value
Returns the number of menu items.

Errors
2501 Invalid object.

29.3.2.2.2 DeleteItem

Method: DeleteItem`(IndexPosition as long)`

Return Value
Deletes the menu item that has the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

29.3.2.2.3 GetItemText

Method: GetItemText`(IndexPosition as long)` MenuItemName as string

Return Value
Gets the name of the menu item located at the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

29.3.2.2.4 InsertItem

Method: InsertItem`(IndexPosition as long, MenuItemName as string, MacroName as string)`

Return Value
Inserts a user-defined menu item at the position in the menu specified in the first parameter and having the name submitted in the second parameter. The menu item will start a macro, so a valid macro name must be submitted.

Errors
2501 Invalid object
2502 Invalid index
2503 No such macro
2504 Internal error

(C) 2015-2021 Altova GmbH

29.3.2.2.5 SetItemText

Method: SetItemText`(IndexPosition as long, MenuItemName as string)`

Return Value
Sets the name of the menu item located at the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

29.3.2.3 AuthenticDataTransfer

Renamed from DocEditDataTransfer to AuthenticDataTransfer

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage—except for AuthenticDataTransfer—is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interfaces. All functionality available up to now in DocEditView[1821], DocEditSelection[1819], DocEditEvent[1806] and DocEditDataTransfer[1602] is now available via AuthenticView[1634], AuthenticRange[1606] and AuthenticDataTransfer[1601]. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

Methods

getData[1602]

Properties

dropEffect[1602]
ownDrag[1602]
type[1602]

Description

The events OnDragOver and OnBeforeDrop provide information about the object being dragged with an instance of type AuthenticDataTransfer. It contains a description of the dragged object and its content. The latter is available either as string or a pointer to a COM object supporting the IUnkown interface.

29.3.2.3.1 dropEffect

Property: dropEffect as long

Description
The property stores the drop effect from the default event handler. You can set the drop effect if you change this value and return TRUE for the event handler (or set AuthenticEvent.cancelBubble 1810 to TRUE if you are still using the now obsolete AuthenticEvent interface).

Errors
2101 Invalid address for the return parameter was specified.

29.3.2.3.2 getData

Method: getData() as Variant

Description
Retrieve the data associated with the dragged object. Depending on AuthenticDataTransfer.type 1602, that data is either a string or a COM interface pointer of type IUnknown.

Errors
2101 Invalid address for the return parameter was specified.

29.3.2.3.3 ownDrag

Property: ownDrag as Boolean (read-only)

Description
The property is TRUE if the current dragging source comes from inside Authentic View.

Errors
2101 Invalid address for the return parameter was specified.

29.3.2.3.4 type

Property: type as String (read-only)

Description
Holds the type of data you get with the DocEditDataTransfer.getData 1602 method.

Currently supported data types are:

OWN	data from Authentic View itself
TEXT	plain text
UNICODETEXT	plain text as UNICODE

(C) 2015-2021 Altova GmbH

Errors
2101 Invalid address for the return parameter was specified.

29.3.2.4 AuthenticEventContext

The `EventContext` interface gives access to many properties of the context in which a macro is executed.

29.3.2.4.1 EvaluateXPath

Method: EvaluateXPath (strExpression as string) as strValue as string

Return Value
The method evaluates the XPath expression in the context of the node within which the event was triggered and returns a string.

Description
`EvaluateXPath()` executes an XPath expression with the given event context. The result is returned as a string, in the case of a sequence it is a space-separated string.

Errors
2201 Invalid object.
2202 No context.
2209 Invalid parameter.
2210 Internal error.
2211 XPath error.

29.3.2.4.2 GetEventContextType

Method: GetEventContextType `()` Type as `AuthenticEventContextType` enumeration

Return Value
Returns the context node type.

Description
`GetEventContextType` allows the user to determine whether the macro is in an XML node or in an XPath atomic item context. The enumeration `AuthenticEventContextType` is defined as follows:

authenticEventContextXML,
authenticEventContextAtomicItem,
authenticEventContextOther

If the context is a normal XML node, the `GetXMLNode()` function gives access to it (returns `NULL` if not).

Errors
2201 Invalid object.
2202 No context.

2209 Invalid parameter.

29.3.2.4.3 GetNormalizedTextValue

Method: GetNormalizedTextValue() strValue as string

Return Value
Returns the value of the current node as string

Errors
2201 Invalid object.
2202 No context.
2203 Invalid context
2209 Invalid parameter.

29.3.2.4.4 GetVariableValue

Method: GetVariableValue(strName as string) strValue as string

Return Value
Gets the value of the variable submitted as the parameter.

Description
`GetVariableValue` gets the variable's value in the scope of the context.

```
nZoom = parseInt( AuthenticView.EventContext.GetVariableValue( 'Zoom' ) );
if ( nZoom > 1 )
{
    AuthenticView.EventContext.SetVariableValue( 'Zoom', nZoom - 1 );
}
```

Errors
2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2209 Invalid parameter

29.3.2.4.5 GetXMLNode

Method: GetXMLNode() Node as XMLData object

Return Value
Returns the context XML node or `NULL`

(C) 2015-2021 Altova GmbH

Errors
2201 Invalid object.
2202 No context.
2203 Invalid context
2209 Invalid parameter.

29.3.2.4.6 IsAvailable

Method: IsAvailable() as Boolean

Return Value
Returns true if `EventContext` is set, false otherwise.

Errors
2201 Invalid object.

29.3.2.4.7 SetVariableValue

Method: SetVariableValue(strName as string, strValue as string)

Return Value
Sets the value (second parameter) of the variable submitted in the first parameter.

Description
`SetVariableValue` sets the variable's value in the scope of the context.

```
nZoom = parseInt( AuthenticView.EventContext.GetVariableValue( 'Zoom' ) );
if ( nZoom > 1 )
{
    AuthenticView.EventContext.SetVariableValue( 'Zoom', nZoom - 1 );
}
```

Errors
2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2207 Variable read-only
2208 No modification allowed

29.3.2.5 AuthenticRange

The first table lists the properties and methods of AuthenticRange that can be used to navigate through the document and select specific portions.

Properties	Methods	
Application 1608	Clone 1609	MoveBegin 1627
FirstTextPosition 1613	CollapsToBegin 1609	MoveEnd 1627
FirstXMLData 1613	CollapsToEnd 1610	NextCursorPosition 1619
FirstXMLDataOffset 1614	ExpandTo 1612	PreviousCursorPosition 1620
LastTextPosition 1624	Goto 1618	Select 1630
LastXMLData 1625	GotoNext 1618	SelectNext 1630
LastXMLDataOffset 1626	GotoPrevious 1619	SelectPrevious 1631
Parent 1628	IsEmpty 1622	SetFromRange 1633
	IsEqual 1623	

The following table lists the content modification methods, most of which can be found on the right/button mouse menu.

Properties	Edit operations	Dynamic table operations
Text 1633	Copy 1610	AppendRow 1607
	Cut 1610	DeleteRow 1611
	Delete 1610	DuplicateRow 1611
	IsCopyEnabled 1622	InsertRow 1621
	IsCutEnabled 1622	IsFirstRow 1623
	IsDeleteEnabled 1622	IsInDynamicTable 1623
	IsPasteEnabled 1624	IsLastRow 1623
	Paste 1628	MoveRowDown 1628
		MoveRowUp 1628

The following methods provide the functionality of the Authentic entry helper windows for range objects.

Operations of the entry helper windows		
Elements	Attributes	Entities
CanPerformActionWith 1609	GetElementAttributeValue 1616	GetEntityNames 1617

(C) 2015-2021 Altova GmbH

CanPerformAction 1608	GetElementAttributeNames 1615	InsertEntity 1620
PerformAction 1629	GetElementHierarchy 1616	
	HasElementAttribute 1620	
	IsTextStateApplied 1624	
	SetElementAttributeValue 1632	

Description
AuthenticRange objects are the 'cursor' selections of the automation interface. You can use them to point to any cursor position in the Authentic view, or select a portion of the document. The operations available for AuthenticRange objects then work on this selection in the same way, as the corresponding operations of the user interface do with the current user interface selection. The main difference is that you can use an arbitrary number of AuthenticRange objects at the same time, whereas there is exactly one cursor selection in the user interface.

To get to an initial range object use AuthenticView.Selection 1650, to obtain a range corresponding with the current cursor selection in the user interface. Alternatively, some trivial ranges are accessible via the read/only properties AuthenticView.DocumentBegin 1645, AuthenticView.DocumentEnd 1645, and AuthenticView.WholeDocument 1652. The most flexible method is AuthenticView.Goto 1647, which allows navigation to a specific portion of the document within one call. For more complex selections, combine the above with the various navigation methods on range objects listed in the first table on this page.

Another method to select a portion of the document is to use the position properties of the range object. Two positioning systems are available and can be combined arbitrarily:

- **Absolute** text cursor positions, starting with position 0 at the document beginning, can be set and retrieved for the beginning and end of a range. For more information see FirstTextPosition 1613 and LastTextPosition 1624. This method requires complex internal calculations and should be used with care.

- The **XMLData** element and a text position inside this element, can be set and retrieved for the beginning and end of a range. For more information see FirstXMLData 1613, FirstXMLDataOffset 1614, LastXMLData 1625, and LastXMLDataOffset 1626. This method is very efficient but requires knowledge of the underlying document structure. It can be used to locate XMLData objects and perform operations on them otherwise not accessible through the user interface.

Modifications to the document content can be achieved by various methods:

- The Text 1633 property allows you to retrieve the document text selected by the range object. If set, the selected document text gets replaced with the new text.
- The standard document edit functions Cut 1610, Copy 1610, Paste 1628 and Delete 1610.
- Table operations for tables that can grow dynamically.
- Methods that map the functionality of the Authentic entry helper windows.
- Access to the XMLData 1795 objects of the underlying document to modify them directly.

29.3.2.5.1 AppendRow

Method: AppendRow()as Boolean

Description

If the beginning of the range is inside a dynamic table, this method inserts a new row at the end of the selected table. The selection of the range is modified to point to the beginning of the new row. The function returns *true* if the append operation was successful, otherwise *false*.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------------------
' Scripting environment - VBScript
' Append row at end of current dynamically growable table
' ---------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
        objRange.AppendRow
        ' objRange points to beginning of new row
        objRange.Select
End If
```

29.3.2.5.2 Application

Property: Application as Application[1580] (read-only)

Description
Accesses the XMLSpy application object.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.3 CanPerformAction

Method: CanPerformAction (*eAction* as SPYAuthenticActions[1844], *strElementName* as String) as Boolean

Description
CanPerformAction and its related methods enable access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content, without having to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If the location can be found, the method returns *True*, otherwise it returns *False*.

HINT: To find out all valid element names for a given action, use CanPerformActionWith[1609].

Errors
2001 The authentic range object or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples
See PerformAction[1629].

29.3.2.5.4 CanPerformActionWith

Method: CanPerformActionWith (*eAction* as SPYAuthenticActions[1844], *out_arrElementNames* as Variant)

Description
PerformActionWith and its related methods, enable access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content without having to know exactly where the modification will take place.

This method returns an array of those element names that the specified action can be performed with.

HINT: To apply the action use CanPerformActionWith[1609].

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples
See PerformAction[1629].

29.3.2.5.5 Clone

Method: Clone() as AuthenticRange[1606]

Description
Returns a copy of the range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.6 CollapsToBegin

Method: CollapsToBegin() as AuthenticRange[1606]

Description
Sets the end of the range object to its begin. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

29.3.2.5.7 CollapsToEnd

Method: CollapsToEnd() as AuthenticRange[1606]

Description
Sets the beginning of the range object to its end. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.8 Copy

Method: Copy() as Boolean

Description
Returns *False* if the range contains no portions of the document that may be copied.
Returns *True* if text, and in case of fully selected XML elements the elements as well, has been copied to the copy/paste buffer.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.9 Cut

Method: Cut() as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted.
Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document and saved in the copy/paste buffer.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.10 Delete

Method: Delete() as Boolean

(C) 2015-2021 Altova GmbH

Description
Returns *False* if the range contains portions of the document that may not be deleted.
Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.11 DeleteRow

Method: DeleteRow() as Boolean

Description
If the beginning of the range is inside a dynamic table, this method deletes the selected row. The selection of the range gets modified to point to the next element after the deleted row. The function returns *true,* if the delete operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' ----------------------------------------------------------
' Scripting environment - VBScript
' Delete selected row from dynamically growing table
' ----------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we are in a table
If objRange.IsInDynamicTable Then
        objRange.DeleteRow
End If
```

29.3.2.5.12 DuplicateRow

Method: DuplicateRow() as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a duplicate of the current row after the selected one. The selection of the range gets modified to point to the beginning of the new row. The function returns *true* if the duplicate operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------------------
' Scripting environment - VBScript
' duplicate row in current dynamically growable table
' ---------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
        objRange.DuplicateRow
        ' objRange points to beginning of new row
        objRange.Select
End If
```

29.3.2.5.13 EvaluateXPath

Method: EvaluateXPath (strExpression as string) strValue as string

Return Value
The method returns a string

Description
`EvaluateXPath()` executes an XPath expression with the context node being the beginning of the range selection. The result is returned as a string, in the case of a sequence it is a space-separated string. If XML context node is irrelevant, the user may provide any node, like `AuthenticView.XMLDataRoot`.

Errors

2001	Invalid object
2005	Invalid parameter
2008	Internal error
2202	Missing context node
2211	XPath error

29.3.2.5.14 ExpandTo

Method: ExpandTo (*eKind* as SPYAuthenticElementKind[1845]), as AuthenticRange[1606]

Description
Selects the whole element of type eKind, that starts at, or contains, the first cursor position of the range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Range expansion would be beyond end of document.
2005	Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.5.15 FirstTextPosition

Property: FirstTextPosition as Long

Description
Set or get the left-most text position index of the range object. This index is always less or equal to LastTextPosition[1624]. Indexing starts with 0 at document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decrementing the test position by 1 has the same effect as the cursor-left key.

If you set FirstTextPosition to a value greater than the current LastTextPosition[1624], LastTextPosition[1624] gets set to the new FirstTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors

2001	The authentic range object, or its related view object is not valid.
2005	Invalid address for the return parameter was specified.
2006	A text position outside the document was specified.

Examples

```
' ---------------------------------------
' Scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

' let's create a range that selects the whole document
' in an inefficient way
Dim objRange
' we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

' let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
        MsgBox "Test using direct text cursor positioning was ok"
Else
        MsgBox "Ooops!"
End If
```

29.3.2.5.16 FirstXMLData

Property: FirstXMLData as XMLData[1795]

Description
Set or get the first XMLData element in the underlying document that is partially, or completely selected by the range. The exact beginning of the selection is defined by the FirstXMLDataOffset[1614] attribute.

Whenever you set FirstXMLData to a new data object, FirstXMLDataOffset[1614] gets set to the first cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set FirstXMLData / FirstXMLDataOffset[1614] selects a position greater then the current LastXMLData[1625] / LastXMLDataOffset[1626], the latter gets moved to the new start position.

HINT: You can use the FirstXMLData[1613] and LastXMLData[1625] properties to directly access and manipulate the underlying XML document in those cases where the methods available with the AuthenticRange[1606] object are not sufficient.

Errors

2001	The authentic range object, or its related view object is not valid.
2005	Invalid address for the return parameter was specified.
2008	Internal error
2009	The XMLData object cannot be accessed.

Examples

```
' ------------------------------------------------
' Scripting environment - VBScript
' show name of currently selected XMLData element
' ------------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objXmlData
Set objXMLData = objAuthenticView.Selection.FirstXMLData
' authentic view adds a 'text' child element to elements
' of the document which have content. So we have to go one
' element up.
Set objXMLData = objXMLData.Parent
MsgBox "Current selection selects element " & objXMLData.Name
```

29.3.2.5.17 FirstXMLDataOffset

Property: FirstXMLDataOffset as Long

Description
Set or get the cursor position offset inside FirstXMLData[1613] element for the beginning of the range. Offset positions are based on the characters returned by the Text[1633] property, and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on screen. Although the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.

- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

(C) 2015-2021 Altova GmbH

If FirstXMLData / FirstXMLDataOffset[1614] selects a position after the current LastXMLData[1625] / LastXMLDataOffset[1626], the latter gets moved to the new start position.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  ' start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1  ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

' were we successful?
If objRange.IsEqual(objRange2) Then
        objRange.Select()
Else
        MsgBox "Oops"
End If
```

29.3.2.5.18 GetElementAttributeNames

Method: GetElementAttributeNames (*strElementName* as String, *out_arrAttributeNames* as Variant)

Description

Retrieve the names of all attributes for the enclosing element with the specified name. Use the element/attribute pairs, to set or get the attribute value with the methods GetElementAttributeValue[1616] and SetElementAttributeValue[1632].

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue[1632].

29.3.2.5.19 GetElementAttributeValue

Method: GetElementAttributeValue (*strElementName* as String, *strAttributeName* as String) as String

Description
Retrieve the value of the attribute specified in strAttributeName, for the element identified with strElementName. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use GetElementAttributeNames[1615], or HasElementAttribute[1620].

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid attribute name was specified.
Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue[1632].

29.3.2.5.20 GetElementHierarchy

Method: GetElementHierarchy (*out_arrElementNames* as Variant)

Description
Retrieve the names of all XML elements that are parents of the current selection. Inner elements get listed before enclosing elements. An empty list is returned whenever the current selection is not inside a single XMLData element.

The names of the element hierarchy, together with the range object uniquely identify XMLData elements in the document. The attributes of these elements can be directly accessed by GetElementAttributeNames[1615], and related methods.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

C# Examples

```
' ----------------------------------------
' C#
' ----------------------------------------

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
```

```
        {
            XMLSpyLib.Application app = new XMLSpyLib.Application();

            app.ShowApplication(true);

            XMLSpyLib.AuthenticView view = app.ActiveDocument.AuthenticView;
            XMLSpyLib.AuthenticRange range = view.DocumentBegin;

            object o = null;
            range.GetElementHierarchy(ref o);

            object[] elements = (object[])o;

            foreach (string e in elements)
            {
                Console.WriteLine(e);
            }
        }
    }
}
```

Also see: SetElementAttributeValue[1632].

29.3.2.5.21 GetEntityNames

Method: GetEntityNames (*out_arrEntityNames* as Variant)

Description
Retrieve the names of all defined entities. The list of retrieved entities is independent of the current selection, or location. Use one of these names with the InsertEntity[1620] function.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples
See: GetElementHierarchy[1616] and InsertEntity[1620].

29.3.2.5.22 GetVariableValue

Method: GetVariableValue`(strName as string) strVal as string`

Return Value
Gets the value of the variable named as the method's parameter.

Errors
2001 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence

2209 Invalid parameter

29.3.2.5.23 Goto

Method: Goto (*eKind* as SPYAuthenticElementKind[1845], *nCount* as Long, *eFrom* as SPYAuthenticDocumentPosition[1844]) as AuthenticRange[1606]

Description
Sets the range to point to the beginning of the nCount element of type eKind. The start position is defined by the parameter eFrom.

Use positive values for nCount to navigate to the document end. Use negative values to navigate to the beginning of the document. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before begin of document.
2005	Invalid element kind specified.
	Invalid start position specified.
	Invalid address for the return parameter was specified.

29.3.2.5.24 GotoNext

Method: GotoNext (*eKind* as SPYAuthenticElementKind[1845]) as AuthenticRange[1606]

Description
Sets the range to the beginning of the next element of type eKind. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2005	Invalid element kind specified.
	Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document word-by-word
' ------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
```

```
bEndOfDocument = False

On Error Resume Next
While Not bEndOfDocument
        objRange.GotoNext(spyAuthenticWord).Select
        If ((Err.number - vbObjecterror) = 2003) Then
                bEndOfDocument = True
                Err.Clear
        ElseIf (Err.number <> 0) Then
                Err.Raise  ' forward error
        End If
Wend
```

29.3.2.5.25 GotoNextCursorPosition

Method: GotoNextCursorPosition() as AuthenticRange[1606]

Description
Sets the range to the next cursor position after its current end position. Returns the modified object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2005	Invalid address for the return parameter was specified.

29.3.2.5.26 GotoPrevious

Method: GotoPrevious (*eKind* as SPYAuthenticElementKind[1845]) as AuthenticRange[1606]

Description
Sets the range to the beginning of the element of type eKind which is before the beginning of the current range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2004	Target lies before beginning of document.
2005	Invalid element kind specified. Invalid address for the return parameter was specified.

Examples

```
' ----------------------------------------
' Scripting environment - VBScript
' Scan through the whole document tag-by-tag
' ----------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
```

```
Dim bBeginOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
        objRange.GotoPrevious(spyAuthenticTag).Select
        If ((Err.number - vbObjecterror) = 2004) Then
                bBeginOfDocument = True
                Err.Clear
        ElseIf (Err.number <> 0) Then
                Err.Raise  ' forward error
        End If
Wend
```

29.3.2.5.27 GotoPreviousCursorPosition

Method: GotoPreviousCursorPosition() as AuthenticRange[1606]

Description
Set the range to the cursor position immediately before the current position. Returns the modified object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2004 Target lies before begin of document.
2005 Invalid address for the return parameter was specified.

29.3.2.5.28 HasElementAttribute

Method: HasElementAttribute (*strElementName* as String, *strAttributeName* as String) as Boolean

Description
Tests if the enclosing element with name strElementName, supports the attribute specified in strAttributeName.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid address for the return parameter was specified.

29.3.2.5.29 InsertEntity

Method: InsertEntity (*strEntityName* as String)

Description
Replace the ranges selection with the specified entity. The specified entity must be one of the entity names returned by GetEntityNames[1617].

Errors
2001 The authentic range object, or its related view object is no longer valid.

(C) 2015-2021 Altova GmbH

2005 Unknown entry name was specified.

Examples

```
' ----------------------------------------------------------
' Scripting environment - VBScript
' Insert the first entity in the list of available entities
' ----------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we get the names of all available entities as they
' are shown in the entry helper of XMLSpy
Dim arrEntities
objRange.GetEntityNames arrEntities

' we insert the first one of the list
If UBound(arrEntities) >= 0 Then
        objRange.InsertEntity arrEntities(0)
Else
        MsgBox "Sorry, no entities are available for this document"
End If
```

29.3.2.5.30 InsertRow

Method: InsertRow() as Boolean

Description

If the beginning of the range is inside a dynamic table, this method inserts a new row before the current one. The selection of the range gets modified to point to the beginning of the newly inserted row. The function returns *true* if the insert operation was successful, otherwise *false*.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' ----------------------------------------------------------------
' Scripting environment - VBScript
' Insert row at beginning of current dynamically growing table
' ----------------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
        objRange.InsertRow
        ' objRange points to beginning of new row
        objRange.Select
  End If
```

29.3.2.5.31 IsCopyEnabled

Property: IsCopyEnabled as Boolean (read-only)

Description
Checks if the copy operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.32 IsCutEnabled

Property: IsCutEnabled as Boolean (read-only)

Description
Checks if the cut operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.33 IsDeleteEnabled

Property: IsDeleteEnabled as Boolean (read-only)

Description
Checks if the delete operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.34 IsEmpty

Method: IsEmpty() as Boolean

Description
Tests if the first and last position of the range are equal.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.5.35 IsEqual

Method: IsEqual (*objCmpRange* as AuthenticRange[1606]) as Boolean

Description
Tests if the start and end of both ranges are the same.

Errors
2001 One of the two range objects being compared, is invalid.
2005 Invalid address for a return parameter was specified.

29.3.2.5.36 IsFirstRow

Property: IsFirstRow as Boolean (read-only)

Description
Test if the range is in the first row of a table. Which table is taken into consideration depends on the extent of the range. If the selection exceeds a single row of a table, the check is if this table is the first element in an embedding table. See the entry helpers of the user manual for more information.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.37 IsInDynamicTable

Method: IsInDynamicTable() as Boolean

Description
Test if the whole range is inside a table that supports the different row operations like 'insert', 'append', duplicate, etc.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.38 IsLastRow

Property: IsLastRow as Boolean (read-only)

Description
Test if the range is in the last row of a table. Which table is taken into consideration depends on the extent of the range. If the selection exceeds a single row of a table, the check is if this table is the last element in an embedding table. See the entry helpers of the user manual for more information.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.39 IsPasteEnabled

Property: IsPasteEnabled as Boolean (read-only)

Description
Checks if the paste operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.40 IsSelected

Property: IsSelected as Boolean

Description
Returns true() if selection is present. The selection range still can be empty: that happens when e.g. only the cursor is set.

29.3.2.5.41 IsTextStateApplied

Method: IsTextStateApplied (*i_strElementName* as String) as Boolean

Description
Checks if all the selected text is embedded into an XML Element with name i_strElementName. Common examples for the parameter i_strElementName are "strong", "bold" or "italic".

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.42 LastTextPosition

Property: LastTextPosition as Long

Description
Set or get the rightmost text position index of the range object. This index is always greater or equal to FirstTextPosition[1613]. Indexing starts with 0 at the document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decreasing the test position by 1 has the same effect as the cursor-left key.

(C) 2015-2021 Altova GmbH

If you set LastTextPosition to a value less then the current FirstTextPosition[1613], FirstTextPosition[1613] gets set to the new LastTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2006 A text position outside the document was specified.

Examples

```
' ---------------------------------------
' Scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

' let's create a range that selects the whole document
' in an inefficient way
Dim objRange
' we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

' let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
        MsgBox "Test using direct text cursor positioning was ok"
Else
        MsgBox "Oops!"
End If
```

29.3.2.5.43 LastXMLData

Property: LastXMLData as XMLData[1795]

Description

Set or get the last XMLData element in the underlying document that is partially or completely selected by the range. The exact end of the selection is defined by the LastXMLDataOffset[1626] attribute.

Whenever you set LastXMLData to a new data object, LastXMLDataOffset[1626] gets set to the last cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set LastXMLData / LastXMLDataOffset[1626], select a position less then the current FirstXMLData[1613] / FirstXMLDataOffset[1614], the latter gets moved to the new end position.

HINT: You can use the FirstXMLData[1613] and LastXMLData[1625] properties to directly access and manipulate the underlying XML document in those cases, where the methods available with the AuthenticRange[1606] object are not sufficient.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2008 Internal error
2009 The XMLData object cannot be accessed.

29.3.2.5.44 LastXMLDataOffset

Property: LastXMLDataOffset as Long

Description

Set or get the cursor position inside LastXMLData[1625] element for the end of the range.

Offset positions are based on the characters returned by the Text[1633] property and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on the screen. Although, the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If LastXMLData[1625] / LastXMLDataOffset[1626] selects a position before FirstXMLData[1613] / FirstXMLDataOffset[1614], the latter gets moved to the new end position.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  ' start at beginning of element text
```


(C) 2015-2021 Altova GmbH

```
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1  ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

' were we successful?
If objRange.IsEqual(objRange2) Then
        objRange.Select()
Else
        MsgBox "Ooops"
End If
```

29.3.2.5.45 MoveBegin

Method: MoveBegin (*eKind* as SPYAuthenticElementKind[1845], *nCount* as Long) as AuthenticRange[1606]

Description
Move the beginning of the range to the beginning of the nCount element of type eKind. Counting starts at the current beginning of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The end of the range stays unmoved, unless the new beginning would be larger than it. In this case, the end is moved to the new beginning. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before beginning of document.
2005	Invalid element kind specified.
	Invalid address for the return parameter was specified.

29.3.2.5.46 MoveEnd

Method: MoveEnd (*eKind* as SPYAuthenticElementKind[1845], *nCount* as Long) as AuthenticRange[1606]

Description
Move the end of the range to the begin of the nCount element of type eKind. Counting starts at the current end of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The beginning of the range stays unmoved, unless the new end would be less than it. In this case, the beginning gets moved to the new end. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before begin of document.
2005	Invalid element kind specified.
	Invalid address for the return parameter was specified.

29.3.2.5.47 MoveRowDown

Method: MoveRowDown() as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the last row in this table, this method swaps this row with the row immediately below. The selection of the range moves with the row, but does not otherwise change. The function returns *true* if the move operation was successful, otherwise *false*.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.48 MoveRowUp

Method: MoveRowUp() as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the first row in this table, this method swaps this row with the row above. The selection of the range moves with the row, but does not change otherwise. The function returns *true* if the move operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.49 Parent

Property: Parent as AuthenticView[1634] (read-only)

Description
Access the view that owns this range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.50 Paste

Method: Paste() as Boolean

Description
Returns *False* if the copy/paste buffer is empty, or its content cannot replace the current selection.

(C) 2015-2021 Altova GmbH

Otherwise, deletes the current selection, inserts the content of the copy/paste buffer, and returns *True*.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.5.51 PerformAction

Method: PerformAction (*eAction* as SPYAuthenticActions[1844], *strElementName* as String) as Boolean

Description

PerformAction and its related methods, give access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content without a need to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If no such location can be found, the method returns *False*. Otherwise, the document gets modified and the range points to the beginning of the modification.

HINT: To find out element names that can be passed as the second parameter use CanPerformActionWith[1609].

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples

```
' ------------------------------------------
' Scripting environment - VBScript
' Insert the innermost element
' ------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' we determine the elements that can be inserted at the current position
Dim arrElements()
objRange.CanPerformActionWith spyAuthenticInsertBefore, arrElements

' we insert the first (innermost) element
If UBound(arrElements) >= 0 Then
        objRange.PerformAction spyAuthenticInsertBefore, arrElements(0)
        ' objRange now points to the beginning of the inserted element
        ' we set a default value and position at its end
        objRange.Text = "Hello"
        objRange.ExpandTo(spyAuthenticTag).CollapsToEnd().Select
Else
        MsgBox "Can't insert any elements at current position"
End If
```

29.3.2.5.52 Select

Method: Select()

Description
Makes this range the current user interface selection. You can achieve the same result using: '*objRange.Parent.Selection = objRange*'

Errors
2001 The authentic range object or its related view object is no longer valid.

Examples
```
' ---------------------------------------
' Scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' set current selection to end of document
objAuthenticView.DocumentEnd.Select()
```

29.3.2.5.53 SelectNext

Method: SelectNext (*eKind* as SPYAuthenticElementKind[1845]) as AuthenticRange[1606]

Description
Selects the element of type eKind after the current end of the range. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples
```
' -------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document word-by-word
' -------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False
```

```
On Error Resume Next
While Not bEndOfDocument
	objRange.SelectNext(spyAuthenticWord).Select
	If ((Err.number - vbObjecterror) = 2003) Then
		bEndOfDocument = True
		Err.Clear
	ElseIf (Err.number <> 0) Then
		Err.Raise  ' forward error
	End If
Wend
```

29.3.2.5.54 SelectPrevious

Method: GotoPrevious (*eKind* as SPYAuthenticElementKind[1845]) as AuthenticRange[1606]

Description
Selects the element of type eKind before the current beginning of the range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2004	Target lies before begin of document.
2005	Invalid element kind specified.
	Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document tag-by-tag
' ------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
Dim bBeginOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
	objRange.SelectPrevious(spyAuthenticTag).Select
	If ((Err.number - vbObjecterror) = 2004) Then
		bBeginOfDocument = True
		Err.Clear
	ElseIf (Err.number <> 0) Then
		Err.Raise  ' forward error
	End If
Wend
```

29.3.2.5.55 SetElementAttributeValue

Method: SetElementAttributeValue (*strElementName* as String, *strAttributeName* as String, *strAttributeValue* as String)

Description
Set the value of the attribute specified in strAttributeName for the element identified with strElementName. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use GetElementAttributeNames[1615], or HasElementAttribute[1620].

Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid element name was specified.
	Invalid attribute name was specified.
	Invalid attribute value was specified.

Examples

```
' ------------------------------------------
' Scripting environment - VBScript
' Get and set element attributes
' ------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we find out all the elements below the beginning of the range
Dim arrElements
objRange.GetElementHierarchy arrElements

If IsArray(arrElements) Then
	If UBound(arrElements) >= 0 Then
		' we use the top level element and find out its valid attributes
		Dim arrAttrs()
		objRange.GetElementAttributeNames arrElements(0), arrAttrs

		If UBound(arrAttrs) >= 0 Then
			' we retrieve the current value of the first valid attribute
			Dim strAttrVal
			strAttrVal = objRange.GetElementAttributeValue (arrElements(0), arrAttrs(0))
			msgbox "current value of " & arrElements(0) & "//" & arrAttrs(0) & " is: " & strAttrVal

			' we change this value and read it again
			strAttrVal = "Hello"
			objRange.SetElementAttributeValue arrElements(0), arrAttrs(0), strAttrVal
			strAttrVal = objRange.GetElementAttributeValue (arrElements(0), arrAttrs(0))
			msgbox "new value of " & arrElements(0) & "//" & arrAttrs(0) & " is: " & strAttrVal
		End If
	End If
End If
```


(C) 2015-2021 Altova GmbH

29.3.2.5.56 SetFromRange

Method: SetFromRange (*objSrcRange* as AuthenticRange[1606])

Description
Sets the range object to the same beginning and end positions as objSrcRange.

Errors
2001 One of the two range objects, is invalid.
2005 Null object was specified as source object.

29.3.2.5.57 SetVariableValue

Method: SetVariableValue`(strName as string, strValue as string)`

Return Value
Sets the value (second parameter) of the variable named in the first parameter.

Errors
2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2207 Variable read-only
2208 No modification allowed

29.3.2.5.58 Text

Property: Text as String

Description
Set or get the textual content selected by the range object.

The number of characters retrieved are not necessarily identical, as there are text cursor positions between the beginning and end of the selected range. Most document elements support an end cursor position different to the beginning cursor position of the following element. Drop-down lists maintain only one cursor position, but can select strings of any length. In the case of radio buttons and check boxes, the text property value holds the string of the corresponding XML element.

If the range selects more then one element, the text is the concatenation of the single texts. XML entities are expanded so that '&' is expected as '&'.

Setting the text to the empty string, does not delete any XML elements. Use Cut[1610], Delete[1610] or PerformAction[1629] instead.

Errors

2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for a return parameter was specified.

29.3.2.6 AuthenticView

Properties
Application 1643
AsXMLString 1644
DocumentBegin 1645
DocumentEnd 1645
Event 1646
MarkupVisibility 1649
Parent 1649
Selection 1650
XMLDataRoot 1652
WholeDocument 1652

Methods
Goto 1647
IsRedoEnabled 1648
IsUndoEnabled 1648
Print 1649
Redo 1650
Undo 1651
UpdateXMLInstanceEntities 1651

Events
OnBeforeCopy 1635
OnBeforeCut 1635
OnBeforeDelete 1636
OnBeforeDrop 1636
OnBeforePaste 1637
OnDragOver 1638
OnKeyBoardEvent 1639
OnMouseEvent 1640
OnSelectionChanged 1641

Description

AuthenticView and its child objects AuthenticRange 1606 and AuthenticDataTransfer provide you with an interface for Authentic View, which allow easy and consistent modification of document contents. These interfaces replace the following interfaces which are marked now as **obsolete:**

OldAuthenticView (old name was DocEditView)
AuthenticSelection (old name was DocEditSelection, superseded by AuthenticRange 1606)
AuthenticEvent (old name was DocEditEvent)

AuthenticView gives you easy access to specific features such as printing, the multi-level undo buffer, and the current cursor selection, or position.

AuthenticView uses objects of type AuthenticRange 1606 to make navigation inside the document straight-forward, and to allow for the flexible selection of logical text elements. Use the properties DocumentBegin 1645, DocumentEnd 1645, or WholeDocument 1652 for simple selections, while using the Goto 1647 method for more complex selections. To navigate relative to a given document range, see the methods and properties of the AuthenticRange 1606 object.

Examples

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' secure access to authentic view object
' ---------------------------------------
Dim objDocument
Set objDocument = Application.ActiveDocument
If (Not objDocument Is Nothing) Then
        ' we have an active document, now check for view mode
        If (objDocument.CurrentViewMode <> spyViewAuthentic) Then
                If (Not objDocument.SwitchViewMode (spyViewAuthentic)) Then
                        MsgBox "Active document does not support authentic view mode"
                Else
                        ' now it is safe to access the authentic view object
```


(C) 2015-2021 Altova GmbH

```
                    Dim objAuthenticView
                    Set objAuthenticView = objDocument.AuthenticView
                    ' now use the authentic view object

                End If
        End If
Else
        MsgBox "No document is open"
End If
```

29.3.2.6.1 Events

29.3.2.6.1.1 OnBeforeCopy

Event: OnBeforeCopy() as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticBeforeCopy()
        ' On_AuthenticBeforeCopy = False  ' to disable operation
End Function
```

Scripting environment - JScript:

```
function On_AuthenticBeforeCopy()
{
        // return false;  /* to disable operation */
}
```

IDE Plugin:

IXMLSpyPlugIn.OnEvent (21, ...)// nEventId = 21

Description

This event gets triggered before a copy operation gets performed on the document. Return *True* (or nothing) to allow copy operation. Return *False* to disable copying.

29.3.2.6.1.2 OnBeforeCut

Event: OnBeforeCut() as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticBeforeCut()
        ' On_AuthenticBeforeCut = False  ' to disable operation
End Function
```

Scripting environment - JScript:

```
function On_AuthenticBeforeCut()
{
        // return false;  /* to disable operation */
}
```

IDE Plugin:

IXMLSpyPlugIn.OnEvent (20, ...)// nEventId = 20

Description
This event gets triggered before a cut operation gets performed on the document. Return *True* (or nothing) to allow cut operation. Return *False* to disable operation.

29.3.2.6.1.3 OnBeforeDelete

Event: OnBeforeDelete() as Boolean

Scripting environment - VBScript:
```
Function On_AuthenticBeforeDelete()
	' On_AuthenticBeforeDelete = False  ' to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticBeforeDelete()
{
	// return false;  /* to disable operation */
}
```

IDE Plugin:
IXMLSpyPlugIn.OnEvent (22, ...)// nEventId = 22

Description
This event gets triggered before a delete operation gets performed on the document. Return *True* (or nothing) to allow delete operation. Return *False* to disable operation.

29.3.2.6.1.4 OnBeforeDrop

Event: OnBeforeDrop (*i_nXPos* as Long, *i_nYPos* as Long, *i_ipRange* as AuthenticRange 1606, *i_ipData* as cancelBoolean

Scripting environment - VBScript:
```
Function On_AuthenticBeforeDrop(nXPos, nYPos, objRange, objData)
	' On_AuthenticBeforeDrop = False  ' to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticBeforeDrop(nXPos, nYPos, objRange, objData)
{
	// return false;  /* to disable operation */
}
```

IDE Plugin:
IXMLSpyPlugIn.OnEvent (11, ...)// nEventId = 11

Description
This event gets triggered whenever a previously dragged object gets dropped inside the application window. All event related information gets passed as parameters.

The first two parameters specify the mouse position at the time when the event occurred. The parameter *objRange* passes a range object that selects the XML element below the mouse position. The value of this parameter might be *NULL*. Be sure to check before you access the range object. The parameter *objData* allows to access information about the object being dragged.

Return *False* to cancel the drop operation. Return *True* (or nothing) to continue normal operation.

Examples

```
' ----------------------------------------------------------------------------
' VB code snippet - connecting to object level events
' ----------------------------------------------------------------------------
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnBeforeDrop
' event of object objView
Private Function objView_OnBeforeDrop(ByVal i_nXPos As Long, ByVal i_nYPos As Long,
                                      ByVal i_ipRange As IAuthenticRange,
                                      ByVal i_ipData As IAuthenticDataTransfer) As Boolean

   If (Not i_ipRange Is Nothing) Then
      MsgBox ("Dropping on content is prohibited");
      Return False;
   Else
      Return True;
   End If
End Function

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

29.3.2.6.1.5 OnBeforePaste

Event: OnBeforePaste (*objData* as Variant, *strType* as String) as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticBeforePaste(objData, strType)
        ' On_AuthenticBeforePaste = False  ' to disable operation
End Function
```

Scripting environment - JScript:

```
function On_AuthenticBeforePaste(objData, strType)
{
        // return false;  /* to disable operation */
}
```

IDE Plugin:

IXMLSpyPlugIn.OnEvent (19, ...)// nEventId = 19

Description
This event gets triggered before a paste operation gets performed on the document. The parameter *strType* is one of "TEXT", "UNICODETEXT" or "IUNKNOWN". In the first two cases *objData* contains a string representation of the object that will be pasted. In the later case, *objData* contains a pointer to an IUnknown COM interface.

Return *True* (or nothing) to allow paste operation. Return *False* to disable operation.

29.3.2.6.1.6 OnBeforeSave

Event: OnBeforeSave (SaveAs flag) as Boolean

Description: `OnBeforeSave` gives the opportunity to e.g. warn the user about overwriting the existing XML document, or to make the document read-only when specific circumstances are not met. The event will be fired before the file dialog is shown.

29.3.2.6.1.7 OnDragOver

Event: OnDragOver (*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as SPYMouseEvent[1850], *objRange* as AuthenticRange[1606], *objData* as AuthenticDataTransfer) as Boolean

Scripting environment - VBScript:
```
Function On_AuthenticDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
        ' On_AuthenticDragOver = False  ' to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
{
        // return false;  /* to disable operation */
}
```

IDE Plugin:
IXMLSpyPlugIn.OnEvent (10, ...)// nEventId = 10

Description
This event gets triggered whenever an object from within or outside of Authentic View gets dragged with the mouse over the application window. All event related information gets passed as parameters.

The first three parameters specify the mouse position, the mouse button status and the status of the virtual keys at the time when the event occurred. The parameter *objRange* passes a range object that selects the XML element below the mouse position. The value of this parameter might be *NULL*. Be sure to check before you access the range object. The parameter *objData* allows to access information about the object being dragged.

Return *False* to cancel the drag operation. Return *True* (or nothing) to continue normal operation.

Examples
```
' ------------------------------------------------------------------------
' VB code snippet - connecting to object level events
' ------------------------------------------------------------------------
```

```
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnDragOver
' event of object objView
Private Function objView_OnDragOver(ByVal i_nXPos As Long, ByVal i_nYPos As Long,
                                    ByVal i_eMouseEvent As SPYMouseEvent,
                                    ByVal i_ipRange As IAuthenticRange,
                                    ByVal i_ipData As IAuthenticDataTransfer) As Boolean

    If (((i_eMouseEvent And spyShiftKeyDownMask) <> 0) And
            (Not i_ipRange Is Nothing)) Then
        MsgBox ("Floating over element " & i_ipRange.FirstXMLData.Parent.Name);
    End If

    Return True;
End Function

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

29.3.2.6.1.8 OnKeyboardEvent

Event: OnKeyboardEvent (*eKeyEvent* as SPYKeyEvent[1849], *nKeyCode* as Long, *nVirtualKeyStatus* as Long) as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticKeyboardEvent(eKeyEvent, nKeyCode, nVirtualKeyStatus)
        ' On_AuthenticKeyboardEvent = True  ' to cancel bubbling of event
End Function
```

Scripting environment - JScript:

```
function On_AuthenticKeyboardEvent(eKeyEvent, nKeyCode, nVirtualKeyStatus)
{
        // return true;  /* to cancel bubbling of event */
}
```

IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (30, ...)// nEventId = 30
```

Description

This event gets triggered for *WM_KEYDOWN*, *WM_KEYUP* and *WM_CHAR* Windows messages.

The actual message type is available in the *eKeyEvent* parameter. The status of virtual keys is combined in the parameter *nVirtualKeyStatus*. Use the bit-masks defined in the enumeration datatype SPYVirtualKeyMask[1856], to test for the different keys or their combinations.

29.3.2.6.1.9 OnLoad

Event: OnLoad ()

Description: `OnLoad` can be used e.g. to restrict some AuthenticView functionality, as shown in the example below:

```
function On_AuthenticLoad(  )
{
    // We are disabling all entry helpers in order to prevent user from manipulating XML tree
    AuthenticView.DisableElementEntryHelper();
    AuthenticView.DisableAttributeEntryHelper();

    // We are also disabling the markup buttons for the same purpose
    AuthenticView.SetToolbarButtonState( 'AuthenticMarkupSmall', authenticToolbarButtonDisabled );
    AuthenticView.SetToolbarButtonState( 'AuthenticMarkupLarge', authenticToolbarButtonDisabled );
    AuthenticView.SetToolbarButtonState( 'AuthenticMarkupMixed', authenticToolbarButtonDisabled );
}
```

In the example the status of the Markup Small, Markup Large, Markup Mixed toolbar buttons are manipulated with the help of button identifiers. See complete list 1641.

29.3.2.6.1.10 OnMouseEvent

Event: OnMouseEvent (*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as SPYMouseEvent 1850, *objRange* as AuthenticRange 1606) as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
        ' On_AuthenticMouseEvent = True  ' to cancel bubbling of event
End Function
```

Scripting environment - JScript:

```
function On_AuthenticMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
{
        // return true;  /* to cancel bubbling of event */
}
```

IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (31, ...)// nEventId = 31
```

Description

This event gets triggered for every mouse movement and mouse button Windows message.

The actual message type and the mouse buttons status, is available in the *eMouseEvent* parameter. Use the bit-masks defined in the enumeration datatype SPYMouseEvent 1850 to test for the different messages, button status, and their combinations.

The parameter *objRange* identifies the part of the document found at the current mouse cursor position. The range object always selects a complete tag of the document. (This might change in future versions, when a more precise positioning mechanism becomes available). If no selectable part of the document is found at the current position, the range object is *null*.

29.3.2.6.1.11 OnSelectionChanged

Event: OnSelectionChanged (*objNewSelection* as AuthenticRange[1606])

Scripting environment - VBScript:
Function On_AuthenticSelectionChanged (*objNewSelection*)
End Function

Scripting environment - JScript:
function On_AuthenticSelectionChanged (*objNewSelection*)
{
}

IDE Plugin:
IXMLSpyPlugIn.OnEvent (23, ...)// nEventId = 23

Description
This event gets triggered whenever the selection in the user interface changes.

Examples
```
' ---------------------------------------------------------------------------
' VB code snippet - connecting to object level events
' ---------------------------------------------------------------------------
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnSelectionChanged
' event of object objView
Private Sub objView_OnSelectionChanged (ByVal i_ipNewRange As XMLSpyLib.IAuthenticRange)
    MsgBox ("new selection: " & i_ipNewRange.Text)
End Sub

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

29.3.2.6.1.12 OnToolbarButtonClicked

Event: OnToolbarButtonClicked (Button identifier)

Description: `OnToolbarButtonClicked` is fired when a toolbar button was clicked by user. The parameter button identifier helps to determine which button was clicked. The list of predefined button identifiers is below:

- AuthenticPrint
- AuthenticPrintPreview
- AuthenticUndo
- AuthenticRedo
- AuthenticCut

- AuthenticCopy
- AuthenticPaste
- AuthenticClear
- AuthenticMarkupHide
- AuthenticMarkupLarge
- AuthenticMarkupMixed
- AuthenticMarkupSmall
- AuthenticValidate
- AuthenticChangeWorkingDBXMLCell
- AuthenticSave
- AuthenticSaveAs
- AuthenticReload
- AuthenticTableInsertRow
- AuthenticTableAppendRow
- AuthenticTableDeleteRow
- AuthenticTableInsertCol
- AuthenticTableAppendCol
- AuthenticTableDeleteCol
- AuthenticTableJoinCellRight
- AuthenticTableJoinCellLeft
- AuthenticTableJoinCellAbove
- AuthenticTableJoinCellBelow
- AuthenticTableSplitCellHorizontally
- AuthenticTableSplitCellVertically
- AuthenticTableAlignCellContentTop
- AuthenticTableCenterCellVertically
- AuthenticTableAlignCellContentBottom
- AuthenticTableAlignCellContentLeft
- AuthenticTableCenterCellContent
- AuthenticTableAlignCellContentRight
- AuthenticTableJustifyCellContent
- AuthenticTableInsertTable
- AuthenticTableDeleteTable
- AuthenticTableProperties
- AuthenticAppendRow
- AuthenticInsertRow
- AuthenticDuplicateRow
- AuthenticMoveRowUp
- AuthenticMoveRowDown
- AuthenticDeleteRow
- AuthenticDefineEntities
- AuthenticXMLSignature

For custom buttons the user might add his own identifiers. Please, note that the user must take care, as the identifiers are not checked for uniqueness. The same identifiers can be used to identify buttons in the `Set/GetToolbarState()` COM API calls. By adding code for different buttons, the user is in the position to completely redefine the AuthenticView toolbar behavior, adding own methods for table manipulation, etc.

(C) 2015-2021 Altova GmbH

29.3.2.6.1.13 OnToolbarButtonExecuted

Event: OnToolbarButtonExecuted (Button identifier)

Description: `OnToolbarButtonClicked` is fired when a toolbar button was clicked by user. The parameter button identifier helps to determine which button was clicked. See the list of predefined button identifiers[1641].

`OnToolbarButtonExecuted` is fired after the toolbar action was executed. It is useful e.g. to add update code, as shown in the example below:

```
//event fired when a toolbar button action was executed
function On_AuthenticToolbarButtonExecuted( varBtnIdentifier )
{
    // After whatever command user has executed - make sure to update toolbar button states
    UpdateOwnToolbarButtonStates();
}
```

In this case `UpdateOwnToolbarButtonStates` is a user function defined in the Global Declarations.

29.3.2.6.1.14 OnUserAddedXMLNode

Event: OnUserAddedXMLNode (XML node)

Description: `OnUserAddedXMLNode` will be fired when the user adds an XML node as a primary action. This happens in the situations, where the user clicks on

- auto-add hyperlinks (see example OnUserAddedXMLNode.sps)
- the Insert…, Insert After…, Insert Before… context menu items
- Append row, Insert row toolbar buttons
- Insert After…, Insert Before… actions in element entry helper (outside StyleVision)

The event doesn't get fired on Duplicate row, or when the node was added externally (e.g. via COM API), or on Apply (e.g. Text State Icons), or when in XML table operations or in DB operations.

The event parameter is the XML node object, which was added giving the user an opportunity to manipulate the XML node added. An elaborate example for an event handler can be found in the `OnUserAddedXMLNode.sps` file.

29.3.2.6.2 Application

Property: Application as Application[1580] (read-only)

Description
Accesses the XMLSpy application object.

Errors

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

29.3.2.6.3 AsXMLString

Property: AsXMLString as String

Description
Returns or sets the document content as an XML string. Setting the content to a new value does not change the schema file or sps file in use. If the new XMLString does not match the actual schema file error 2011 gets returned.

Errors
2000 The authentic view object is no longer valid.
2011 AsXMLString was set to a value which is no valid XML for the current schema file.

29.3.2.6.4 ContextMenu

Property: ContextMenu() as ContextMenu

Description
The property `ContextMenu` gives access to customize the context menu. The best place to do it is in the event handler `OnContextMenuActivated`.

Errors
2000 Invalid object.
2005 Invalid parameter.

29.3.2.6.5 CreateXMLNode

Method: CreateXMLNode (*nKind* as SPYXMLDataKind[1857]) as XMLData[1795]

Return Value
The method returns the new XMLData[1795] object.

Description
To create a new XMLData object use the CreateXMLNode() method.

Errors
2000 Invalid object.
2012 Cannot create XML node.

29.3.2.6.6 DisableAttributeEntryHelper

Method: DisableAttributeEntryHelper ()

Description

(C) 2015-2021 Altova GmbH

DisableAttributeEntryHelper() disables the attribute entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

29.3.2.6.7 DisableElementEntryHelper

Method: DisableElementEntryHelper()

Description
DisableElementEntryHelper() disables the element entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

29.3.2.6.8 DisableEntityEntryHelper

Method: DisableEntityEntryHelper()

Description
DisableEntityEntryHelper() disables the entity entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

29.3.2.6.9 DocumentBegin

Property: DocumentBegin as AuthenticRange[1606] (read-only)

Description
Retrieve a range object that points to the beginning of the document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.6.10 DocumentEnd

Property: DocumentEnd as AuthenticRange[1606] (read-only)

Description
Retrieve a range object that points to the end of the document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.6.11 DoNotPerformStandardAction

Method: DoNotPerformStandardAction ()

Description
`DoNotPerformStandardAction()` serves as cancel bubble for macros, and stops further execution after macro has finished.

Errors
2000 Invalid object.

29.3.2.6.12 EvaluateXPath

Method: EvaluateXPath (XMLData as XMLData[1795], strExpression as string) strValue as string

Return Value
The method returns a string

Description
`EvaluateXPath()` executes an XPath expression with the given XML context node. The result is returned as a string, in the case of a sequence it is a space-separated string.

Errors
2000 Invalid object.
2005 Invalid parameter.
2008 Internal error.
2013 XPath error.

29.3.2.6.13 Event

Property: Event as AuthenticEvent (read-only)

Description
This property gives access to parameters of the last event in the same way as OldAuthenticView.event does. Since all events for the scripting environment and external clients are now available with parameters this Event property should only be used from within IDE-Plugins.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.6.14 EventContext

Property: EventContext() as EventContext

Description
`EventContext` property gives access to the running macros context. See the `EventContext` 1603 interface description for more details.

Errors
2000 Invalid object.

29.3.2.6.15 GetToolbarButtonState

Method: GetToolbarButtonState (`ButtonIdentifier` as string) as `AuthenticToolbarButtonState`

Return Value
The method returns `AuthenticToolbarButtonState`

Description
`Get/SetToolbarButtonState` queries the status of a toolbar button, and lets the user disable or enable the button, identified via its button identifier (see list above 1641). One usage is to disable toolbar buttons permanently. Another usage is to put `SetToolbarButtonState` in the `OnSelectionChanged` event handler, as toolbar buttons are updated regularly when the selection changes in the document.

Toolbar button states are given by the listed enumerations 1846.

The default state means that the enable/disable of the button is governed by AuthenticView. When the user sets the button state to enable or disable, the button remains in that state as long as the user does not change it.

Errors
2000 Invalid object.
2005 Invalid parameter.
2008 Internal error.
2014 Invalid button identifier.

29.3.2.6.16 Goto

Method: Goto (*eKind* as SPYAuthenticElementKind 1845, *nCount* as Long, *eFrom* as *SPYAuthenticDocumentPosition* 1844) as *AuthenticRange* 1606

Description

Retrieve a range object that points to the beginning of the *nCount* element of type *eKind*. The start position is defined by the parameter *eFrom*. Use positive values for *nCount* to navigate to the document end. Use negative values to navigate towards the beginning of the document.

Errors

2000	The authentic view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before beginning of document.
2005	Invalid element kind specified. The document position to start from is not one of *spyAuthenticDocumentBegin* or *spyAuthenticDocumentEnd*. Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------
' Scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

On Error Resume Next
Dim objRange
' goto beginning of first table in document
Set objRange = objAuthenticView.Goto (spyAuthenticTable, 1, spyAuthenticDocumentBegin)
If (Err.number = 0) Then
        objRange.Select()
Else
        MsgBox "No table found in document"
End If
```

29.3.2.6.17 IsRedoEnabled

Property: IsRedoEnabled as Boolean (read-only)

Description

True if redo steps are available and Redo[1650] is possible.

Errors

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

29.3.2.6.18 IsUndoEnabled

Property: IsUndoEnabled as Boolean (read-only)

Description

True if undo steps are available and Undo[1651] is possible.

(C) 2015-2021 Altova GmbH

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.6.19 MarkupVisibility

Property: MarkupVisibility as SPYAuthenticMarkupVisibility 1845

Description
Set or get current visibility of markup.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid enumeration value was specified.
Invalid address for the return parameter was specified.

29.3.2.6.20 Parent

Property: Parent as Document 1668 (read-only)

Description
Access the document shown in this view.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.6.21 Print

Method: Print (*bWithPreview* as Boolean, *bPromptUser* as Boolean)

Description
Print the document shown in this view. If *bWithPreview* is set to *True*, the print preview dialog pops up. If *bPromptUser* is set to *True*, the print dialog pops up. If both parameters are set to *False*, the document gets printed without further user interaction.

Errors
2000 The authentic view object is no longer valid.

29.3.2.6.22 Redo

Method: Redo() as Boolean

Description
Redo the modification undone by the last undo command.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.6.23 Selection

Property: Selection as AuthenticRange[1606]

Description
Set or get current text selection in user interface.

Errors
2000 The authentic view object is no longer valid.
2002 No cursor selection is active.
2005 Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------
' Scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' if we are the end of the document, re-start at the beginning
If (objAuthenticView.Selection.IsEqual(objAuthenticView.DocumentEnd)) Then
        objAuthenticView.Selection = objAuthenticView.DocumentBegin
Else
        ' objAuthenticView.Selection = objAuthenticView.Selection.GotoNextCursorPosition()
        ' or shorter:
         objAuthenticView.Selection.GotoNextCursorPosition().Select
End If
```

29.3.2.6.24 SetToolbarButtonState

Method: SetToolbarButtonState (`ButtonIdentifier` as string, `AuthenticToolbarButtonState` state)

Description
`Get/SetToolbarButtonState` queries the status of a toolbar button, and lets the user disable or enable the button, identified via its button identifier (see list above[1641]). One usage is to disable toolbar buttons permanently. Another usage is to put `SetToolbarButtonState` in the `OnSelectionChanged` event handler, as toolbar buttons are updated regularly when the selection changes in the document.

(C) 2015-2021 Altova GmbH

Toolbar button states are given by the listed enumerations 1846.

The default state means that the enable/disable of the button is governed by AuthenticView. When the user sets the button state to enable or disable, the button remains in that state as long as the user does not change it.

Errors
2000 Invalid object.
2008 Internal error.
2014 Invalid button identifier.

29.3.2.6.25 Undo

Method: Undo() as Boolean

Description
Undo the last modification of the document from within this view.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.6.26 UpdateXMLInstanceEntities

Method: UpdateXMLInstanceEntities()

Description
Updates the internal representation of the declared entities, and refills the entry helper. In addition, the validator is reloaded, allowing the XML file to validate correctly. Please note that this may also cause schema files to be reloaded.

Errors
The method never returns an error.

Example

```
// ----------------------------------------
// Scripting environment - JavaScript
// ----------------------------------------
if(Application.ActiveDocument && (Application.ActiveDocument.CurrentViewMode == 4))
{
	var objDocType;
	objDocType = Application.ActiveDocument.DocEditView.XMLRoot.GetFirstChild(10);

	if(objDocType)
	{
		var objEntity = Application.ActiveDocument.CreateChild(14);
		objEntity.Name = "child";
		objEntity.TextValue = "SYSTEM \"child.xml\"";
```

```
            objDocType.AppendChild(objEntity);

            Application.ActiveDocument.AuthenticView.UpdateXMLInstanceEntities();
        }
}
```

29.3.2.6.27 WholeDocument

Property: WholeDocument as AuthenticRange[1606] (read-only)

Description
Retrieve a range object that selects the whole document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.6.28 XMLDataRoot

Property: XMLDataRoot as XMLData[1795] (read-only)

Description
Returns or sets the top-level XMLData element of the current document. This element typically describes the document structure and would be of kind spyXMLDataXMLDocStruct, spyXMLDataXMLEntityDocStruct or spyXMLDataDTDDocStruct..

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

29.3.2.7 CodeGeneratorDlg

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Properties and Methods

Standard automation properties
Application[1653]
Parent[1657]

Programming language selection properties
ProgrammingLanguage[1657]
TemplateFileName[1658]

Settings for C++ code
CPPSettings_DOMType[1654]
CPPSettings_LibraryType[1655]
CPPSettings_UseMFC[1655]

(C) 2015-2021 Altova GmbH

CPPSettings_GenerateVC6ProjectFile 1654
CPPSettings_GenerateVSProjectFile 1655

Settings for C# code
CSharpSettings_ProjectType 1655

Dialog handling for above code generation properties
PropertySheetDialogAction 1657

Output path selection properties
OutputPath 1656
OutputPathDialogAction 1656

Presentation of result
OutputResultDialogAction 1656

Description
Use this object to configure the generation of program code for schema files. The method GenerateProgramCode 1684 expects a CodeGeneratorDlg as parameter to configure code generation as well as the associated user interactions.

29.3.2.7.1 Application

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: Application as Application 1580 (read-only)

Description
Access the XMLSpy application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.7.2 CompatibilityMode (obsolete)

Property: CompatibilityMode as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Set to true to generate code compatible to XMLSpy 2005R3. Set to false to use newly added code-generation features.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

29.3.2.7.3 CPPSettings_DOMType

Property: CPPSettings_DOMType as SPYDOMType 1847
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

29.3.2.7.4 CPPSettings_GenerateVC6ProjectFile

Property: CPPSettings_GenerateVC6ProjectFile as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

29.3.2.7.5 CPPSettings_GenerateGCCMakefile

Property: CPPSettings_GenerateGCCMakefile as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Creates makefiles to compile the generated code under Linux with GCC.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

(C) 2015-2021 Altova GmbH

29.3.2.7.6 CPPSettings_GenerateVSProjectFile

Property: CSharpSettings_GenerateVSProjectFile as SPYProjectType 1851
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code. Only `spyVisualStudio2005Project (=4)` and `spyVisualStudio2008Project (=5)` and `spyVisualStudio2010Project (=6)` are valid project types.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

29.3.2.7.7 CPPSettings_LibraryType

Property: CPPSettings_LibraryType as SPYLibType 1849
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

29.3.2.7.8 CPPSettings_UseMFC

Property: CPPSettings_UseMFC as Boolean
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

29.3.2.7.9 CSharpSettings_ProjectType

Property: CSharpSettings_ProjectType as SPYProjectType 1851
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines the only setting to configure generation of C# code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

29.3.2.7.10 OutputPath

Property: OutputPath as String
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the base directory for all generated code.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.7.11 OutputPathDialogAction

Property: OutputPathDialogAction as SPYDialogAction 1846
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog for selecting the code generation output path gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current value of the OutputPath 1656 property as default. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

29.3.2.7.12 OutputResultDialogAction

Property: OutputResultDialogAction as SPYDialogAction 1846
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog that asks to show the result of the code generation process gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors

(C) 2015-2021 Altova GmbH

2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

29.3.2.7.13 Parent

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: Parent as Dialogs[1664] (read-only)

Description
Access the parent of the object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.7.14 ProgrammingLanguage

Property: ProgrammingLanguage as ProgrammingLanguage[1851]
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the output language for the code to be generated.

CAUTION: Setting this property to one of C++, C# or Java, changes the property TemplateFileName[1658] to the appropriate template file delivered with XMLSpy as well. If you want to generate C++, C# or Java code based on your own templates, set first the programming language and then select your template file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.7.15 PropertySheetDialogAction

Property: PropertySheetDialogAction as SPYDialogAction[1846]
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Defines how the sub-dialog that configures the code generation process gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current values as defaults. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
2200 The object is no longer valid.

2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

29.3.2.7.16 TemplateFileName

Property: TemplateFileName as String
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Selects the code generation template file. XMLSpy comes with template files for C++, C# or Java in the SPL folder of your installation directory.

Setting this property to one of the code generation template files of your XMLSpy installation automatically sets the ProgrammingLanguage[1657] property to its appropriate value.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.8 DatabaseConnection

Properties for import and export
File[1661] or
ADOConnection[1659] or
ODBCConnection[1662]

Properties for import only
DatabaseKind[1660]
SQLSelect[1663]
AsAttributes[1659]
ExcludeKeys[1661]
IncludeEmptyElements[1662]
NumberDateTimeFormat[1662]
NullReplacement[1662]
CommentIncluded[1659]

Properties for export only
CreateMissingTables[1660]
CreateNew[1660]
TextFieldLen[1664]
DatabaseSchema[1660]

Properties for XML Schema from DB Structure generation
PrimaryKeys[1663]
ForeignKeys[1661]
UniqueKeys[1664]
SchemaExtensionType[1663]
SchemaFormat[1663]
ImportColumnsType[1661]

(C) 2015-2021 Altova GmbH

Description
DatabaseConnection specifies the parameters for the database connection.

Please note that the properties of the DatabaseConnection interface are referring to the settings of the import and export dialogs of XMLSpy.

29.3.2.8.1 ADOConnection

Property: ADOConnection as String

Description
The property ADOConnection contains a connection string. Either use this property or ODBCConnection[1662] or File[1661] to refer to a database.

Errors
No error codes are returned.

Example

```
Dim objSpyConn As DatabaseConnection
Set objSpyConn = objSpy.GetDatabaseSettings

Dim objADO As DataLinks
Set objADO = CreateObject("DataLinks")

If Not (objADO Is Nothing) Then
        Dim objConn As Connection
        Set objConn = objADO.PromptNew
        objSpyConn.ADOConnection = objConn.ConnectionString
End If
```

29.3.2.8.2 AsAttributes

Property: AsAttributes as Boolean

Description
Set AsAttributes to true if you want to initialize all import fields to be imported as attributes. Default is false and will initialize all fields to be imported as elements. This property is used only in calls to Application.GetDatabaseImportElementList[1586].

Errors
No error codes are returned.

29.3.2.8.3 CommentIncluded

Property: CommentIncluded as Boolean

Description

This property tells whether additional comments are added to the generated XML. Default is true. This property is used only when importing from databases.

Errors
No error codes are returned.

29.3.2.8.4 CreateMissingTables

Property: CreateMissingTables as Boolean

Description
If CreateMissingTables is true, tables which are not already defined in the export database will be created during export. Default is true. This property is used only when exporting to databases.

Errors
No error codes are returned.

29.3.2.8.5 CreateNew

Property: CreateNew as Boolean

Description
Set CreateNew true if you want to create a new database on export. Any existing database will be overwritten. See also DatabaseConnection.File[1661]. Default is false. This property is used only when exporting to databases.

Errors
No error codes are returned.

29.3.2.8.6 DatabaseKind

Property: DatabaseKind as SPYDatabaseKind[1846]

Description
Select the kind of database that gets access. The default value is spyDB_Unspecified(7) and is sufficient in most cases. This property is used only when importing from databases.

Errors
No error codes are returned.

29.3.2.8.7 DatabaseSchema

Property: DatabaseSchema as String

Description
This property specifies the Schema used for export in Schema aware databases. Default is "". This property is used only when exporting to databases.

(C) 2015-2021 Altova GmbH

Errors
No error codes are returned.

29.3.2.8.8 ExcludeKeys

Property: ExcludeKeys as Boolean

Description
Set ExcludeKeys to true if you want to exclude all key columns from the import data. Default is false. This property is used only when importing from databases.

Errors
No error codes are returned.

29.3.2.8.9 File

Property: File as String

Description
The property File sets the path for the database during export or import. This property can only be used in conjunction with a Microsoft Access database. Either use this property or ODBCConnection[1662] or ADOConnection[1659] to refer to the database.

See also Import and Export[1552].

Errors
No error codes are returned.

29.3.2.8.10 ForeignKeys

Property: ForeignKeys as Boolean

Description
Specifies whether the Foreign Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

29.3.2.8.11 ImportColumnsType

Property: ImportColumnsType as SPYImportColumnsType[1848]

Description
Defines if column information from the DB is saved as element or attribute in the XML Schema. Default is as element. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

29.3.2.8.12 IncludeEmptyElements

Property: IncludeEmptyElements as Boolean

Description
Set IncludeEmptyElements to false if you want to exclude all empty elements. Default is true. This property is used only when importing from databases.

Errors
No error codes are returned.

29.3.2.8.13 NullReplacement

Property: NullReplacement as String

Description
This property contains the text value that is used during import for empty elements (null values). Default is "". This property is used only when importing from databases.

Errors
No error codes are returned.

29.3.2.8.14 NumberDateTimeFormat

Property: NumberDateTimeFormat as SPYNumberDateTimeFormat 1850

Description
The property NumberDateTimeFormat sets the format of numbers and date- and time-values. Default is spySystemLocale 1850. This property is used only when importing from databases.

Errors
No error codes are returned.

29.3.2.8.15 ODBCConnection

Property: ODBCConnection as String

Description
The property ODBCConnection contains a ODBC connection string. Either use this property or ADOConnection 1659 or File 1661 to refer to a database.

Errors
No error codes are returned.

(C) 2015-2021 Altova GmbH

29.3.2.8.16 PrimaryKeys

Property: PrimaryKeys as Boolean

Description
Specifies whether the Primary Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

29.3.2.8.17 SchemaExtensionType

Property: SchemaExtensionType as SPYSchemaExtensionType 1854

Description
Defines the Schema extension type used during the Schema generation. This property is used only when creating a XML Schema from a DB structure.

See also Create XML Schema from DB Structure 1339.

Errors
No error codes are returned.

29.3.2.8.18 SchemaFormat

Property: SchemaFormat as SPYSchemaFormat 1854

Description
Defines the Schema format used during the Schema generation. This property is used only when creating a XML Schema from a DB structure.

See also Create XML Schema from DB Structure 1339.

Errors
No error codes are returned.

29.3.2.8.19 SQLSelect

Property: SQLSelect as String

Description
The SQL query for the import is stored in the property SQLSelect. This property is used only when importing from databases. See also Import and Export 1552.

Errors
No error codes are returned.

29.3.2.8.20 TextFieldLen

Property: TextFieldLen as long

Description
The property TextFieldLen sets the length for created text fields during the export. Default is 255. This property is used only when exporting to databases.

Errors
No error codes are returned.

29.3.2.8.21 UniqueKeys

Property: UniqueKeys as Boolean

Description
Specifies whether the Unique Keys constraint is created or not. Default is true. This property is used only when creating a XML Schema from a DB structure.

Errors
No error codes are returned.

29.3.2.9 Dialogs

Properties and Methods

Standard automation properties
Application 1665
Parent 1665

Various dialog objects
CodeGeneratorDlg 1665
FileSelectionDlg 1665
SchemaDocumentationDlg 1666
GenerateSampleXMLDlg 1666
DTDSchemaGeneratorDlg 1666
FindInFilesDlg 1667
WSDLDocumentationDlg 1667
WSDL20DocumentationDlg 1667
XBRLDocumentationDlg 1667

Description
The Dialogs object provides access to different built-in dialogs of XMLSpy. These dialog objects allow to initialize the fields of user dialogs before they get presented to the user or allow to simulate complete user input by your program.

(C) 2015-2021 Altova GmbH

29.3.2.9.1 Application

Property: Application as Application[1580] (read-only)

Description
Access the XMLSpy application object.

Errors
2300 The object is no longer valid.
2301 Invalid address for the return parameter was specified.

29.3.2.9.2 CodeGeneratorDlg

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Property: CodeGeneratorDlg as CodeGeneratorDlg[1652] (read-only)

Description
Get a new instance of a code generation dialog object. You will need this object to pass the necessary parameters to the code generation methods. Initial values are taken from last usage of the code generation dialog.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

29.3.2.9.3 FileSelectionDlg

Property: FileSelectionDlg as FileSelectionDlg[1711] (read-only)

Description
Get a new instance of a file selection dialog object.

File selection dialog objects are passed to you with the some events that signal opening or saving of documents and projects.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

29.3.2.9.4 Parent

Property: Parent as Application[1580] (read-only)

Description

Access the XMLSpy application object.

Errors
2300 The object is no longer valid.
2301 Invalid address for the return parameter was specified.

29.3.2.9.5 SchemaDocumentationDlg

Property: SchemaDocumentationDlg as SchemaDocumentationDlg[1733] (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of schema documentation. See Document.GenerateSchemaDocumentation[1684] for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

29.3.2.9.6 GenerateSampleXMLDlg

Property: GenerateSampleXMLDlg as GenerateSampleXMLDlg[1724] (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of a sample XML based on a W3C schema or DTD. See GenerateSampleXML[1684] for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

29.3.2.9.7 DTDSchemaGeneratorDlg

Property: DTDSchemaGeneratorDlg as DTDSchemaGeneratorDlg[1702] (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of a schema or DTD. See Document.GenerateDTDOrSchemaEx[1683] for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.9.8 FindInFilesDlg

Property: FindInFilesDlg as FindInFilesDlg[1712] (read-only)

Description
Get a new instance of a dialog object that parameterizes the search (or replacement) of strings in files. See Application.FindInFiles[1586] for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

29.3.2.9.9 WSDLDocumentationDlg

Property: WSDLDocumentationDlg as WSDLDocumentationDlg[1761] (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of WSDL documentation. See Document.GenerateWSDLDocumentation[1685] for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

29.3.2.9.10 WSDL20DocumentationDlg

Property: WSDL20DocumentationDlg as WSDL20DocumentationDlg[1772] (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of WSDL 2.0 documentation. See Document.GenerateWSDL20Documentation[1685] for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.
2301 Invalid address for the return parameter was specified.

29.3.2.9.11 XBRLDocumentationDlg

Property: XBRLDocumentationDlg as XBRLDocumentationDlg[1784] (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of XBRL documentation. See Document.GenerateXBRLDocumentation[1685] for its usage.

Errors
2300 The Dialogs object or one of its parents is no longer valid.

2301 Invalid address for the return parameter was specified.

29.3.2.10 Document

Properties and Methods

Standard automation properties
Application 1672
Parent 1690

Various document properties and methods
SetActiveDocument 1693
Encoding 1679
SetEncoding (obsolete) 1693
Suggestions 1695

XML validation
IsValid 1688
SetExternalIsValid 1694

Document conversion and transformation
AssignDTD 1672
AssignSchema 1673
AssignXSL 1673
AssignXSLFO 1673
ConvertDTDOrSchema 1675
ConvertDTDOrSchemaEx 1675
GenerateDTDOrSchema 1683
GenerateDTDOrSchemaEx 1683
FlattenDTDOrSchema 1682
CreateSchemaDiagram 1677
ExecuteXQuery 1679
TransformXSL 1696
TransformXSLEx 1696
TransformXSLFO 1696
GenerateProgramCode 1684 (Enterprise Edition only)
GenerateSchemaDocumentation 1684
GenerateSampleXML 1684
GenerateWSDL20Documentation 1685
GenerateWSDLDocumentation 1685
GenerateXBRLDocumentation 1685
ConvertToWSDL20 1676

Document export
GetExportElementList 1686
ExportToText 1681
ExportToDatabase 1680
CreateDBStructureFromXMLSchema 1677
GetDBStructureList 1686

File saving and naming
FullName 1682
Name 1690

(C) 2015-2021 Altova GmbH

Path 1691
GetPathName (obsolete) 1687
SetPathName (obsolete) 1694
Title 1696
IsModified 1687
Saved 1692
SaveAs 1692
Save 1691
SaveInString 1692
SaveToURL 1693
Close 1675

View access
CurrentViewMode 1678
SwitchViewMode 1695
AuthenticView 1674
GridView 1687
DocEditView (obsolete) 1678

Access to XMLData
RootElement 1691
DataRoot 1678
CreateChild 1677
UpdateViews 1697
StartChanges 1695
EndChanges 1679
UpdateXMLData 1697

Description
Document objects represent XML documents opened in XMLSpy.

Use one of the following properties to access documents that are already open XMLSpy:
Application.ActiveDocument 1583
Application.Documents 1585

Use one of the following methods to open a new document in XMLSpy:
Documents.OpenFile 1700
Documents.OpenURL 1701
Documents.OpenURLDialog 1701
Documents.NewFile 1699
Documents.NewFileFromText 1700
SpyProjectItem.Open 1749
Application.ImportFromDatabase 1590
Application.ImportFromSchema 1591
Application.ImportFromText 1592
Application.ImportFromWord 1593
Document.ConvertDTDOrSchema 1675
Document.GenerateDTDOrSchema 1683

29.3.2.10.1 Events

29.3.2.10.1.1 OnBeforeSaveDocument

Event: OnBeforeSaveDocument(*objDocument* as Document[1668], *objDialog* as FileSelectionDlg[1711])

XMLSpy scripting environment - VBScript:
```
Function On_BeforeSaveDocument(objDocument, objDialog)
End Function

' old handler - now obsolete
' return string to save to new file name
' return empty string to cancel save operation
' return nothing to save to original name
Function On_SaveDocument(objDocument, strFilePath)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeSaveDocument(objDocument, objDialog)
{
}

// old handler - now obsolete
// return string to save to new file name
// return empty string to cancel save operation
// return nothing to save to original name
function On_SaveDocument(objDocument, strFilePath)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (27, ...)// nEventId = 27
```

Description
This event gets fired on any attempt to save a document. The file selection dialog object is initialized with the name chosen for the document file. You can modify this selection. To continue saving the document leave the FileSelectionDlg.DialogAction[1711] property of *io_objDialog* at its default value spyDialogOK[1846]. To abort saving of the document set this property to spyDialogCancel[1846].

29.3.2.10.1.2 OnBeforeCloseDocument

Event: OnBeforeCloseDocument(*objDocument* as Document[1668])as Boolean

XMLSpy scripting environment - VBScript:
```
Function On_BeforeCloseDocument(objDocument)
        ' On_BeforeCloseDocument = False  ' to prohibit closing of document
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeCloseDocument(objDocument)
```

```
{
        // return false;  /* to prohibit closing of document */
}
```

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (28, ...)// nEventId = 28

Description
This event gets fired on any attempt to close a document. To prevent the document from being closed return false.

29.3.2.10.1.3 OnBeforeValidate

Event: OnBeforeValidate(*objDocument* as Document[1668], *bOnLoading* as Boolean, *bOnCommand* as Boolean) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_BeforeValidate(objDocument, bOnLoading, bOnCommand)
        On_BeforeValidate = bCancelDefaultValidation    'set by the script if necessary
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeValidate(objDocument, bOnLoading, bOnCommand)
{
        return bCancelDefaultValidation  //set by the script if necessary
}
```

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (32, ...)// nEventId = 32

Description
This event gets fired before the document is validated. It is possible to suppress the default validation by returning false from the event handler. In this case the script should also set the validation result using the SetExternalIsValid[1694] method.

bOnLoading is true if the event is raised on the initial validation on loading the document.

bOnCommand is true whenever the user selected the Validate command from the Toolbar or menu.

Available with TypeLibrary version 1.5

29.3.2.10.1.4 OnCloseDocument

Event: OnCloseDocument(*objDocument* as Document[1668])

XMLSpy scripting environment - VBScript:

```
Function On_Close Document(objDocument)
End Function
```

XMLSpy scripting environment - JScript:

function On_Close Document(*objDocument*)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (8, ...) // nEventId = 8

Description
This event gets fired as a result of closing a document. Do not modify the document from within this event.

29.3.2.10.1.5 OnViewActivation

Event: OnViewActivation(*objDocument* as Document[1668], *eViewMode* as SPYViewModes[1856], *bActivated* as Boolean)

XMLSpy scripting environment - VBScript:
Function On_ViewActivation(*objDocument*, *eViewMode*, *bActivated*)
End Function

XMLSpy scripting environment - JScript:
function On_ViewActivation(*objDocument*, *eViewMode*, *bActivated*)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (29, ...)// nEventId = 29

Description
This event gets fired whenever a view of a document becomes visible (i.e. becomes the active view) or invisible (i.e. another view becomes the active view or the document gets closed). However, the first view activation event after a document gets opened cannot be received, since there is no document object to get the event from. Use the Application.OnDocumentOpened[1582] event instead.

29.3.2.10.2 Application

Property: Application as Application[1580] (read-only)

Description
Accesses the XMLSpy application object.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

29.3.2.10.3 AssignDTD

Method: AssignDTD(*strDTDFile* as String, *bDialog* as Boolean)

(C) 2015-2021 Altova GmbH

Description
The method places a reference to the DTD file "strDTDFile" into the document. Note that no error occurs if the file does not exist, or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign a DTD to the document.

29.3.2.10.4 AssignSchema

Method: AssignSchema (*strSchemaFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the schema file "strSchemaFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign a schema file to the document.

29.3.2.10.5 AssignXSL

Method: AssignXSL (*strXSLFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the XSL file "strXSLFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign an XSL file to the document.

29.3.2.10.6 AssignXSLFO

Method: AssignXSLFO (*strXSLFOFile* as String, *bDialog* as Boolean)

Description
The method places a reference to the XSLFO file "strXSLFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If bDialog is true XMLSpy presents a dialog to set the file.

Errors
1400 The object is no longer valid.
1409 You are not allowed to assign an XSL file to the document.

29.3.2.10.7 AsXMLString

Property: AsXMLString as String

Description
This property can be used to get or set the document content.

Errors
1400 The document object is no longer valid.
1404 Cannot create XMLData object.
1407 View mode cannot be switched.

29.3.2.10.8 AuthenticView

Method: AuthenticView as AuthenticView[1634] (read-only)

Description
Returns an object that gives access to properties and methods specific to Authentic view. The object returned is only valid if the current document is opened in Authentic view mode. The lifetime of an object ends with the next view switch. Any attempt to access objects or any of its children afterwards will result in an error indicating that the object is invalid.

AuthenticView[1634] and DocEditView[1821] both provide automation access to the Authentic view mode of XMLSpy. Functional overlap is intentional. A future version of Authentic View will include all functionality of DocEditView[1821] and its sub-objects, thereby making usage of DocEditView[1821] obsolete.

Errors
1400 The object is no longer valid.
1417 Document needs to be open in authentic view mode.

Examples

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' secure access to authentic view object
' ---------------------------------------
Dim objDocument
Set objDocument = Application.ActiveDocument
If (Not objDocument Is Nothing) Then
	' we have an active document, now check for view mode
	If (objDocument.CurrentViewMode <> spyViewAuthentic) Then
		If (Not objDocument.SwitchViewMode (spyViewAuthentic)) Then
			MsgBox "Active document does not support authentic view mode"
		Else
			' now it is safe to access the authentic view object
			Dim objAuthenticView
			Set objAuthenticView = objDocument.AuthenticView
			' now use the authentic view object

		End If
	End If
```

```
Else
        MsgBox "No document is open"
End If
```

29.3.2.10.9 Close

Method: Close (*bDiscardChanges* as Boolean)

Description
To close the document call this method. If bDiscardChanges is true and the document is modified, the document will be closed but not saved.

Errors

1400	The object is no longer valid.
1401	Document needs to be saved first.

29.3.2.10.10 ConvertDTDOrSchema

Method: ConvertDTDOrSchema (*nFormat* as SPYDTDSchemaFormat[1847], *nFrequentElements* as SPYFrequentElements[1848])

Parameters
nFormat
Sets the schema output format to DTD or W3C.

nFrequentElements
Create complex elements as elements or complex types.

Description
ConvertDTDOrSchema takes an existing schema format and converts it into a different format.
For a finer tuning of DTD/XSD conversion, use ConvertDTDOrSchemaEx[1675].

Errors

1400	The object is no longer valid.
1412	Error during conversion. In the case of DTD to DTD or XSD to XSD conversion, the following errors are returned: *DTD to DTD conversion is not supported. Please use function FlattenDTDOrSchema instead* and *Schema to schema conversion is not supported. Please use function FlattenDTDOrSchema instead.*

29.3.2.10.11 ConvertDTDOrSchemaEx

Method: ConvertDTDOrSchemaEx (*nFormat* as SPYDTDSchemaFormat[1847], *nFrequentElements* as SPYFrequentElements[1848], sOutputPath as String, nOutputPathDialogAction as SPYDialogAction[1846])

Parameters
nFormat
Sets the schema output format to DTD, or W3C.

nFrequentElements
Create complex elements as elements or complex types.

`sOutputPath`
The file path for the newly generated file.

`nOutputPathDialogAction`
Defines the dialog interaction for this call.

Description
`ConvertDTDOrSchemaEx` takes an existing schema format and converts it into a different format.

Errors

1400	The object is no longer valid.
1412	Error during conversion. In the case of DTD to DTD or XSD to XSD conversion, the following errors are returned: *DTD to DTD conversion is not supported. Please use function FlattenDTDOrSchema instead* and *Schema to schema conversion is not supported. Please use function FlattenDTDOrSchema instead*.

29.3.2.10.12 ConvertToWSDL20

Method: ConvertToWSDL20 (*sFilePath* as String, *bShowDialogs* as Boolean)

Parameters
sFilePath
This specifies the file name of the converted WSDL. In case the source WSDL includes files which also must be converted, then only the directory part of the given path is used and the file names are generated automatically.

bShowDialogs
Defines whether file/folder selection dialogs are shown.

Description
Converts the WSDL 1.1 document to a WSDL 2.0 file. It will also convert any referenced WSDL files that are referenced from within this document. Note that this functionality is limited to WSDL View only. See Document.CurrentViewMode 1678. and SPYViewModes 1856.

Errors

1400	The document object is no longer valid.
1407	Invalid parameters have been passed or an empty file name has been specified as output target.
1417	The document is not opened in WSDL view, maybe it is not an '.wsdl' file.
1421	Feature is not available in this edition.

(C) 2015-2021 Altova GmbH

1433 WSDL 1.1 to WSDL 2.0 conversion failed.

29.3.2.10.13 CreateChild

Method: CreateChild (*nKind* as SPYXMLDataKind[1857]) as XMLData[1795]

Return Value
The method returns the new XMLData object.

Description
To create a new XMLData object use the CreateChild() method.

Errors
1400 The object is no longer valid.
1404 Cannot create XMLData object.
1407 Invalid address for the return parameter was specified.

29.3.2.10.14 CreateDBStructureFromXMLSchema

Method: CreateDBStructureFromXMLSchema (*pDatabase* as DatabaseConnection[1658], *pTables* as ElementList[1706], *bDropTableWithExistingName* as Boolean) as String

Description
CreateDBStructureFromXMLSchema exports the given tables to the specified database. The function returns the SQL statements that were necessary to perform the changes.

See also GetDBStructureList[1686].

Errors
1429 Database selection missing.
1430 Document export failed.

29.3.2.10.15 CreateSchemaDiagram

Method: CreateSchemaDiagram (*nKind* as SPYSchemaDefKind[1853], *strName* as String, *strFile* as String)

Return Value
None.

Description

The method creates a diagram of the schema type strName of kind nKind and saves the output file into strFile. Note that this functionality is limited to Schema View only. See Document.CurrentViewMode[1678]. and SPYViewModes[1856].

Errors
1400 The object is no longer valid.
1414 Failed to save diagram.
1415 Invalid schema definition type specified.

29.3.2.10.16 CurrentViewMode

Method: CurrentViewMode as SPYViewModes[1856]

Description
The property holds the current view mode of the document. See also Document.SwitchViewMode[1695].

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

29.3.2.10.17 DataRoot

Property: DataRoot as XMLData[1795] (read-only)

Description
This property provides access to the document's first XMLData object of type *spyXMLDataElement*. This is typically the root element for all document content data. See XMLSpyDocument.RootElement[1691] to get the root element of the whole document including XML prolog data. If the CurrentViewMode[1678] is not *spyViewGrid* or *spyViewAuthentic* an UpdateXMLData[1697] may be necessary to get access to the latest XMLData[1795].

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

29.3.2.10.18 DocEditView

Method: DocEditView as DocEditView

Description
Holds a reference to the current Authentic View object.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Document needs to be open in authentic view mode.

(C) 2015-2021 Altova GmbH

29.3.2.10.19 Encoding

Property: Encoding as String

Description
This property provides access to the document's encoding value. However, this property can only be accessed when the document is opened in *spyViewGrid*, *spyViewText* or *spyViewAuthentic*. See CurrentViewMode[1678] on how to detect a document's actual view mode.

This property makes the method SetEncoding[1693] obsolete.

Possible values are, for example:

8859-1,
8859-2,
ASCII, ISO-646,
850,
1252,
1255,
SHIFT-JIS, MS-KANJI,
BIG5, FIVE,
UTF-7,
UTF-8,
UTF-16

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.
1416 Operation not supported in current view mode.

29.3.2.10.20 EndChanges

Method: EndChanges()

Description
Use the method EndChanges to display all changes since the call to Document.StartChanges[1695].

Errors
1400 The object is no longer valid.

29.3.2.10.21 ExecuteXQuery

Method: ExecuteXQuery (*strXMLFileName* as String)

Description

Execute the XQuery statements contained in the document of the document object. Either an XQuery execution or an XQuery Update is performed depending on the file extension of the document. Use the XML file specified in the argument as the XML target document that the XQuery document processes.

- If the document has an XQuery file extension as defined in the Options dialog of XMLSpy, then an XQuery execution is performed. By default: `.xq`, `.xql`, and `.xquery` are set as XQuery file extensions in XMLSpy.
- If the document has an XQuery Update file extension as defined in the Options dialog of XMLSpy, then an XQuery Update action is performed. By default: `.xqu` is set as an XQuery Update file extension in XMLSpy.

If your XQuery script does not use an XML source, set the parameter `strXMLFileName` to an empty string.

Errors

1400 The document object is no longer valid.
1423 XQuery transformation error.
1424 Not all files required for operation could be loaded. Most likely, the file specified in `strXMLFileName` does not exist or is not valid.

29.3.2.10.22 ExportToDatabase

Method: ExportToDatabase (*pFromChild* as XMLData[1795], *pExportSettings* as ExportSettings[1708], *pDatabase* as DatabaseConnection[1653])

Description

ExportToDatabase exports the XML document starting with the element pFromChild. The parameter pExportSettings defines the behaviour of the export (see Application.GetExportSettings[1588]). The parameter pDatabase specifies the destination of the export (see Application.GetDatabaseSettings[1587]). `UpdateXMLData()`[1697] might be indirectly needed as you have to pass the `XMLData`[1795] as parameter to this function.

Errors

1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1416 Error during export.
1429 Database selection missing.
1430 Document export failed.

Example

```
Dim objDoc As Document
Set objDoc = objSpy.ActiveDocument

'set the behaviour of the export with ExportSettings
Dim objExpSettings As ExportSettings
Set objExpSettings = objSpy.GetExportSettings

'set the destination with DatabaseConnection
Dim objDB As DatabaseConnection
```


(C) 2015-2021 Altova GmbH

```
Set objDB = objSpy.GetDatabaseSettings

objDB.CreateMissingTables = True
objDB.CreateNew = True
objDB.File = "C:\Export.mdb"

objDoc.ExportToDatabase objDoc.RootElement, objExpSettings, objDB
If Err.Number <> 0 Then
        a = MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) &
                "Description: " & Err.Description)
End If
```

29.3.2.10.23 ExportToText

Method: ExportToText (*pFromChild* as XMLData[1795], *pExportSettings* as ExportSettings[1708], *pTextSettings* as TextImportExportSettings[1753])

Description
ExportToText exports tabular information from the document starting at pFromChild into one or many text files. Columns of the resulting tables are generated in alphabetical order of the column header names. Use GetExportElementList[1686] to learn about the data that will be exported. The parameter pExportSettings defines the specifics for the export. Set the property ExportSettings.ElementList[1709] to the - possibly modified - list returned by GetExportElementList[1686] to avoid exporting all contained tables. The parameter pTextSettings defines the options specific to text export and import. You need to set the property TextImportExportSettings.DestinationFolder[1753] before you call ExportToText. `UpdateXMLData()`[1697] might be indirectly needed as you have to pass the `XMLData`[1795] as parameter to this function.

See also Import and export of data[1552].

Errors

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.
1416	Error during export.
1430	Document export failed.

Example

```
' ---------------------------------------------------------
' VBA client code fragment - export document to text files
' ---------------------------------------------------------
        Dim objDoc As Document
        Set objDoc = objSpy.ActiveDocument

        Dim objExpSettings As ExportSettings
        Set objExpSettings = objSpy.GetExportSettings
        objExpSettings.ElementList = objDoc.GetExportElementList(
                                                        objDoc.RootElement,
                                                        objExpSettings)

        Dim objTextExp As TextImportExportSettings
        Set objTextExp = objSpy.GetTextImportExportSettings
        objTextExp.HeaderRow = True
        objTextExp.DestinationFolder = "C:\Exports"
```

```
On Error Resume Next
objDoc.ExportToText objDoc.RootElement, objExpSettings, objTextExp

If Err.Number <> 0 Then
        a = MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) & "Description: "
        & Err.Description)
End If
```

29.3.2.10.24 FlattenDTDOrSchema

Method: FlattenDTDOrSchema (sOutputPath as String, nOutputPathDialogAction as SPYDialogAction[1846])

Parameters

`sOutputPath`
The file path for the newly generated file.

`nOutputPathDialogAction`
Defines the dialog interaction for this call.

Description

`FlattenDTDOrSchema` takes an existing DTD or schema, generates a flattened file, and saves the generated file at the specified location. In the case of DTDs, flattening removes parameter entities and produces a single DTD from a collection of modules; sections marked `IGNORE` are suppressed and unused parameter entities are deleted. When an XML Schema is flattened, (i) the components of all included schemas are added as global components of the active schema, and (ii) included schemas are deleted.

Errors

1400 The object is no longer valid.
1412 Error during conversion.

29.3.2.10.25 FullName

Property: FullName as String

Description

This property can be used to get or set the full file name - including the path - to where the document gets saved. The validity of the name is not verified before the next save operation.

This property makes the methods GetPathName[1687] and SetPathName[1694] obsolete.

Errors

1400 The document object is no longer valid.
1402 Empty string has been specified as full file name.

(C) 2015-2021 Altova GmbH

29.3.2.10.26 GenerateDTDOrSchema

Method: GenerateDTDOrSchema (*nFormat* as SPYDTDSchemaFormat[1847], *nValuesList* as integer, *nDetection* as SPYTypeDetection[1855], *nFrequentElements* as SPYFrequentElements[1848])

Parameters
nFormat
Sets the schema output format to DTD, or W3C.

nValuesList
Generate not more than this amount of enumeration-facets per type. Set to -1 for unlimited.

nDetection
Specifies granularity of simple type detection.

nFrequentElements
Shall the types for all elements be defined as global? Use the value *spyGlobalComplexType* to define them on global scope. Otherwise, use the value *spyGlobalElements*.

Description
Use this method to automatically generate a DTD or schema for the current XML document.
For a finer tuning of DTD / schema generation, use GenerateDTDOrSchemaEx[1683].
Note that this functionality is not available in ZIP View only. See Document.CurrentViewMode[1678]. and SPYViewModes[1856].

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

29.3.2.10.27 GenerateDTDOrSchemaEx

Method: GenerateDTDOrSchemaEx (*objDlg* as DTDSchemaGeneratorDlg[1702]) as Document[1668]

Description
Use this method to automatically generate a DTD or schema for the current XML document. A DTDSchemaGeneratorDlg[1702] object is used to pass information to the schema/DTD generator. The generation process can be configured to allow user interaction or run without further user input.
Note that this functionality is not available in ZIP View only. See Document.CurrentViewMode[1678]. and SPYViewModes[1856].

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

29.3.2.10.28 GenerateProgramCode

Method: GenerateProgramCode (*objDlg* as CodeGeneratorDlg[1652])
Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Description
Generate Java, C++ or C# class files from the XML Schema definitions in your document. A CodeGeneratorDlg[1652] object is used to pass information to the code generator. The generation process can be configured to allow user interaction or run without further user input.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1421 Feature not available in this edition

29.3.2.10.29 GenerateSampleXML

Method: GenerateSampleXML (*objDlg* as GenerateSampleXMLDlg[1724]) as Document[1668]

Description
Generates a sample XML if the document is a schema or DTD. Use Dialogs.GenerateSampleXMLDlg[1666] to get an initialized set of options.

Available with TypeLibrary version 1.5

Errors
1400 The document object is no longer valid.

29.3.2.10.30 GenerateSchemaDocumentation

Method: GenerateSchemaDocumentation (*objDlg* as SchemaDocumentationDlg[1733])

Description
Generate documentation for a schema definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.SchemaDocumentationDlg[1666] to get an initialized set of options. As a minimum, you will need to set the property SchemaDocumentationDlg.OutputFile[1741] before starting the generation process. Note that this functionality is limited to Schema View only. See Document.CurrentViewMode[1678] and SPYViewModes[1856].

Errors
1400 The document object is no longer valid.
1407 Invalid parameters have been passed or an empty file name has been specified as output target.
1417 The document is not opened in schema view, maybe it is not an '.xsd' file.
1421 Feature is not available in this edition.
1422 Error during generation

(C) 2015-2021 Altova GmbH

29.3.2.10.31 GenerateWSDL20Documentation

Method: GenerateWSDL20Documentation (*objDlg* as WSDL20DocumentationDlg[1772])

Description
Generate documentation for a WSDL definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.WSDL20DocumentationDlg[1667] to get an initialized set of options. As a minimum, you will need to set the property WSDL20DocumentationDlg.OutputFile[1778] before starting the generation process. Note that this functionality is limited to WSDL View only. See Document.CurrentViewMode[1678] and SPYViewModes[1856].

Errors

1400	The document object is no longer valid.
1407	Invalid parameters have been passed or an empty file name has been specified as output target.
1417	The document is not opened in schema view, maybe it is not an '.xsd' file.
1421	Feature is not available in this edition.
1422	Error during generation

29.3.2.10.32 GenerateWSDLDocumentation

Method: GenerateWSDLDocumentation (*objDlg* as WSDLDocumentationDlg[1761])

Description
Generate documentation for a WSDL definition file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.WSDLDocumentationDlg[1667] to get an initialized set of options. As a minimum, you will need to set the property WSDLDocumentationDlg.OutputFile[1767] before starting the generation process. Note that this functionality is limited to WSDL View only. See Document.CurrentViewMode[1678] and SPYViewModes[1856].

Errors

1400	The document object is no longer valid.
1407	Invalid parameters have been passed or an empty file name has been specified as output target.
1417	The document is not opened in schema view, maybe it is not an '.xsd' file.
1421	Feature is not available in this edition.
1422	Error during generation

29.3.2.10.33 GenerateXBRLDocumentation

Method: GenerateXBRLDocumentation (*objDlg* as XBRLDocumentationDlg[1784])

Description

Generate documentation for an XBRL file in HTML, MS-Word, or RTF format. The parameter objDlg is used to parameterize the generation process. Use Dialogs.XBRLDocumentationDlg[1667] to get an initialized set of options. As a minimum, you will need to set the property XBRLDocumentationDlg.OutputFile[1789] before starting the generation process. Note that this functionality is limited to XBRL View only. See Document.CurrentViewMode[1678] and SPYViewModes[1856].

Errors

1400	The document object is no longer valid.
1407	Invalid parameters have been passed or an empty file name has been specified as output target.
1417	The document is not opened in schema view, maybe it is not an '.xsd' file.
1421	Feature is not available in this edition.
1422	Error during generation

29.3.2.10.34 GetDBStructureList

Method: GetDBStructureList (*pDatabase* as DatabaseConnection[1658]) as ElementList[1706]

Description
GetDBStructureList creates a collection of elements from the Schema document for which tables in the specified database are created. The function returns a collection of ElementListItems where the properties ElementListItem.Name[1708] contain the names of the tables.

See also CreateDBStructureFromXMLSchema[1677].

Errors

1400	The object is no longer valid.
1427	Failed creating parser for the specified XML.
1428	Export of element list failed.
1429	Database selection missing.

29.3.2.10.35 GetExportElementList

Method: GetExportElementList (*pFromChild* as XMLData[1795], *pExportSettings* as ExportSettings[1708]) as ElementList[1706]

Description
GetExportElementList creates a collection of elements to export from the document, depending on the settings in pExportSettings and starting from the element pFromChild. The function returns a collection of ElementListItems where the properties ElementListItem.Name[1708] contain the names of the tables that can be exported from the document. The property ElementListItem.FieldCount[1708] contains the number of columns in the table. The property ElementListItem.RecordCount[1708] contains the number of records in the table. The property ElementListItem.ElementKind[1707] is unused. `UpdateXMLData()`[1697] might be indirectly needed as you have to pass the `XMLData`[1795] as parameter to this function.

See also Import and export of data[1552].

Errors

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.
1427	Failed creating parser for the specified XML.
1428	Export of element list failed.

29.3.2.10.36 GetPathName (obsolete)

Superseded by Document.FullName 1682

```
// ----- javascript sample -----
// instead of:
// strPathName = Application.ActiveDocument.GetPathName();
// use now:
strPathName = Application.ActiveDocument.FullName;
```

Method: GetPathName() as String

Description
The method GetPathName gets the path of the active document.

See also Document.SetPathName 1694 (obsolete).

29.3.2.10.37 GridView

Property: GridView as GridView 1730

Description
This property provides access to the grid view functionality of the document.

Errors

1400	The object is no longer valid.
1407	Invalid address for the return parameter was specified.
1417	Document needs to be open in enhanced grid view mode.

29.3.2.10.38 IsModified

Property: IsModified as Boolean

Description
True if the document is modified.

Errors

1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

29.3.2.10.39 IsValid

Method: HRESULT IsValid([in, out] VARIANT *strError, [in, out] VARIANT *nErrorPos, [in, out] VARIANT *pBadData, [out,retval] VARIANT_BOOL *bValid);

Return Value
True if the document is valid, false if not. To call `IsValid()`, the application GUI must be visible. (If you wish to validate without the GUI being visible, please use Altova RaptorXML Server.)

Description
`IsValid` validates the document against its associated schema or DTD. `strError` gives you the same error message as when you validate the file within the GUI.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1408 Unable to validate file.

Example
The following C++ code snippet provides an example of how to use the **`IsValid`** method.

```
#import "XMLSpy.tlb"


CComPtr< XMLSpyLib::IDocument12> ipDoc = ipXMLSpy->GetActiveDocument();
if ( ipDoc )
{
	// prepare in/out parameters for IsValid call
	CComVariant variantError;
	CComVariant variantErrorPos;
	CComVariant variantBadData;
	// IsValid always shows a dialog with the validation result. This cannot be turned
off.
	bool bIsValid = ipDoc->IsValid( &variantError, &variantErrorPos, &variantBadData )
== VARIANT_TRUE;

	if ( !bIsValid )
	{
		// retrieve values from out parameters
		CString strError = (V_VT( &variantError ) == VT_BSTR ?
V_BSTR( &variantError ) : _T( "" ));
		long npos = (V_VT( &variantErrorPos ) == VT_I4 ? V_I4( &variantErrorPos ) :
-1);
		CComQIPtr< XMLSpyLib::IXMLData > ipXMLBadData = (V_VT( &variantBadData ) ==
VT_DISPATCH ? V_DISPATCH( &variantBadData ) : nullptr);

		if ( ipXMLBadData )
```

```
                        strError += CString( _T("\n\n Node: ") ) + (LPCWSTR)ipXMLBadData-
>GetName();

                if ( !strError.IsEmpty() )
                        AfxMessageBox( "Validation failed - " + strError );
        }
}
```

29.3.2.10.40 IsValidEx

Method: IsValidEx (*nXSDVersion* as SPYValidateXSDVersion[1855], *nErrorLimit* as `int`, *nErrorFormat* as SPYValidateErrorFormat[1855], out *strError* as Variant) as Boolean

Return Value
True if the document is valid, false if not.

Description
`IsValidEx` validates the document against its associated schema or DTD.

In parameters:
`nXSDVersion` which is an enumeration value of SPYValidateXSDVersion[1855] that selects the XSD version to validate against.
`nErrorLimit` which is an integer. Values must be `1` to `999`.
`nErrorFormat` which is an enumeration value of SPYValidateErrorFormat[1855] that selects the XSD version to validate against.

Out parameter:
`strError` is the error message, and is the same as that received when validating the file within the GUI.

Errors

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.
1408	Unable to validate file.

Example
The following C++ code snippet provides an example of how to use the **`IsValidEx`** method.

```
#import "XMLSpy.tlb"


CComPtr< XMLSpyLib::IDocument12> ipDoc = ipXMLSpy->GetActiveDocument();
if ( ipDoc )
{
        CComVariant variantErrorEx;
        bool bIsValidEx = ipDoc->IsValidEx( XMLSpyLib::spyValidateXSDVersion_AutoDetect,
100, XMLSpyLib::SPYValidateErrorFormat_LongXML, &variantErrorEx ) == VARIANT_TRUE;

        //      // retrieve values from out parameters
```

```
        CString strErrorEx = (V_VT( &variantErrorEx ) == VT_BSTR ?
V_BSTR( &variantErrorEx ) : _T( "" ));
        if ( !strErrorEx.IsEmpty() )
                AfxMessageBox( "Validation failed - " + strErrorEx );
}
```

29.3.2.10.41 IsWellFormed

Method: IsWellFormed (*pData* as XMLData, *bWithChildren* as Boolean, *strError* as Variant, *nErrorPos* as Variant, *pBadXMLData* as Variant) as Boolean

Return Value
True if the document is well formed.

Description
IsWellFormed checks the document for well-formedness starting at the element pData.

If the document is not well formed, strError contains an error message, nErrorPos the position in the file and pBadXMLData holds a reference to the element which breaks the well-formedness. These out-parameters are defined as VARIANTs to support scripting languages like VBScript.

Errors

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.

Example

See IsValid.

29.3.2.10.42 Name

Property: Name as String (read-only)

Description
Use this property to retrieve the name - not including the path - of the document file. To change the file name for a document use the property FullName[1682].

Errors

1400	The document object is no longer valid.
1407	Invalid address for the return parameter was specified.

29.3.2.10.43 Parent

Property: Parent as Documents[1698] (read-only)

Description

Access the parent of the document object.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Property: Parent as Application[1580] (read-only)

29.3.2.10.44 Path

Property: Path as String (read-only)

Description
Use this property to retrieve the path - not including the file name - of the document file. To change the file name and path for a document use the property FullName[1682].

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

29.3.2.10.45 RootElement

Property: RootElement as XMLData[1795] (read-only)

Description
The property RootElement provides access to the root element of the XML structure of the document including the XML prolog data. To access the first element of a document's content navigate to the first child of kind *spyXMLDataElement* or use the Document.DataRoot[1678] property. If the CurrentViewMode[1678] is not *spyViewGrid* or *spyViewAuthentic* an UpdateXMLData[1697] may be necessary to get access to the latest XMLData[1795].

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

29.3.2.10.46 Save

Method: Save()

Description
The method writes any modifications of the document to the associated file. See also Document.FullName[1682].

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1403 Error when saving file, probably the file name is invalid.

29.3.2.10.47 SaveAs

Method: SaveAs (*strFileName* as String)

Description
Save the document to the file specified. If saving was successful, the FullName[1682] property gets set to the specified file name.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1403 Error when saving file, probably the file name is invalid.

29.3.2.10.48 Saved

Property: Saved as Boolean (read-only)

Description
This property can be used to check if the document has been saved after the last modifications. It returns the negation of IsModified[1687].

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

29.3.2.10.49 SaveInString

Method: SaveInString (*pData* as XMLData[1795], *bMarked* as Boolean) as String

Parameters
pData
XMLData element to start. Set pData to Document.RootElement[1691] if you want to copy the complete file.

bMarked
If bMarked is true, only the elements selected in the grid view are copied.

Return Value
Returns a string with the XML data.

Description
SaveInString starts at the element pData and converts the XMLData objects to a string representation. `UpdateXMLData()`[1697] might be indirectly needed as you have to pass the `XMLData`[1795] as parameter to this function.

Errors
1400 The object is no longer valid.

(C) 2015-2021 Altova GmbH

1407 Invalid parameter or invalid address for the return parameter was specified.

29.3.2.10.50 SaveToURL

Method: SaveToURL (*strURL* as String, *strUser* as String, *strPassword* as String)

Return Value

Description
SaveToURL() writes the document to the URL strURL. This method does not set the permanent file path of the document.

Errors
1400 The object is no longer valid.
1402 Invalid URL specified.
1403 Error while saving to URL.

29.3.2.10.51 SetActiveDocument

Method: SetActiveDocument()

Description
The method sets the document as the active and brings it to the front.

Errors
1400 The object is no longer valid.

29.3.2.10.52 SetEncoding (obsolete)

Superseded by Document.Encoding 1679

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetEncoding("UTF-16");
// use now:
Application.ActiveDocument.Encoding = "UTF-16";
```

Method: SetEncoding (*strEncoding* as String)

Description
SetEncoding sets the encoding of the document like the menu item "File/Encoding..." in XMLSpy. Possible values for strEncoding are, for example:

8859-1,

8859-2,
ASCII, ISO-646,
850,
1252,
1255,
SHIFT-JIS, MS-KANJI,
BIG5, FIVE,
UTF-7,
UTF-8,
UTF-16

29.3.2.10.53 SetExternalIsValid

Method: SetExternalIsValid (*bValid* as Boolean)

Parameters

bValid
Sets the result of an external validation process.

Description
The internal information set by this method is only queried on cancelling the default validation in any OnBeforeValidate[1671] handler.

Available with TypeLibrary version 1.5

Errors
1400 The object is no longer valid.

29.3.2.10.54 SetPathName (obsolete)

Superseded by Document.FullName[1682]

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetPathName("C:\\myXMLFiles\\test.xml");
// use now:
Application.ActiveDocument.FullName = "C:\\myXMLFiles\\test.xml";
```

Method: SetPathName (*strPath* as String)

Description
The method SetPathName sets the path of the active document. SetPathName only copies the string and does not check if the path is valid. All succeeding save operations are done into this file.

(C) 2015-2021 Altova GmbH

29.3.2.10.55 StartChanges

Method: StartChanges()

Description
After StartChanges is executed XMLSpy will not update its editor windows until Document.EndChanges[1679] is called. This increases performance of complex tasks to the XML structure.

Errors
1400 The object is no longer valid.

29.3.2.10.56 Suggestions

Property: Suggestions as Array

Description
This property contains the last valid user suggestions for this document. The XMLSpy generated suggestions can be modified before they are shown to the user in the OnBeforeShowSuggestions[1756] event.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.

29.3.2.10.57 SwitchViewMode

Method: SwitchViewMode (*nMode* as SPYViewModes[1856]) as Boolean

Return value
Returns true if view mode is switched.

Description
The method sets the current view mode of the document in XMLSpy. See also Document.CurrentViewMode[1678] .

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Invalid view mode specified.

29.3.2.10.58 TextView

Property: TextView as TextView[1755]

Description
This property provides access to the text view functionality of the document.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

29.3.2.10.59 Title

Property: Title as String (read-only)

Description
Title contains the file name of the document. To get the path and filename of the file use FullName[1682].

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

29.3.2.10.60 TransformXSL

Method: TransformXSL()

Description
TransformXSL processes the XML document via the associated XSL file. See Document.AssignXSL[1673] on how to place a reference to a XSL file into the document.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

29.3.2.10.61 TransformXSLEx

Method: TransformXSLEx(*nAction* as SPYDialogAction[1846])

Description
TransformXSLEx processes the XML document via the associated XSL file. The parameter specifies whether a dialog asking for the result document name should pop up or not. See Document.AssignXSL[1673] on how to place a reference to a XSL file into the document.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

29.3.2.10.62 TransformXSLFO

Method: TransformXSLFO()

(C) 2015-2021 Altova GmbH

Description
TransformXSLFO processes the XML document via the associated XSLFO file. See AssignXSLFO[1673] on how to place a reference to a XSLFO file into the document. You need to assign a FOP processor to XMLSpy before you can use this method.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

29.3.2.10.63 TreatXBRLInconsistenciesAsErrors

Property: TreatXBRLInconsistenciesAsErrors as Boolean

Description
If this is set to `true` the `Document.IsValid()` method will return `false` for XBRL instances containing inconsistencies as defined by the XBRL Specification. The default value of this property is `false`.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

29.3.2.10.64 UpdateViews

Method: UpdateViews()

Description
To redraw the Enhanced Grid View and the Tree View call UpdateViews. This can be important after you changed the XMLData structure of a document. This method does not redraw the text view of XMLSpy.

Errors
1400 The document object is no longer valid.

29.3.2.10.65 UpdateXMLData

Method: UpdateXMLData() as Boolean

Description
The XMLData[1795] tree is updated from the current view. Please note that this can fail in case of the TextView if the current XML text is not well-formed. This is not necessary if CurrentViewMode[1678] is *spyViewGrid* or *spyViewAuthentic* because these views keep the XMLData[1795] updated.

Available with TypeLibrary version 1.5

Errors
1400 The document object is no longer valid.

29.3.2.11 Documents

Properties
Count 1699
Item 1699

Methods
NewAuthenticFile 1699
NewFile 1699
NewFileFromText 1700
OpenAuthenticFile 1700
OpenFile 1700
OpenURL 1701
OpenURLDialog 1701

Description
This object represents the set of documents currently open in XMLSpy. Use this object to open further documents or iterate through already opened documents.

Examples

```
' ----------------------------------------
' XMLSpy scripting environment - VBScript
' iterate through open documents
' ----------------------------------------
Dim objDocuments
Set objDocuments = Application.Documents

For Each objDoc In objDocuments
        'do something useful with your document
        objDoc.SetActiveDocument()
Next
```

```
// ----------------------------------------
// XMLSpy scripting environment - JScript
// close all open documents
// ----------------------------------------
for (var iter = new Enumerator (Application.Documents);
    ! iter.atEnd();
    iter.moveNext())
{
        // MsgBox ("Closing file " + iter.item().Name);
        iter.item().Close (true);
}
```


(C) 2015-2021 Altova GmbH

29.3.2.11.1 Count

Property: Count as long

Description
Count of open documents.

Errors
1600 Invalid `Documents` object
1601 Invalid input parameter

29.3.2.11.2 Item

Method: Item (*n* as long) as Document[1668]

Description
Gets the document with the index n in this collection. Index is 1-based.

Errors
1600 Invalid `Documents` object
1601 Invalid input parameter

29.3.2.11.3 NewAuthenticFile

Method: NewAuthenticFile (*strSPSPath* as String, *strXMLPath* as String) as Document[1668]

Parameters
strSPSPath
The path to the SPS document.

strXMLPath
The new XML document name.

Return Value
The method returns the new document.

Description
NewAuthenticFile creates a new XML file and opens it in Authentic View using SPS design strSPSPath.

29.3.2.11.4 NewFile

Method: NewFile (*strFile* as String, *strType* as String) as Document[1668]

Parameters
strFile

Full path of new file.

strType
Type of new file as string (i.e. "xml", "xsd", ...)

Return Value
Returns the new file.

Description
NewFile creates a new file of type strType (i.e. "xml"). The newly created file is also the ActiveDocument.

29.3.2.11.5 NewFileFromText

Method: NewFileFromText (*strText* as String, *strType* as String) as Document[1668]

Parameters
strText
The content of the new document in plain text.

strType
Type of the document to create (i.e. "xml").

Return Value
The method returns the new document.

Description
NewFileFromText creates a new document with strText as its content.

29.3.2.11.6 OpenAuthenticFile

Method: OpenAuthenticFile (*strSPSPath* as String, *strXMLPath* as String) as Document[1668]

Parameters
strSPSPath
The path to the SPS document.

strXMLPath
The path to the XML document (can be empty).

Return Value
The method returns the new document.

Description
OpenAuthenticFile opens an XML file or database in Authentic View using SPS design strSPSPath.

29.3.2.11.7 OpenFile

Method: OpenFile (*strPath* as String, *bDialog* as Boolean) as Document[1668]

Parameters

strPath
Path and file name of file to open.

bDialog
Show dialogs for user input.

Return Value
Returns the opened file on success.

Description
OpenFile opens the file strPath. If bDialog is TRUE, a file-dialog will be displayed.

Example

```
Dim objDoc As Document
Set objDoc = objSpy.Documents.OpenFile(strFile, False)
```

29.3.2.11.8 OpenURL

Method: OpenURL (*strURL* as String, *nURLType* as SPYURLTypes[1855], *nLoading* as SPYLoading[1849], *strUser* as String, *strPassword* as String) as Document[1668]

Parameters
strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

nLoading
Set nLoading to 0 (zero) if you want to load it from cache or proxy. Otherwise set nLoading to 1.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentification. Can be empty.

Return Value
The method returns the opened document.

Description
OpenURL opens the URL strURL.

29.3.2.11.9 OpenURLDialog

Method: OpenURLDialog (*strURL* as String, *nURLType* as SPYURLTypes[1855], *nLoading* as SPYLoading[1849], *strUser* as String, *strPassword* as String) as Document[1668]

Parameters
strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

nLoading
Set nLoading to 0 (zero) if you want to load it from cache or proxy. Otherwise set nLoading to 1.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentification. Can be empty.

Return Value
The method returns the opened document.

Description
OpenURLDialog displays the "open URL" dialog to the user and presets the input fields with the given parameters.

29.3.2.12 DTDSchemaGeneratorDlg

Properties and Methods

Standard automation properties
Application 1702
Parent 1705

DTDSchemaFormat 1703
ValueList 1706
TypeDetection 1706
FrequentElements 1703
MergeAllEqualNamed 1704
ResolveEntities 1705
AttributeTypeDefinition 1703
GlobalAttributes 1704
OnlyStringEnums 1704
MaxEnumLength 1704
OutputPath 1705
OutputPathDialogAction 1705

Description
Use this object to configure the generation of a schema or DTD. The method GenerateDTDOrSchemaEx 1683 expects a DTDSchemaGeneratorDlg as parameter to configure the generation as well as the associated user interactions.

29.3.2.12.1 Application

Property: Application as Application 1580 (read-only)

Description
Access the XMLSpy application object.

(C) 2015-2021 Altova GmbH

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.12.2 AttributeTypeDefinition

Property: AttributeTypeDefinition as SPYAttributeTypeDefinition 1844

Description
Specifies how attribute definitions get merged.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.12.3 DTDSchemaFormat

Property: DTDSchemaFormat as SPYDTDSchemaFormat 1847

Description
Sets the schema output format to DTD, or W3C.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.12.4 FrequentElements

Property: FrequentElements as SPYFrequentElements 1848

Description
Shall the types for all elements be defined as global? Use the value *spyGlobalComplexType* to define them on global scope. Otherwise, use the value *spyGlobalElements*.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.12.5 GlobalAttributes

Property: GlobalAttributes as Boolean

Description
Shall attributes with same name and type be resolved globally?

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.12.6 MaxEnumLength

Property: MaxEnumLength as Integer

Description
Specifies the maximum number of characters allowed for enumeration names. If one value is longer than this, no enumeration will be generated.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.12.7 MergeAllEqualNamed

Property: MergeAllEqualNamed as Boolean

Description
Shall types of all elements with the same name be merged into one type?

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.12.8 OnlyStringEnums

Property: OnlyStringEnums as Boolean

Description
Specifies if enumerations will be created only for plain strings or all types of values.

Errors
3000 The object is no longer valid.

(C) 2015-2021 Altova GmbH

3001 Invalid address for the return parameter was specified.

29.3.2.12.9 OutputPath

Property: OutputPath as String

Description
Selects the file name for the generated schema/DTD.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.12.10 OutputPathDialogAction

Property: OutputPathDialogAction as SPYDialogAction[1846]

Description
Defines how the sub-dialog for selecting the schema/DTD output path gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current value of the OutputPath[1705] property as default. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.12.11 Parent

Property: Parent as Dialogs[1664] (read-only)

Description
Access the parent of the object.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.12.12 ResolveEntities

Property: ResolveEntities as Boolean

Description

Shall all entities be resolved before generation starts? If yes, an info-set will be built.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.12.13 TypeDetection

Property: TypeDetection as SPYTypeDetection[1855]

Description
Specifies granularity of simple type detection.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.12.14 ValueList

Property: ValueList as Integer

Description
Generate not more than this amount of enumeration-facets per type. Set to -1 for unlimited.

Errors
3000 The object is no longer valid.
3001 Invalid address for the return parameter was specified.

29.3.2.13 ElementList

Properties
Count[1707]
Item[1707]

Methods
RemoveElement[1707]

Description
Element lists are used for different purposes during export and import of data. Depending on this purpose, different properties of ElementListItem[1707] are used.

It can hold
- a list of table names returned by a call to Application.GetDatabaseTables[1587],

(C) 2015-2021 Altova GmbH

- a list of field names retuned by a call to Application.GetDatabaseImportElementList[1586] or Application.GetTextImportElementList[1588],
- a field name filter list used in Application.ImportFromDatabase[1590] and Application.ImportFromText[1592],
- a list of table names and counts for their rows and columns as returned by calls to GetExportElementList[1686] or
- a field name filter list used in Document.ExportToDatabase[1680] and Document.ExportToText[1681].

29.3.2.13.1 Count

Property: Count as long (read-only)

Description
Count of elements in this collection.

29.3.2.13.2 Item

Method: Item(n as long) as ElementListItem[1707]

Description
Gets the element with the index n from this collection. The first item has index 1.

29.3.2.13.3 RemoveElement

Method: RemoveElement(Index as long)

Description
RemoveElement removes the element Index from the collection. The first Item has index 1.

29.3.2.14 ElementListItem

Properties
Name[1708]

ElementKind[1707]

FieldCount[1708]
RecordCount[1708]

Description
An element in an ElementList[1706]. Usage of its properties depends on the purpose of the element list. For details see ElementList[1706].

29.3.2.14.1 ElementKind

Property: ElementKind as SPYXMLDataKind[1857]

Description

Specifies if a field should be imported as XML element (data value of spyXMLDataElement) or attribute (data value of spyXMLDataAttr).

29.3.2.14.2 FieldCount

Property: FieldCount as long (read-only)

Description
Count of fields (i.e. columns) in the table described by this element. This property is only valid after a call to Document.GetExportElementList[1686].

29.3.2.14.3 Name

Property: Name as String (read-only)

Description
Name of the element. This is either the name of a table or a field, depending on the purpose of the element list.

29.3.2.14.4 RecordCount

Property: RecordCount as long (read-only)

Description
Count of records (i.e. rows) in the table described by this element. This property is only valid after a call to Document.GetExportElementList[1686].

29.3.2.15 ExportSettings

Properties

ElementList[1709]

EntitiesToText[1709]

ExportAllElements[1709]
SubLevelLimit[1711]

FromAttributes[1710]
FromSingleSubElements[1710]
FromTextValues[1710]

CreateKeys[1709]
IndependentPrimaryKey[1710]

Namespace[1710]

ExportCompleteXML[1709]
StartFromElement[1710]

(C) 2015-2021 Altova GmbH

Description
ExportSettings contains options used during export of XML data to a database or text file. See Import and export of data[1552] for a general overview.

29.3.2.15.1 CreateKeys

Property: CreateKeys as Boolean

Description
This property turns creation of keys (i.e. primary key and foreign key) on or off. Default is True.

29.3.2.15.2 ElementList

Property: ElementList as ElementList[1706]

Description
Default is empty list. This list of elements defines which fields will be exported. To get the list of available fields use Document.GetExportElementList[1686]. It is possible to prevent exporting columns by removing elements from this list with ElementList.RemoveElement[1707] before passing it to Document.ExportToDatabase[1680] or Document.ExportToText[1681].

29.3.2.15.3 EntitiesToText

Property: EntitiesToText as Boolean

Description
Defines if XML entities should be converted to text or left as they are during export. Default is True.

29.3.2.15.4 ExportAllElements

Property: ExportAllElements as Boolean

Description
If set to true, all elements in the document will be exported. If set to false, then ExportSettings.SubLevelLimit[1711] is used to restrict the number of sub levels to export. Default is true.

29.3.2.15.5 ExportCompleteXML

Property: ExportCompleteXML as Boolean

Description
Defines whether the complete XML is exported or only the element specified by StartFromElement[1710] and its children. Default is True.

29.3.2.15.6 FromAttributes

Property: FromAttributes as Boolean

Description
Set FromAttributes to false if no export data should be created from attributes. Default is True.

29.3.2.15.7 FromSingleSubElements

Property: FromSingleSubElements as Boolean

Description
Set FromSingleSubElements to false if no export data should be created from elements. Default is True.

29.3.2.15.8 FromTextValues

Property: FromTextValues as Boolean

Description
Set FromTextValues to false if no export data should be created from text values. Default is True.

29.3.2.15.9 IndependentPrimaryKey

Property: IndependentPrimaryKey as Boolean

Description
Turns creation of independent primary key counter for every element on or off. If ExportSettings.CreateKeys[1709] is False, this property will be ignored. Default is True.

29.3.2.15.10 Namespace

Property: Namespace as SPYExportNamespace[1847]

Description
The default setting removes all namespace prefixes from the element names. In some database formats the colon is not a legal character. Default is spyNoNamespace.

29.3.2.15.11 StartFromElement

Property: StartFromElement as String

Description
Specifies the start element for the export. This property is only considered when ExportCompleteXML[1709] is false.

(C) 2015-2021 Altova GmbH

29.3.2.15.12 SubLevelLimit

Property: SubLevelLimit as Integer

Description
Defines the number of sub levels to include for the export. Default is 0. This property is ignored if ExportSettings.ExportAllElements[1709] is true.

29.3.2.16 FileSelectionDlg

Properties and Methods

Standard automation properties
Application[1711]
Parent[1712]

Dialog properties
FullName[1712]

Acceptance or cancellation of action that caused event
DialogAction[1711]

Description
The dialog object allows you to receive information about an event and pass back information to the event handler in the same way as with a user dialog. Use the FileSelectionDlg.FullName[1712] to select or modify the file path and set the FileSelectionDlg.DialogAction[1711] property to cancel or agree with the action that caused the event.

29.3.2.16.1 Application

Property: Application as Application[1580] (read-only)

Description
Access the XMLSpy application object.

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

29.3.2.16.2 DialogAction

Property: DialogAction as SPYDialogAction[1846]

Description
If you want your script to perform the file selection operation without any user interaction necessary, simulate user interaction by either setting the property to *spyDialogOK(0)* or *spyDialogCancel(1)*.

To allow your script to fill in the default values but let the user see and react on the dialog, use the value *spyDialogUserInput(2)*. If you receive a FileSelectionDlg object in an event handler, *spyDialogUserInput(2)* is not supported and will be interpreted as *spyDialogOK(0)*.

Errors

2400	The object is no longer valid.
2401	Invalid value for dialog action or invalid address for the return parameter was specified.

29.3.2.16.3 FullName

Property: FullName as String

Description
Access the full path of the file the gets selected by the dialog. Most events that pass a FileSelectionDlg object to you allow you modify this value and thus influence the action that caused the event (e.g. load or save to a different location).

Errors

2400	The object is no longer valid.
2401	Invalid address for the return parameter was specified.

29.3.2.16.4 Parent

Property: Parent as Dialogs 1664 (read-only)

Description
Access the parent of the object.

Errors

2400	The object is no longer valid.
2401	Invalid address for the return parameter was specified.

29.3.2.17 FindInFilesDlg

Properties and Methods

Standard automation properties
Application 1713
Parent 1715

Find 1714
RegularExpression 1715
Replace 1716
DoReplace 1714
ReplaceOnDisk 1716
MatchWholeWord 1715

(C) 2015-2021 Altova GmbH

MatchCase 1715
SearchLocation 1716
StartFolder 1717
IncludeSubfolders 1714
SearchInProjectFilesDoExternal 1716
FileExtension 1714
AdvancedXMLSearch 1713
XMLElementNames 1718
XMLElementContents 1718
XMLAttributeNames 1717
XMLAttributeContents 1717
XMLComments 1718
XMLCData 1718
XMLPI 1719
XMLRest 1719
ShowResult 1717

Description
Use this object to configure the search (or replacement) for strings in files. The method FindInFiles 1586 expects a FindInFilesDlg as parameter.

29.3.2.17.1 AdvancedXMLSearch

Property: AdvancedXMLSearch as Boolean

Description
Specifies if the XML search properties (XMLElementNames 1718, XMLElementContents 1718, XMLAttributeNames 1717, XMLAttributeContents 1718, XMLComments 1718, XMLCData 1718, XMLPI 1719 and XMLRest 1719) are considered. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.2 Application

Property: Application as Application 1580 (read-only)

Description
Access the XMLSpy application object.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.3 DoReplace

Property: DoReplace as Boolean

Description
Specifies if the matched string is replaced by the string defined in Replace[1716]. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.4 FileExtension

Property: FileExtension as String

Description
Specifies the file filter of the files that should be considered during the search. Multiple file filters must be delimited with a semicolon (eg: *.xml;*.dtd;a*.xsd). Use the wildcards * and ? to define the file filter.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.5 Find

Property: Find as String

Description
Specifies the string to search for.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.6 IncludeSubfolders

Property: IncludeSubfolders as Boolean

Description
Specifies if subfolders are searched too. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.17.7 MatchCase

Property: MatchCase as Boolean

Description
Specifies if the search is case sensitive. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.8 MatchWholeWord

Property: MatchWholeWord as Boolean

Description
Specifies whether the whole word or just a part of it must match. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.9 Parent

Property: Parent as Dialogs 1664 (read-only)

Description
Access the parent of the object.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.10 RegularExpression

Property: RegularExpression as Boolean

Description
Specifies if Find 1714 contains a regular expression. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.11 Replace

Property: Replace as String

Description
Specifies the replacement string. The matched string is only replaced if DoReplace[1714] is set true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.12 ReplaceOnDisk

Property: ReplaceOnDisk as Boolean

Description
Specifies if the replacement is done directly on disk. The modified file is not opened. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.13 SearchInProjectFilesDoExternal

Property: SearchInProjectFilesDoExternal as Boolean

Description
Specifies if the external folders in the open project are searched, when a project search is performed. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.14 SearchLocation

Property: SearchLocation as SPYFindInFilesSearchLocation[1848]

Description
Specifies the location of the search. The default is spyFindInFiles_Documents.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.17.15 ShowResult

Property: ShowResult as Boolean

Description
Specifies if the result is displayed in the Find in Files output window. The default is false.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.16 StartFolder

Property: StartFolder as String

Description
Specifies the folder where the disk search starts.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.17 XMLAttributeContents

Property: XMLAttributeContents as Boolean

Description
Specifies if attribute contents are searched when AdvancedXMLSearch[1713] is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.18 XMLAttributeNames

Property: XMLAttributeNames as Boolean

Description
Specifies if attribute names are searched when AdvancedXMLSearch[1713] is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.19 XMLCData

Property: XMLCData as Boolean

Description
Specifies if CData tags are searched when AdvancedXMLSearch[1713] is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.20 XMLComments

Property: XMLComments as Boolean

Description
Specifies if comments are searched when AdvancedXMLSearch[1713] is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.21 XMLElementContents

Property: XMLElementContents as Boolean

Description
Specifies if element contents are searched when AdvancedXMLSearch[1713] is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.22 XMLElementNames

Property: XMLElementNames as Boolean

Description
Specifies if element names are searched when AdvancedXMLSearch[1713] is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.17.23 XMLPI

Property: XMLPI as Boolean

Description
Specifies if XML processing instructions are searched when AdvancedXMLSearch[1713] is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.17.24 XMLRest

Property: XMLRest as Boolean

Description
Specifies if the rest of the XML (which is not covered by the other XML search properties) is searched when AdvancedXMLSearch[1713] is true. The default is true.

Errors
3500 The object is no longer valid.
3501 Invalid address for the return parameter was specified.

29.3.2.18 FindInFilesResult

Properties and Methods

Standard automation properties
Application[1719]
Parent[1720]

Count[1720]
Item[1720]

Path[1721]
Document[1720]

Description
This object represents a file that matched the search criteria. It contains a list of FindInFilesResultMatch[1721] objects that describe the matching position.

29.3.2.18.1 Application

Property: Application as Application[1580] (read-only)

Description

Access the XMLSpy application object.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

29.3.2.18.2 Count

Property: Count as long (read-only)

Description
Count of elements in this collection.

29.3.2.18.3 Document

Property: Path as Document[1668] (read-only)

Description
This property returns the Document[1668] object if the matched file is already open in XMLSpy.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

29.3.2.18.4 Item

Method: Item(n as long) as FindInFilesResultMatch[1721]

Description
Gets the element with the index n from this collection. The first item has index 1.

29.3.2.18.5 Parent

Property: Parent as FindInFilesResults[1723] (read-only)

Description
Access the parent of the object.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.18.6 Path

Property: Path as String (read-only)

Description
Returns the path of the file that matched the search criteria.

Errors
3700 The object is no longer valid.
3701 Invalid address for the return parameter was specified.

29.3.2.19 FindInFilesResultMatch

Properties and Methods

Standard automation properties
Application 1721
Parent 1722

Line 1722
Position 1723
Length 1721
LineText 1722
Replaced 1723

Description
Contains the exact position in the file of the matched string.

29.3.2.19.1 Application

Property: Application as Application 1580 (read-only)

Description
Access the XMLSpy application object.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

29.3.2.19.2 Length

Property: Length as Long (read-only)

Description
Returns the length of the matched string.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

29.3.2.19.3 Line

Property: Line as Long (read-only)

Description
Returns the line number of the match. The line numbering starts with 0.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

29.3.2.19.4 LineText

Property: LineText as String (read-only)

Description
Returns the text of the line.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

29.3.2.19.5 Parent

Property: Parent as FindInFilesResult[1719] (read-only)

Description
Access the parent of the object.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.19.6 Position

Property: Position as Long (read-only)

Description
Returns the start position of the match in the line. The position numbering starts with 0.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

29.3.2.19.7 Replaced

Property: Replaced as Boolean (read-only)

Description
True if the matched string was replaced.

Errors
3800 The object is no longer valid.
3801 Invalid address for the return parameter was specified.

29.3.2.20 FindInFilesResults

Properties and Methods

Standard automation properties
Application 1723
Parent 1724

Count 1724
Item 1724

Description
This is the result of the FindInFiles 1586 method. It is a list of FindInFilesResult 1719 objects.

29.3.2.20.1 Application

Property: Application as Application 1580 (read-only)

Description
Access the XMLSpy application object.

Errors
3600 The object is no longer valid.

3601 Invalid address for the return parameter was specified.

29.3.2.20.2 Count

Property: Count as long (read-only)

Description
Count of elements in this collection.

29.3.2.20.3 Item

Method: Item(n as long) as FindInFilesResult[1719]

Description
Gets the element with the index n from this collection. The first item has index 1.

29.3.2.20.4 Parent

Property: Parent as Application[1580] (read-only)

Description
Access the parent of the object.

Errors
3600 The object is no longer valid.
3601 Invalid address for the return parameter was specified.

29.3.2.21 GenerateSampleXMLDlg

Properties and Methods

Standard automation properties
Application[1725]
Parent[1728]

NonMandatoryAttributes[1727]
NonMandatoryElements[1727]
RepeatCount[1728]
FillAttributesWithSampleData[1726]
FillElementsWithSampleData[1726]
ContentOfNillableElementsIsNonMandatory[1726]
TryToUseNonAbstractTypes[1729]
SchemaOrDTDAssignment[1729]
LocalNameOfRootElement[1727]
NamespaceURIOfRootElement[1727]

OptionsDialogAction 1728

Properties that are no longer supported
TakeFirstChoice - obsolete 1729
FillWithSampleData - obsolete 1726
Optimization - obsolete 1728

Description
Used to set the parameters for the generation of sample XML instances based on a W3C schema or DTD.

29.3.2.21.1 Application

Property: Application as Application 1580 (read-only)

Description
Access the XMLSpy application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.2 ChoiceMode

Property: `ChoiceMode` as SPYSampleXMLGenerationChoiceMode 1852

Description
Specifies which elements will be generated.

Errors

2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.3 ConsiderSampleValueHints

Property: `ConsiderSampleValueHints` as Boolean

Description
Selects whether to use SampleValueHints 1729 or not.

Errors

2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.4 ContentOfNillableElementsIsNonMandatory

Property: ContentOfNillableElementsIsNonMandatory as Boolean

Description
If true, the contents of elements that are nillable will not be treated as mandatory.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.5 FillAttributesWithSampleData

Property: FillAttributesWithSampleData as Boolean

Description
If true, attributes will have sample content.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.6 FillElementsWithSampleData

Property: FillElementsWithSampleData as Boolean

Description
If true, elements will have sample content.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.7 FillWithSampleData – obsolete

Property: FillWithSampleData as Boolean

Description
Do no longer access this property. Use FillAttributesWithSampleData[1726] and FillElementsWithSampleData[1726], instead.

(C) 2015-2021 Altova GmbH

Errors
0001 The property is no longer accessible.

29.3.2.21.8 LocalNameOfRootElement

Property: LocalNameOfRootElement as String

Description
Specifies the local name of the root element for the generated sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.9 NamespaceURIOfRootElement

Property: NamespaceURIOfRootElement as String

Description
Specifies the namespace URI of the root element for the generated sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.10 NonMandatoryAttributes

Property: NonMandatoryAttributes as Boolean

Description
If true attributes which are not mandatory are created in the sample XML instance file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.11 NonMandatoryElements

Property: NonMandatoryElements as Boolean

Description
If `true`, elements which are not mandatory are created in the sample XML instance file.

Errors

2200 The object is no longer valid.
2201 Invalid address was specified for the return parameter.

29.3.2.21.12 Optimization – obsolete

Property: Optimization as SPYSampleXMLGenerationOptimization 1852

Description
Do not use this property any longer. Use ChoiceMode and NonMandatoryElements.

Errors
0001 The property is no longer accessible.

29.3.2.21.13 OptionsDialogAction

Property: OptionsDialogAction as SPYDialogAction 1846

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors
2200 The object is no longer valid.
2201 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

29.3.2.21.14 Parent

Property: Parent as Dialogs 1664 (read-only)

Description
Access the parent of the object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.15 RepeatCount

Property: RepeatCount as long

Description
Number of elements to create for repeated types.

(C) 2015-2021 Altova GmbH

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.16 SampleValueHints

Property: `SampleValueHints` as `SPYSampleXMLGenerationSampleValueHints` 1852

Description
Specifies how to select data for the generated sample file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.17 SchemaOrDTDAssignment

Property: SchemaOrDTDAssignment as SPYSampleXMLGenerationSchemaOrDTDAssignment 1852

Description
Specifies in which way a reference to the related schema or DTD - which is this document - will be generated into the sample XML.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.21.18 TakeFirstChoice – obsolete

Property: TakeFirstChoice as Boolean

Description
Do no longer use this property.

Errors
0001 The property is no longer accessible.

29.3.2.21.19 TryToUseNonAbstractTypes

Property: TryToUseNonAbstractTypes as Boolean

Description
If true, tries to use a non-abstract type for xsi:type, if element has an abstract type.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

29.3.2.22 GridView

Methods
Deselect 1733
Select 1733

SetFocus 1733

Properties
CurrentFocus 1732

IsVisible 1733

Description
GridView Class

29.3.2.22.1 Events

29.3.2.22.1.1 OnBeforeDrag

Event: OnBeforeDrag() as Boolean

XMLSpy scripting environment - VBScript:
```
Function On_BeforeDrag()
	' On_BeforeStartEditing = False  ' to prohibit dragging
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeDrag()
{
	// return false;  /* to prohibit dragging */
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (4, ...)  // nEventId = 4
```

Description
This event gets fired on an attempt to drag an XMLData element on the grid view. Return *false* to prevent dragging the data element to a different position.

(C) 2015-2021 Altova GmbH

29.3.2.22.1.2 OnBeforeDrop

Event: OnBeforeDrop(*objXMLData* as XMLData[1795]) as Boolean

XMLSpy scripting environment - VBScript:
```
Function On_BeforeDrop(objXMLData)
	' On_BeforeStartEditing = False  ' to prohibit dropping
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeDrop(objXMLData)
{
	// return false;  /* to prohibit dropping */
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (5, ...)  // nEventId = 5
```

Description
This event gets fired on an attempt to drop a previously dragged XMLData element on the grid view. Return *false* to prevent the data element to be moved from its original position to the drop destination position.

29.3.2.22.1.3 OnBeforeStartEditing

Event: OnBeforeStartEditing(*objXMLData* as XMLData[1795], *bEditingName* as Boolean)as Boolean

XMLSpy scripting environment - VBScript:
```
Function On_BeforeStartEditing(objXMLData, bEditingName)
	' On_BeforeStartEditing = False  ' to prohibit editing the field
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeStartEditing(objXMLData, bEditingName)
{
	// return false;  /* to prohibit editing the field */
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (1, ...)  // nEventId = 1
```

Description
This event gets fired before the editing mode for a grid cell gets entered. If the parameter *bEditingName* is true, the name part of the element will be edited, it its value is false, the value part will be edited.

29.3.2.22.1.4 OnEditingFinished

Event: OnEditingFinished(*objXMLData* as XMLData[1795], *bEditingName* as Boolean)

XMLSpy scripting environment - VBScript:
Function On_EditingFinished(*objXMLData*, *bEditingName*)
End Function

XMLSpy scripting environment - JScript:
function On_EditingFinished(*objXMLData*, *bEditingName*)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (2, ...) // nEventId = 2

Description
This event gets fired when the editing mode of a grid cell is exited. The parameter *bEditingName* specifies if the name part of the element has been edited.

29.3.2.22.1.5 OnFocusChanged

Event: OnFocusChanged(*objXMLData* as XMLData[1795], *bSetFocus* as Boolean, *bEditingName* as Boolean)

XMLSpy scripting environment - VBScript:
Function On_FocusChanged(*objXMLData*, *bSetFocus*, *bEditingName*)
End Function

XMLSpy scripting environment - JScript:
function On_FocusChanged(*objXMLData*, *bSetFocus*, *bEditingName*)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (3, ...) // nEventId = 3

Description
This event gets fired whenever a grid cell receives or loses the cursor focus. If the parameter *bEditingName* is *true*, focus of the name part of the grid element has changed. Otherwise, focus of the value part has changed.

29.3.2.22.2 CurrentFocus

Property: CurrentFocus as XMLData[1795]

Description
Holds the XML element with the current focus. This property is read-only.

(C) 2015-2021 Altova GmbH

29.3.2.22.3 Deselect

Method: Deselect(*pData* as XMLData[1795])

Description
Deselects the element pData in the grid view.

29.3.2.22.4 IsVisible

Property: IsVisible as Boolean

Description
True if the grid view is the active view of the document. This property is read-only.

29.3.2.22.5 Select

Method: Select (*pData* as XMLData[1795])

Description
Selects the XML element pData in the grid view.

29.3.2.22.6 SetFocus

Method: SetFocus (*pFocusData* as XMLData[1795])

Description
Sets the focus to the element pFocusData in the grid view.

29.3.2.23 SchemaDocumentationDlg

Properties and Methods

Standard automation properties
Application[1735]
Parent[1742]

Interaction and visibility properties
OutputFile[1741]
OutputFileDialogAction[1741]
OptionsDialogAction[1740]
ShowProgressBar[1744]
ShowResult[1745]

Document generation options and methods
OutputFormat[1741]
UseFixedDesign[1747]
SPSFile[1746]

EmbedDiagrams 1736
DiagramFormat 1735
MultipleOutputFiles 1740
EmbedCSSInHTML 1736
CreateDiagramsFolder 1735
GenerateRelativeLinks 1736

IncludeAll 1737
IncludeIndex 1738
IncludeGlobalAttributes 1737
IncludeGlobalElements 1738
IncludeLocalAttributes 1739
IncludeLocalElements 1739
IncludeGroups 1738
IncludeComplexTypes 1737
IncludeSimpleTypes 1740
IncludeAttributeGroups 1737
IncludeRedefines 1739
IncludeReferencedSchemas 1739

AllDetails 1734
ShowDiagram 1743
ShowNamespace 1744
ShowType 1746
ShowChildren 1742
ShowUsedBy 1746
ShowProperties 1745
ShowSingleFacets 1745
ShowPatterns 1744
ShowEnumerations 1743
ShowAttributes 1742
ShowIdentityConstraints 1743
ShowAnnotations 1742
ShowSourceCode 1745

Description
This object combines all options for schema document generation as they are available through user interface dialog boxes in XMLSpy. The document generation options are initialized with the values used during the last generation of schema documentation. However, before using the object you have to set the SetOutputFile 1741 property to a valid file path. Use OptionsDialogAction 1740, OutputFileDialogAction 1741 and ShowProgressBar 1744 to specify the level of user interaction desired. You can use IncludeAll 1737 and AllDetails 1734 to set whole option groups at once or the individual properties to operate on a finer granularity.

29.3.2.23.1 AllDetails

Method: AllDetails (i_bDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

(C) 2015-2021 Altova GmbH

Errors
2900 The object is no longer valid.

29.3.2.23.2 Application

Property: Application as Application[1580] (read-only)

Description
Access the XMLSpy application object.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.3 CreateDiagramsFolder

Property: CreateDiagramsFolder as Boolean

Description
Set this property to true, to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is false.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.4 DiagramFormat

Property: DiagramFormat as SPYImageKind[1848]

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is PNG.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.5 EmbedCSSInHTML

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.6 EmbedDiagrams

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.7 GenerateRelativeLinks

Property: GenerateRelativeLinks as Boolean

Description
Set this property to true, to create relative paths to local files. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is false.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.23.8 IncludeAll

Method: IncludeAll (i_bInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
2900 The object is no longer valid.

29.3.2.23.9 IncludeAttributeGroups

Property: IncludeAttributeGroups as Boolean

Description
Set this property to true, to include attribute groups in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.10 IncludeComplexTypes

Property: IncludeComplexTypes as Boolean

Description
Set this property to true, to include complex types in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.11 IncludeGlobalAttributes

Property: IncludeGlobalAttributes as Boolean

Description
Set this property to true, to include global attributes in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.12 IncludeGlobalElements

Property: IncludeGlobalElements as Boolean

Description
Set this property to true, to include global elements in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.13 IncludeGroups

Property: IncludeGroups as Boolean

Description
Set this property to true, to include groups in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.14 IncludeIndex

Property: IncludeIndex as Boolean

Description
Set this property to true, to include an index in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.23.15 IncludeLocalAttributes

Property: IncludeLocalAttributes as Boolean

Description
Set this property to true, to include local attributes in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.16 IncludeLocalElements

Property: IncludeLocalElements as Boolean

Description
Set this property to true, to include local elements in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.17 IncludeRedefines

Property: IncludeRedefines as Boolean

Description
Set this property to true, to include redefines in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.18 IncludeReferencedSchemas

Property: IncludeReferencedSchemas as Boolean

Description

Set this property to true, to include referenced schemas in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.19 IncludeSimpleTypes

Property: IncludeSimpleTypes as Boolean

Description
Set this property to true, to include simple types in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.20 MultipleOutputFiles

Property: MultipleOutputFiles as Boolean

Description
Set this property to true, to split the documentation files. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is false.

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

29.3.2.23.21 OptionsDialogAction

Property: OptionsDialogAction as SPYDialogAction[1846]

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.

(C) 2015-2021 Altova GmbH

Invalid address for the return parameter was specified.

29.3.2.23.22 OutputFile

Property: OutputFile as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to Document.GenerateSchemaDocumentation[1684].

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.23 OutputFileDialogAction

Property: OutputFileDialogAction as SPYDialogAction[1846]

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile[1741] should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

29.3.2.23.24 OutputFormat

Property: OutputFormat as SPYSchemaDocumentationFormat[1853]

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is HTML.

Errors
2900 The object is no longer valid.
2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

29.3.2.23.25 Parent

Property: Parent as Dialogs[1664] (read-only)

Description
Access the parent of the object.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.26 ShowAnnotations

Property: ShowAnnotations as Boolean

Description
Set this property to true, to show the annotations to a type definition in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.27 ShowAttributes

Property: ShowAttributes as Boolean

Description
Set this property to true, to show the type definitions attributes in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.28 ShowChildren

Property: ShowChildren as Boolean

Description

Set this property to true, to show the children of a type definition as links in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.29 ShowDiagram

Property: ShowDiagram as Boolean

Description
Set this property to true, to show type definitions as diagrams in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.30 ShowEnumerations

Property: ShowEnumerations as Boolean

Description
Set this property to true, to show the enumerations contained in a type definition in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.31 ShowIdentityConstraints

Property: ShowIdentityConstraints as Boolean

Description
Set this property to true, to show a type definitions identity constraints in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

29.3.2.23.32 ShowNamespace

Property: ShowNamespace as Boolean

Description
Set this property to true, to show the namespace of type definitions in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.33 ShowPatterns

Property: ShowPatterns as Boolean

Description
Set this property to true, to show the patterns of a type definition in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.34 ShowProgressBar

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.23.35 ShowProperties

Property: ShowProperties as Boolean

Description
Set this property to true, to show the type definition properties in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.36 ShowResult

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in XMLSpy. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.37 ShowSingleFacets

Property: ShowSingleFacets as Boolean

Description
Set this property to true, to show the facets of a type definition in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.38 ShowSourceCode

Property: ShowSourceCode as Boolean

Description
Set this property to true, to show the XML source code for type definitions in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.39 ShowType

Property: ShowType as Boolean

Description
Set this property to true, to show the type of type definitions in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.40 ShowUsedBy

Property: ShowUsedBy as Boolean

Description
Set this property to true, to show the used-by relation for type definitions in the schema documentation. The property is initialized with the value used during the last call to Document.GenerateSchemaDocumentation[1684]. The default for the first run is true.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.23.41 SPSFile

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
2900 The object is no longer valid.

(C) 2015-2021 Altova GmbH

2901 Invalid address for the return parameter was specified.

29.3.2.23.42 UseFixedDesign

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

29.3.2.24 SpyProject

Methods
CloseProject 1747
SaveProject 1748
SaveProjectAs 1748

Properties
RootItems 1748
ProjectFile 1748

Description
SpyProject Class

29.3.2.24.1 CloseProject

Declaration: CloseProject(*bDiscardChanges* as Boolean, *bCloseFiles* as Boolean, *bDialog* as Boolean)

Parameters
bDiscardChanges
Set bDiscardChanges to FALSE if you want to save the changes of the open project files and the project.

bCloseFiles
Set bCloseFiles to TRUE to close all open project files.

bDialog
Show dialogs for user input.

Description
CloseProject closes the current project.

29.3.2.24.2 ProjectFile

Declaration: ProjectFile as String

Description
Path and filename of the project.

29.3.2.24.3 RootItems

Declaration: RootItems as SpyProjectItems[1751]

Description
Root level of collection of project items.

29.3.2.24.4 SaveProject

Declaration: SaveProject

Description
SaveProject saves the current project.

29.3.2.24.5 SaveProjectAs

Declaration: SaveProjectAs (strPath as String, bDialog as Boolean)

Parameters
strPath
Full path with file name of new project file.

bDialog
If bDialog is TRUE, a file-dialog will be displayed.

Description
SaveProjectAs stores the project data into a new location.

29.3.2.25 SpyProjectItem

Methods
Open[1749]

Properties
ChildItems[1749]
ParentItem[1750]
FileExtensions[1749]
ItemType[1749]
Name[1749]
Path[1750]

(C) 2015-2021 Altova GmbH

ValidateWith 1750
XMLForXSLTransformation 1750
XSLForXMLTransformation 1750
XSLTransformationFileExtension 1750
XSLTransformationFolder 1751

Description
SpyProjectItem Class

29.3.2.25.1 ChildItems

Declaration: ChildItems as SpyProjectItems 1751

Description
If the item is a folder, ChildItems is the collection of the folder content.

29.3.2.25.2 FileExtensions

Declaration: FileExtensions as String

Description
Used to set the file extensions if the project item is a folder.

29.3.2.25.3 ItemType

Declaration: ItemType as SPYProjectItemTypes 1851

Description
This property is read-only.

29.3.2.25.4 Name

Declaration: Name as String

Description
Name of the project item. This property is read-only.

29.3.2.25.5 Open

Declaration: Open as Document 1668

Return Value
The project item opened as document.

Description
Opens the project item.

29.3.2.25.6 ParentItem

Declaration: ParentItem as SpyProjectItem[1748]

Description
Parent item of the current project item. Can be NULL (Nothing) if the project item is a top-level item.

29.3.2.25.7 Path

Declaration: Path as String

Description
Path of project item. This property is read-only.

29.3.2.25.8 ValidateWith

Declaration: ValidateWith as String

Description
Used to set the schema/DTD for validation.

29.3.2.25.9 XMLForXSLTransformation

Declaration: XMLForXSLTransformation as String

Description
Used to set the XML for XSL transformation.

29.3.2.25.10 XSLForXMLTransformation

Declaration: XSLForXMLTransformation as String

Description
Used to set the XSL for XML transformation.

29.3.2.25.11 XSLTransformationFileExtension

Declaration: XSLTransformationFileExtension as String

Description
Used to set the file extension for XSL transformation output files.

(C) 2015-2021 Altova GmbH

29.3.2.25.12 XSLTransformationFolder

Declaration: XSLTransformationFolder as String

Description
Used to set the destination folder for XSL transformation output files.

29.3.2.26 SpyProjectItems

Methods
AddFile 1751
AddFolder 1751
AddURL 1752
RemoveItem 1752

Properties
Count 1752
Item 1752

Description
SpyProjectItems Class

29.3.2.26.1 AddFile

Declaration: AddFile (*strPath* as String)

Parameters

strPath
Full path with file name of new project item

Description
The method adds a new file to the collection of project items.

29.3.2.26.2 AddFolder

Declaration: AddFolder (*strName* as String)

Parameters

strName
Name of the new folder.

Description
The method AddFolder adds a folder with the name strName to the collection of project items.

29.3.2.26.3 AddURL

Declaration: AddURL (*strURL* as String, *nURLType* as SPYURLTypes[1855], *strUser* as String, *strPassword* as String, *bSave* as Boolean)

Description

strURL
URL to open as document.

nURLType
Type of document to open. Set to -1 for auto detection.

strUser
Name of the user if required. Can be empty.

strPassword
Password for authentification. Can be empty.

bSave
Save user and password information.

Description
The method adds an URL item to the project collection.

29.3.2.26.4 Count

Declaration: Count as long

Description
This property gets the count of project items in the collection. The property is read-only.

29.3.2.26.5 Item

Declaration: Item (*n* as long) as SpyProjectItem[1748]

Description
Retrieves the n-th element of the collection of project items. The first item has index 1.

29.3.2.26.6 RemoveItem

Declaration: RemoveItem (*pItem* as SpyProjectItem[1748])

Description
RemoveItem deletes the item pItem from the collection of project items.

(C) 2015-2021 Altova GmbH

29.3.2.27 TextImportExportSettings

Properties for import only
ImportFile 1754

Properties for export only
DestinationFolder 1753
FileExtension 1754
CommentIncluded 1753
RemoveDelimiter 1755
RemoveNewline 1755

Properties for import and export
HeaderRow 1754
FieldDelimiter 1754
EnclosingCharacter 1753
Encoding 1754
EncodingByteOrder 1754

Description
TextImportExportSettings contains options common to text import and export functions.

29.3.2.27.1 CommentIncluded

Property: CommentIncluded as Boolean

Description
This property tells whether additional comments are added to the generated text file. Default is true. This property is used only when exporting to text files.

29.3.2.27.2 DestinationFolder

Property: DestinationFolder as String

Description
The property DestinationFolder sets the folder where the created files are saved during text export.

29.3.2.27.3 EnclosingCharacter

Property: EnclosingCharacter as SPYTextEnclosing 1854

Description
This property defines the character that encloses all field values for import and export. Default is spyNoEnclosing 1854.

29.3.2.27.4 Encoding

Property: Encoding as String

Description
The property Encoding sets the character encoding for the text files for importing and exporting.

29.3.2.27.5 EncodingByteOrder

Property: EncodingByteOrder as SPYEncodingByteOrder 1847

Description
The property EncodingByteOrder sets the byte order for Unicode characters. Default is spyNONE 1847.

29.3.2.27.6 FieldDelimiter

Property: FieldDelimiter as SPYTextDelimiters 1854

Description
The property FieldDelimiter defines the delimiter between the fields during import and export. Default is spyTabulator 1854.

29.3.2.27.7 FileExtension

Property: FileExtension as String

Description
This property sets the file extension for files created on text export.

29.3.2.27.8 HeaderRow

Property: HeaderRow as Boolean

Description
The property HeaderRow is used during import and export. Set HeaderRow true on import, if the first line of the text file contains the names of the columns. Set HeaderRow true on export, if the first line in the created text files should contain the name of the columns. Default value is true.

29.3.2.27.9 ImportFile

Property: ImportFile as String

Description
This property is used to set the text file for import. The string has to be a full qualified path. See also Import and Export 1552.

(C) 2015-2021 Altova GmbH

29.3.2.27.10 RemoveDelimiter

Property: RemoveDelimiter as Boolean

Description
The property RemoveDelimiter defines whether characters in the text that are equal to the delimiter character are removed. Default is false. This property is used only when exporting to text files.

29.3.2.27.11 RemoveNewline

Property: RemoveNewline as Boolean

Description
The property RemoveNewline defines whether newline characters in the text are removed. Default is false. This property is used only when exporting to text files.

29.3.2.28 TextView

Properties and Methods

Application 1757
Parent 1759

LineFromPosition 1758
PositionFromLine 1759
LineLength 1758
SelText 1760
GetRangeText 1757
ReplaceText 1759
MoveCaret 1759
GoToLineChar 1757
SelectText 1760
SelectionStart 1760
SelectionEnd 1760
Text 1761
LineCount 1758
Length 1758

Description

29.3.2.28.1 Events

29.3.2.28.1.1 OnBeforeShowSuggestions

Event: OnBeforeShowSuggestions() as Boolean

Description
This event gets fired before a suggestion window is shown. The Document[1668] property Suggestions[1695] contains a string array that is recommended to the user. It is possible to modify the displayed recommendations during this event. Before doing so you have to assign an empty array to the Suggestions[1695] property. The best location for this is the OnDocumentOpened[1582] event. To prevent the suggestion window to show up return false and true to continue its display.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_BeforeShowSuggestions()
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeShowSuggestions()
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (33, ...)// nEventId = 33
```

29.3.2.28.1.2 OnChar

Event: OnChar(nChar as Long, bExistSuggestion as Boolean) as Boolean

Description
This event gets fired on each key stroke. The parameter nChar is the key that was pressed and bExistSuggestions tells whether a XMLSpy generated suggestions window is displayed after this key. The Document[1668] property Suggestions[1695] contains a string array that is recommended to the user. It is possible to modify the displayed recommendations during this event. Before doing so you have to assign an empty array to the Suggestions[1695] property. The best location for this is the OnDocumentOpened[1582] event. To prevent the suggestion window to show up return false and true to continue its display.
It is also possible to create a new suggestions window when none is provided by XMLSpy. Set the Document[1668] property Suggestions[1695] to a string array with your recommendations and return true.
This event is fired before the OnBeforeShowSuggestions[1756] event. If you prevent to show the suggestion window by returning false then OnBeforeShowSuggestions[1756] is not fired.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:
```
Function On_Char(nChar, bExistSuggestions)
```


(C) 2015-2021 Altova GmbH

End Function

XMLSpy scripting environment - JScript:
function On_Char(nChar, bExistSuggestions)
{
}

XMLSpy IDE Plugin:
IXMLSpyPlugIn.OnEvent (35, ...)// nEventId = 35

29.3.2.28.2 Application

Property: Application as Application[1580] (read-only)

Description
Access the XMLSpy application object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.3 GetRangeText

Method: GetRangeText(nStart as Long, nEnd as Long) as String

Description
Returns the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.4 GoToLineChar

Method: GoToLineChar(nLine as Long, nChar as Long)

Description
Moves the caret to the specified line and character position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.5 Length

Property: Length as Long

Description
Returns the character count of the document.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.6 LineCount

Property: LineCount as Long

Description
Returns the number of lines in the document.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.7 LineFromPosition

Method: LineFromPosition(nCharPos as Long) as Long

Description
Returns the line number of the character position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.8 LineLength

Method: LineLength(nLine as Long) as Long

Description
Returns the length of the line.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.28.9 MoveCaret

Method: MoveCaret(nDiff as Long)

Description
Moves the caret nDiff characters.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.10 Parent

Property: Parent as Document[1668] (read-only)

Description
Access the parent of the object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.11 PositionFromLine

Method: PositionFromLine(nLine as Long) as Long

Description
Returns the start position of the line.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.12 ReplaceText

Method: ReplaceText(nPosFrom as Long, nPosTill as Long, sText as String)

Description
Replaces the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.13 SelectionEnd

Property: SelectionEnd as Long

Description
Returns/sets the text selection end position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.14 SelectionStart

Property: SelectionStart as Long

Description
Returns/sets the text selection start position.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.15 SelectText

Method: SelectText(nPosFrom as Long, nPosTill as Long)

Description
Selects the text in the specified range.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.28.16 SelText

Property: SelText as String

Description
Returns/sets the selected text.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.28.17 Text

Property: Text as String

Description
Returns/sets the document text.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29 WSDLDocumentationDlg

Properties and Methods

Standard automation properties
Application 1762
Parent 1768

Interaction and visibility properties
GlobalElementsAndTypesOnly 1764
OptionsDialogAction 1767
OutputFile 1767
OutputFileDialogAction 1767
SeparateSchemaDocument 1768
ShowProgressBar 1770
ShowResult 1770

Document generation options and methods
OutputFormat 1767
UseFixedDesign 1772
SPSFile 1772
EmbedDiagrams 1763
DiagramFormat 1763
MultipleOutputFiles 1766
EmbedCSSInHTML 1763
CreateDiagramsFolder 1762

IncludeAll 1764
IncludeBinding 1764
IncludeImportedWSDLFiles 1765
IncludeMessages 1765
IncludeOverview 1765
IncludePortType 1765
IncludeService 1766
IncludeTypes 1766

AllDetails 1762
ShowBindingDiagram 1768
ShowExtensibility 1769
ShowMessageParts 1769

ShowPort 1769
ShowPortTypeDiagram 1769
ShowPortTypeOperations 1770
ShowServiceDiagram 1771
ShowSourceCode 1771
ShowTypesDiagram 1771
ShowUsedBy 1772

Description
This object combines all options for WSDL document generation as they are available through user interface dialog boxes in XMLSpy. The document generation options are initialized with the values used during the last generation of WSDL documentation. However, before using the object you have to set the OutputFile 1767 property to a valid file path. Use OptionsDialogAction 1767, OutputFileDialogAction 1767 and ShowProgressBar 1770 to specify the level of user interaction desired. You can use IncludeAll 1764 and AllDetails 1762 to set whole option groups at once or the individual properties to operate on a finer granularity.

29.3.2.29.1 AllDetails

Method: AllDetails (i_bDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

Errors
4300 The object is no longer valid.

29.3.2.29.2 Application

Property: Application as Application 1580 (read-only)

Description
Access the XMLSpy application object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.3 CreateDiagramsFolder

Property: CreateDiagramsFolder as Boolean

Description

(C) 2015-2021 Altova GmbH

Set this property to true, to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is false.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.4 DiagramFormat

Property: DiagramFormat as SPYImageKind[1848]

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is PNG.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.5 EmbedCSSInHTML

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.6 EmbedDiagrams

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.7 GlobalElementsAndTypesOnly

Property: GlobalElementsAndTypesOnly as Boolean

Description
Returns/sets a value indicating whether a full Schema documentation is done or only Global Elements and Types are documented.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.8 IncludeAll

Method: IncludeAll (i_bInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
4300 The object is no longer valid.

29.3.2.29.9 IncludeBinding

Property: IncludeBinding as Boolean

Description
Set this property to true, to include bindings in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation 1685. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.29.10 IncludeImportedWSDLFiles

Property: IncludeImportedWSDLFiles as Boolean

Description
Set this property to true, to include imported WSDL files in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.11 IncludeMessages

Property: IncludeMessages as Boolean

Description
Set this property to true, to include messages in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.12 IncludeOverview

Property: IncludeOverview as Boolean

Description
Set this property to true, to include an overview in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.13 IncludePortType

Property: IncludePortType as Boolean

Description

Set this property to true, to include port types in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.14 IncludeService

Property: IncludeService as Boolean

Description
Set this property to true, to include services in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.15 IncludeTypes

Property: IncludeTypes as Boolean

Description
Set this property to true, to include types in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.16 MultipleOutputFiles

Property: MultipleOutputFiles as Boolean

Description
Set this property to true, to split the documentation files. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is false.

Errors
3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.29.17 OptionsDialogAction

Property: OptionsDialogAction as SPYDialogAction[1846]

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

29.3.2.29.18 OutputFile

Property: OutputFile as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to Document.GenerateWSDLDocumentation[1685].

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.19 OutputFileDialogAction

Property: OutputFileDialogAction as SPYDialogAction[1846]

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile[1767] should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

29.3.2.29.20 OutputFormat

Property: OutputFormat as SPYSchemaDocumentationFormat[1853]

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is HTML.

Errors
3900 The object is no longer valid.
3901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

29.3.2.29.21 Parent

Property: Parent as Dialogs[1664] (read-only)

Description
Access the parent of the object.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.22 SeparateSchemaDocument

Property: SeparateSchemaDocument as Boolean

Description
Returns/sets a value indicating whether the Schema documentation should be placed in a separate document.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.23 ShowBindingDiagram

Property: ShowBindingDiagram as Boolean

Description
Set this property to true, to show binding diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.29.24 ShowExtensibility

Property: ShowExtensibility as Boolean

Description
Set this property to true, to show service and binding extensibilities in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.25 ShowMessageParts

Property: ShowMessageParts as Boolean

Description
Set this property to true, to show message parts of messges in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.26 ShowPort

Property: ShowPort as Boolean

Description
Set this property to true, to show service ports in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.27 ShowPortTypeDiagram

Property: ShowPortTypeDiagram as Boolean

Description
Set this property to true, to show port type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.28 ShowPortTypeOperations

Property: ShowPortTypeOperations as Boolean

Description
Set this property to true, to show port type operations in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.29 ShowProgressBar

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.30 ShowResult

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in XMLSpy. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors

(C) 2015-2021 Altova GmbH

3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.31 ShowServiceDiagram

Property: ShowServiceDiagram as Boolean

Description
Set this property to true, to show service diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.32 ShowSourceCode

Property: ShowSourceCode as Boolean

Description
Set this property to true, to show source code for the includes in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.33 ShowTypesDiagram

Property: ShowTypesDiagram as Boolean

Description
Set this property to true, to show type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.34 ShowUsedBy

Property: ShowUsedBy as Boolean

Description
Set this property to true, to show the used-by relation for types, bindings and messages definitions in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDLDocumentation[1685]. The default for the first run is true.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.35 UseFixedDesign

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.29.36 SPSFile

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.30 WSDL20DocumentationDlg

Properties and Methods

Standard automation properties
Application[1774]
Parent[1779]

(C) 2015-2021 Altova GmbH

Interaction and visibility properties
GlobalElementsAndTypesOnly 1775
OptionsDialogAction 1778
OutputFile 1778
OutputFileDialogAction 1778
SeparateSchemaDocument 1779
ShowProgressBar 1781
ShowResult 1782

Document generation options and methods
OutputFormat 1779
UseFixedDesign 1783
SPSFile 1783
EmbedDiagrams 1775
DiagramFormat 1774
MultipleOutputFiles 1778
EmbedCSSInHTML 1775
CreateDiagramsFolder 1774

IncludeAll 1776
IncludeBinding 1776
IncludeImportedWSDLFiles 1776
IncludeInterface 1776
IncludeOverview 1777
IncludeService 1777
IncludeTypes 1777

AllDetails 1774
ShowBindingDiagram 1780
ShowExtensibility 1780
ShowEndpoint 1780
ShowFault 1780
ShowInterfaceDiagram 1781
ShowOperation 1781
ShowServiceDiagram 1782
ShowSourceCode 1782
ShowTypesDiagram 1783
ShowUsedBy 1783

Description
This object combines all options for WSDL document generation as they are available through user interface dialog boxes in XMLSpy. The document generation options are initialized with the values used during the last generation of WSDL documentation. However, before using the object you have to set the OutputFile 1778 property to a valid file path. Use OptionsDialogAction 1778, OutputFileDialogAction 1778 and ShowProgressBar 1781 to specify the level of user interaction desired. You can use IncludeAll 1776 and AllDetails 1774 to set whole option groups at once or the individual properties to operate on a finer granularity.

29.3.2.30.1 AllDetails

Method: AllDetails (i_bDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

Errors
4300 The object is no longer valid.

29.3.2.30.2 Application

Property: Application as Application[1580] (read-only)

Description
Access the XMLSpy application object.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.3 CreateDiagramsFolder

Property: CreateDiagramsFolder as Boolean

Description
Set this property to true, to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is false.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.4 DiagramFormat

Property: DiagramFormat as SPYImageKind[1848]

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is PNG.

(C) 2015-2021 Altova GmbH

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.5 EmbedCSSInHTML

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.6 EmbedDiagrams

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.7 GlobalElementsAndTypesOnly

Property: GlobalElementsAndTypesOnly as Boolean

Description
Returns/sets a value indicating whether a full Schema documentation is done or only Global Elements and Types are documented.

Errors
3900 The object is no longer valid.
3901 Invalid address for the return parameter was specified.

29.3.2.30.8 IncludeAll

Method: IncludeAll (i_bInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
4300 The object is no longer valid.

29.3.2.30.9 IncludeBinding

Property: IncludeBinding as Boolean

Description
Set this property to true, to include bindings in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.10 IncludeImportedWSDLFiles

Property: IncludeImportedWSDLFiles as Boolean

Description
Set this property to true, to include imported WSDL files in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.11 IncludeInterface

Property: IncludeInterface as Boolean

Description
Set this property to true, to include interfaces in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

(C) 2015-2021 Altova GmbH

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.12 IncludeOverview

Property: IncludeOverview as Boolean

Description
Set this property to true, to include an overview in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.13 IncludeService

Property: IncludeService as Boolean

Description
Set this property to true, to include services in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.14 IncludeTypes

Property: IncludeTypes as Boolean

Description
Set this property to true, to include types in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.15 MultipleOutputFiles

Property: MultipleOutputFiles as Boolean

Description
Set this property to true, to split the documentation files. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is false.

Errors
4300 The object is no longer valid.
4301 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

29.3.2.30.16 OptionsDialogAction

Property: OptionsDialogAction as SPYDialogAction[1846]

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
4300 The object is no longer valid.
4301 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

29.3.2.30.17 OutputFile

Property: OutputFile as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to Document.GenerateWSDL20Documentation[1685].

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.18 OutputFileDialogAction

Property: OutputFileDialogAction as SPYDialogAction[1846]

Description

(C) 2015-2021 Altova GmbH

To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile[1778] should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK*.

Errors

4300	The object is no longer valid.
4301	Invalid value has been used to set the property. Invalid address for the return parameter was specified.

29.3.2.30.19 OutputFormat

Property: OutputFormat as SPYSchemaDocumentationFormat[1853]

Description
Defines the kind of documentation that will be generated: HTML (value=`0`), MS-Word (value=`1`), or RTF (value=`2`). The property gets initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is HTML.

Errors

4300	The object is no longer valid.
4301	Invalid value has been used to set the property. Invalid address for the return parameter was specified.

29.3.2.30.20 Parent

Property: Parent as Dialogs[1664] (read-only)

Description
Access the parent of the object.

Errors

4300	The object is no longer valid.
4301	Invalid address for the return parameter was specified.

29.3.2.30.21 SeparateSchemaDocument

Property: SeparateSchemaDocument as Boolean

Description
Returns/sets a value indicating whether the Schema documentation should be placed in a separate document.

Errors

3900	The object is no longer valid.
3901	Invalid address for the return parameter was specified.

29.3.2.30.22 ShowBindingDiagram

Property: ShowBindingDiagram as Boolean

Description
Set this property to true, to show binding diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.23 ShowEndpoint

Property: ShowEndpoint as Boolean

Description
Set this property to true, to show service endpoints in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.24 ShowExtensibility

Property: ShowExtensibility as Boolean

Description
Set this property to true, to show service and binding extensibilities in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.25 ShowFault

Property: ShowFault as Boolean

(C) 2015-2021 Altova GmbH

Description
Set this property to true, to show faults in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.26 ShowInterfaceDiagram

Property: ShowInterfaceDiagram as Boolean

Description
Set this property to true, to show interface diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.27 ShowOperation

Property: ShowOperation as Boolean

Description
Set this property to true, to show interface and binding operations in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.28 ShowProgressBar

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.29 ShowResult

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in XMLSpy. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.30 ShowServiceDiagram

Property: ShowServiceDiagram as Boolean

Description
Set this property to true, to show service diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.31 ShowSourceCode

Property: ShowSourceCode as Boolean

Description
Set this property to true, to show source code for the includes in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.30.32 ShowTypesDiagram

Property: ShowTypesDiagram as Boolean

Description
Set this property to true, to show type diagrams in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.33 ShowUsedBy

Property: ShowUsedBy as Boolean

Description
Set this property to true, to show the used-by relation for types, bindings and messages definitions in the WSDL documentation. The property is initialized with the value used during the last call to Document.GenerateWSDL20Documentation[1685]. The default for the first run is true.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.34 SPSFile

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.30.35 UseFixedDesign

Property: UseFixedDesign as Boolean

Description

Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors

4300 The object is no longer valid.
4301 Invalid address for the return parameter was specified.

29.3.2.31 XBRLDocumentationDlg

Properties and Methods

Standard automation properties
Application 1785
Parent 1790

Interaction and visibility properties
OptionsDialogAction 1789
OutputFile 1789
OutputFileDialogAction 1789
ShowProgressBar 1793
ShowResult 1794

Document generation options and methods
OutputFormat 1789
UseFixedDesign 1795
SPSFile 1794
EmbedDiagrams 1786
DiagramFormat 1786
EmbedCSSInHTML 1786
CreateDiagramsFolder 1785

IncludeAll 1787
IncludeOverview 1788
IncludeNamespacePrefixes 1788
IncludeGlobalElements 1787
IncludeDefinitionLinkroles 1787
IncludePresentationLinkroles 1788
IncludeCalculationLinkroles 1787

AllDetails 1785
ShowDiagram 1791
ShowSubstitutiongroup 1794
ShowItemtype 1792
ShowBalance 1791
ShowPeriod 1793
ShowAbstract 1790
ShowNillable 1792
ShowLabels 1792
ShowReferences 1793
ShowLinkbaseReferences 1792

ShortQualifiedName 1790

(C) 2015-2021 Altova GmbH

ShowImportedElements[1791]

Description
This object combines all options for XBRL document generation as they are available through user interface dialog boxes in XMLSpy. The document generation options are initialized with the values used during the last generation of XBRL documentation. However, before using the object you have to set the OutputFile[1789] property to a valid file path. Use OptionsDialogAction[1789], OutputFileDialogAction[1789] and ShowProgressBar[1793] to specify the level of user interaction desired. You can use IncludeAll[1787] and AllDetails[1785] to set whole option groups at once or the individual properties to operate on a finer granularity.

29.3.2.31.1 AllDetails

Method: AllDetails (i_bDetailsOn as Boolean)

Description
Use this method to turn all details options on or off.

Errors
4400 The object is no longer valid.

29.3.2.31.2 Application

Property: Application as Application[1580] (read-only)

Description
Access the XMLSpy application object.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.3 CreateDiagramsFolder

Property: CreateDiagramsFolder as Boolean

Description
Set this property to true, to create a directory for the created images. Otherwise the diagrams will be created next to the documentation. This property is only available when the diagrams are not embedded. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is false.

Errors
4400 The object is no longer valid.

4401 Invalid address for the return parameter was specified.

29.3.2.31.4 DiagramFormat

Property: DiagramFormat as SPYImageKind 1848

Description
This property specifies the generated diagram image type. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation 1685. The default for the first run is PNG.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.5 EmbedCSSInHTML

Property: EmbedCSSInHTML as Boolean

Description
Set this property to true, to embed the CSS data in the generated HTML document. Otherwise a separate file will be created and linked. This property is only available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation 1685. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.6 EmbedDiagrams

Property: EmbedDiagrams as Boolean

Description
Set this property to true, to embed the diagrams in the generated document. This property is not available for HTML documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation 1685. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.31.7 IncludeAll

Method: IncludeAll (i_bInclude as Boolean)

Description
Use this method to mark or unmark all include options.

Errors
4400 The object is no longer valid.

29.3.2.31.8 IncludeCalculationLinkroles

Property: IncludeCalculationLinkroles as Boolean

Description
Set this property to true, to include calculation linkroles in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.9 IncludeDefinitionLinkroles

Property: IncludeDefinitionLinkroles as Boolean

Description
Set this property to true, to include definition linkroles in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.10 IncludeGlobalElements

Property: IncludeGlobalElements as Boolean

Description
Set this property to true, to include global elements in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.11 IncludeNamespacePrefixes

Property: IncludeNamespacePrefixes as Boolean

Description
Set this property to true, to include namespace prefixes in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.12 IncludeOverview

Property: IncludeOverview as Boolean

Description
Set this property to true, to include an overview in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.13 IncludePresentationLinkroles

Property: IncludePresentationLinkroles as Boolean

Description
Set this property to true, to include presentation linkroles in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.31.14 OptionsDialogAction

Property: OptionsDialogAction as SPYDialogAction[1846]

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
4400 The object is no longer valid.
4401 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

29.3.2.31.15 OutputFile

Property: OutputFile as String

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to Document.GenerateXBRLDocumentation[1685].

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.16 OutputFileDialogAction

Property: OutputFileDialogAction as SPYDialogAction[1846]

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile[1789] should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors
4400 The object is no longer valid.
4401 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

29.3.2.31.17 OutputFormat

Property: OutputFormat as SPYSchemaDocumentationFormat[1853]

Description
Defines the kind of documentation that will be generated: HTML (value=0), MS-Word (value=1), or RTF (value=2). The property gets initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is HTML.

Errors
4400 The object is no longer valid.
4401 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

29.3.2.31.18 Parent

Property: Parent as Dialogs[1664] (read-only)

Description
Access the parent of the object.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.19 ShortQualifiedName

Property: ShortQualifiedName as Boolean

Description
Set this property to true, to use short qualified names in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.20 ShowAbstract

Property: ShowAbstract as Boolean

Description
Set this property to true, to show abstracts in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.31.21 ShowBalance

Property: ShowBalance as Boolean

Description
Set this property to true, to show balances in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.22 ShowDiagram

Property: ShowDiagram as Boolean

Description
Set this property to true, to show diagrams in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.23 ShowImportedElements

Property: ShowImportedElements as Boolean

Description
Set this property to true, to show imported elements in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.24 ShowItemtype

Property: ShowItemtype as Boolean

Description
Set this property to true, to show item types in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.25 ShowLabels

Property: ShowLabels as Boolean

Description
Set this property to true, to show labels in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.26 ShowLinkbaseReferences

Property: ShowLinkbaseReferences as Boolean

Description
Set this property to true, to show linkbase references in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.27 ShowNillable

Property: ShowNillable as Boolean

Description

(C) 2015-2021 Altova GmbH

Set this property to true, to show nillable properties in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.28 ShowPeriod

Property: ShowPeriod as Boolean

Description
Set this property to true, to show periods in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.29 ShowProgressBar

Property: ShowProgressBar as Boolean

Description
Set this property to true, to make the window showing the document generation progress visible. Use false, to hide it. Default is false.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.30 ShowReferences

Property: ShowReferences as Boolean

Description
Set this property to true, to show references in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.31 ShowResult

Property: ShowResult as Boolean

Description
Set this property to true, to automatically open the resulting document when generation was successful. HTML documentation will be opened in XMLSpy. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.32 ShowSubstitutiongroup

Property: ShowSubstitutiongroup as Boolean

Description
Set this property to true, to show substitution groups in the XBRL documentation. The property is initialized with the value used during the last call to Document.GenerateXBRLDocumentation[1685]. The default for the first run is true.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.31.33 SPSFile

Property: SPSFile as String

Description
Full path and name of the SPS file that will be used to generate the documentation.

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.31.34 UseFixedDesign

Property: UseFixedDesign as Boolean

Description
Specifies whether the documentation should be created with a fixed design or with a design specified by a SPS file (which requires StyleVision).

Errors
4400 The object is no longer valid.
4401 Invalid address for the return parameter was specified.

29.3.2.32 XMLData

Properties
Kind 1805
Name 1805
TextValue 1806

HasChildren 1803
MayHaveChildren 1805
Parent 1805

Methods
GetFirstChild 1800
GetNextChild 1801
GetCurrentChild 1800

InsertChild 1803
InsertChildAfter 1804
InsertChildBefore 1804
AppendChild 1796

EraseAllChildren 1797
EraseChild 1798
EraseCurrentChild 1798

IsSameNode 1804

CountChildren 1797
CountChildrenKind 1797

GetChild 1799
GetChildAttribute 1799
GetChildElement 1799
GetChildKind 1800
GetNamespacePrefixForURI 1801

HasChildrenKind 1803
SetTextValueXMLEncoded 1806

Description
The XMLData interface provides direct XML-level access to a document. You can read and directly modify the XML representation of the document. However, please, note the following restrictions:

- The XMLData representation is only valid when the document is shown in grid view or authentic view.
- When in authentic view, additional XMLData elements are automatically inserted as parents of each visible document element. Typically this is an XMLData of kind spyXMLDataElement with the Name[1805] property set to 'Text'.
- When you use the XMLData interface while in a different view mode you will not receive errors, but changes are not reflected to the view and might get lost during the next view switch.

Note also:

- Setting a new text value for an XML element is possible if the element does not have non-text children. A text value can be set even if the element has attributes.
- When setting a new text value for an XML element which has more than one text child, the latter will be deleted and replaced by one new text child.
- When reading the text value of an XML element which has more than one text child, only the value of the first text child will be returned.

Objects of this class represent the different atomic parts of an XML document. See the enumeration type SPYXMLDataKind[1857] for the available part types. Each part knows its children, thus forming a XMLData tree with Document.RootElement[1691] at its top. To get the top element of the document content - ignoring the XML header - use Document.DataRoot[1678]. For an examples on how to traverse the XMLData tree, see GetNextChild[1801].

29.3.2.32.1 AppendChild

Declaration: AppendChild (*pNewData* as XMLData[1795])

Description
AppendChild appends pNewData as last child to the XMLData object.

Errors

1500	The XMLData object is no longer valid.
1505	Invalid XMLData kind was specified.
1506	Invalid address for the return parameter was specified.
1507	Element cannot have Children
1512	Cyclic insertion - new data element is already part of document
1514	Invalid XMLData kind was specified for this position.
1900	Document must not be modified

Example

```
Dim objCurrentParent As XMLData
Dim objNewChild As XMLData

Set objNewChild = objSpy.ActiveDocument.CreateChild(spyXMLDataElement)
Set objCurrentParent = objSpy.ActiveDocument.RootElement
```

```
objCurrentParent.AppendChild objNewChild

Set objNewChild = Nothing
```

29.3.2.32.2 CountChildren

Declaration: CountChildren as long

Description
CountChildren gets the number of children.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.

29.3.2.32.3 CountChildrenKind

Declaration: CountChildrenKind (*nKind* as SPYXMLDataKind[1857]) as long

Description
CountChildrenKind gets the number of children of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.

29.3.2.32.4 EraseAllChildren

Declaration: EraseAllChildren

Description
EraseAllChildren deletes all associated children of the XMLData object.

Errors
1500 The XMLData object is no longer valid.
1900 Document must not be modified

Example
The sample erases all elements of the active document.

```
Dim objCurrentParent As XMLData
```

```
Set objCurrentParent = objSpy.ActiveDocument.RootElement
objCurrentParent.EraseAllChildren
```

29.3.2.32.5 EraseChild

Method: EraseChild (Child as XMLData[1795])

Description
Deletes the given child node.

Errors
1500 Invalid object.
1506 Invalid input xml
1510 Invalid parameter.

29.3.2.32.6 EraseCurrentChild

Declaration: EraseCurrentChild

Description
EraseCurrentChild deletes the current XMLData child object. Before you call EraseCurrentChild you must initialize an internal iterator with XMLData.GetFirstChild[1800]. After deleting the current child, EraseCurrentChild increments the internal iterator of the XMLData element. No error is returned when the last child gets erased and the iterator is moved past the end of the child list. The next call to EraseCurrentChild however, will return error 1503.

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object, or the iterator points past the last child.
1900 Document must not be modified

Examples

```
// ---------------------------------------
// XMLSpy scripting environment - JScript
// erase all children of XMLData
// ---------------------------------------
// let's get an XMLData element, we assume that the
// cursor selects the parent of a list in grid view
var objList = Application.ActiveDocument.GridView.CurrentFocus;

// the following line would be shorter, of course
//      objList.EraseAllChildren ();

// but we want to demonstrate the usage of EraseCurrentChild
if ((objList != null) && (objList.HasChildren))
{
      try
```

```
        {
                objEle = objList.GetFirstChild(-1);
                while (objEle != null)
                        objList.EraseCurrentChild();
                        // no need to call GetNextChild
        }
        catch (err)
                // 1503 - we reached end of child list
                { if ((err.number & 0xffff) != 1503) throw (err); }
}
```

29.3.2.32.7 GetChild

Declaration: GetChild (*position* as long) as XMLData[1795]

Return Value
Returns an XML element as XMLData object.

Description
GetChild() returns a reference to the child at the given index (zero-based).

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

29.3.2.32.8 GetChildAttribute

Method: GetChildAttribute (strName as string) child as XMLData object (`NULL` on error)

Description
Retrieves the attribute having the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

29.3.2.32.9 GetChildElement

Method: GetChildElement (strName as string, nIndex as long) child as XMLData object (`NULL` on error)

Description
Retrieves the Nth child element with the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

29.3.2.32.10 GetChildKind

Declaration: GetChildKind (*position* as long, *nKind* as SPYXMLDataKind[1857]) as XMLData[1795]

Return Value
Returns an XML element as XMLData object.

Description
GetChildKind() returns a reference to a child of this kind at the given index (zero-based). The position parameter is relative to the number of children of the specified kind and not to all children of the object.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

29.3.2.32.11 GetCurrentChild

Declaration: GetCurrentChild as XMLData[1795]

Return Value
Returns an XML element as XMLData object.

Description
GetCurrentChild gets the current child. Before you call GetCurrentChild you must initialize an internal iterator with XMLData.GetFirstChild[1800].

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

29.3.2.32.12 GetFirstChild

Declaration: GetFirstChild (*nKind* as SPYXMLDataKind[1857]) as XMLData[1795]

Return Value
Returns an XML element as XMLData object.

(C) 2015-2021 Altova GmbH

Description
GetFirstChild initializes a new iterator and returns the first child. Set nKind = -1 to get an iterator for all kinds of children.
REMARK: The iterator is stored inside the XMLData object and gets destroyed when the XMLData object gets destroyed. Be sure to keep a reference to this object as long as you want to use GetCurrentChild[1800], GetNextChild[1801] or EraseCurrentChild[1798].

Errors
1500 The XMLData object is no longer valid.
1501 Invalid XMLData kind was specified.
1504 Element has no children of specified kind.
1510 Invalid address for the return parameter was specified.

Example
See the example at XMLData.GetNextChild[1801].

29.3.2.32.13 GetNamespacePrefixForURI

Method: GetNamespacePrefixForURI (strURI as string) strNS as string

Description
Returns the namespace prefix of the supplied URI.

Errors
1500 Invalid object.
1510 Invalid parameter.

29.3.2.32.14 GetNextChild

Declaration: GetNextChild as XMLData[1795]

Return Value
Returns an XML element as XMLData object.

Description
GetNextChild steps to the next child of this element. Before you call GetNextChild you must initialize an internal iterator with XMLData.GetFirstChild[1800].

Check for the last child of the element as shown in the sample below.

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------------
' VBA code snippet - iterate XMLData children
' ------------------------------------------------
On Error Resume Next
Set objParent = objSpy.ActiveDocument.RootElement

'get elements of all kinds
Set objCurrentChild = objParent.GetFirstChild(-1)

Do
        'do something useful with the child

        'step to next child
        Set objCurrentChild = objParent.GetNextChild
Loop Until (Err.Number - vbObjectError = 1503)


// ----------------------------------------
// XMLSpy scripting environment - JScript
// iterate through children of XMLData
// ----------------------------------------
try
{
        var objXMLData = ... // initialize somehow
        var objChild = objXMLData.GetFirstChild(-1);

        while (true)
        {
                // do something usefull with objChild

                objChild = objXMLData.GetNextChild();
        }
}
catch (err)
{
        if ((err.number & 0xffff) == 1504)
                ; // element has no children
        else if ((err.number & 0xffff) == 1503)
                ;       // last child reached
        else
                throw (err);
}
```

29.3.2.32.15 GetTextValueXMLDecoded

Method: GetTextValueXMLDecoded ()as string

Description
Gets the decoded text value of the XML.

Errors
1500 Invalid object.

(C) 2015-2021 Altova GmbH

1510 Invalid parameter.

29.3.2.32.16 HasChildren

Declaration: HasChildren as Boolean

Description
The property is true if the object is the parent of other XMLData objects. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

29.3.2.32.17 HasChildrenKind

Declaration: HasChildrenKind (*nKind* as SPYXMLDataKind[1857]) as Boolean

Description
The method returns true if the object is the parent of other XMLData objects of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

29.3.2.32.18 InsertChild

Declaration: InsertChild (*pNewData* as XMLData[1795])

Description
InsertChild inserts the new child before the current child (see also XMLData.GetFirstChild[1800], XMLData.GetNextChild[1801] to set the current child).

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1505 Invalid XMLData kind was specified.
1506 Invalid address for the return parameter was specified.
1507 Element cannot have Children
1512 Cyclic insertion - new data element is already part of document
1514 Invalid XMLData kind was specified for this position.
1900 Document must not be modified

29.3.2.32.19 InsertChildAfter

Method: InsertChildAfter (Node as XMLData, NewData as XMLData)

Description
Inserts a new XML node (supplied with the second parameter) after the specified node (first parameter).

Errors
1500 Invalid object.
1506 Invalid input xml
1507 No children allowed
1510 Invalid parameter.
1512 Child is already added
1514 Invalid kind at position

29.3.2.32.20 InsertChildBefore

Method: InsertChildBefore (Node as XMLData, NewData as XMLData)

Description
Inserts a new XML node (supplied with the second parameter) before the specified node (first parameter).

Errors
1500 Invalid object.
1506 Invalid input xml
1507 No children allowed
1510 Invalid parameter.
1512 Child is already added
1514 Invalid kind at position

29.3.2.32.21 IsSameNode

Declaration: IsSameNode (*pNodeToCompare* as XMLData[1795]) as Boolean

Description

Returns true if pNodeToCompare references the same node as the object itself.

Errors
1500 The XMLData object is no longer valid.
1506 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

29.3.2.32.22 Kind

Declaration: Kind as SPYXMLDataKind 1857

Description
Kind of this XMLData object. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

29.3.2.32.23 MayHaveChildren

Declaration: MayHaveChildren as Boolean

Description
Indicates whether it is allowed to add children to this XMLData object.
This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

29.3.2.32.24 Name

Declaration: Name as String

Description
Used to modify and to get the name of the XMLData object.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

29.3.2.32.25 Parent

Declaration: Parent as XMLData 1795

Return value
Parent as XMLData object. Nothing (or NULL) if there is no parent element.

Description
Parent of this element. This property is read-only.

Errors
1500 The XMLData object is no longer valid.

1510 Invalid address for the return parameter was specified.

29.3.2.32.26 SetTextValueXMLEncoded

Method: SetTextValueXMLEncoded (*strVal* as String[1857])

Description
Sets the encoded text value of the XML.

Errors
1500 Invalid object.
1513 Modification not allowed.

29.3.2.32.27 TextValue

Declaration: TextValue as String

Description
Used to modify and to get the text value of this XMLData object.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

29.3.3 Interfaces (obsolete)

Interfaces contained in this book are obsolete. It is recommended to migrate your applications to the new interfaces. See the different properties and methods in this book for migration hints.

29.3.3.1 AuthenticEvent (obsolete)

Superseded by AuthenticView[1634] and AuthenticRange[1606]

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interface. All functionality available up to now in DocEditView[1821], DocEditSelection[1819], DocEditEvent[1806] and

(C) 2015-2021 Altova GmbH

DocEditDataTransfer[1602] is now available via AuthenticView[1634], AuthenticRange[1606] and AuthenticDataTransfer[1601]. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

Properties

altKey[1807]
altLeft[1808]
ctrlKey[1812]
ctrlLeft[1813]
shiftKey[1816]
shiftLeft[1817]

keyCode[1815]
repeat[1816]

button[1809]

clientX[1811]
clientY[1811]

dataTransfer[1813]

srcElement[1818]
fromElement[1814]

propertyName[1815]

cancelBubble[1810]
returnValue[1816]

type[1818]

Description
DocEditEvent interface.

29.3.3.1.1 altKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent[1639] (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent[1640] (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver[1638] (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.altKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//         MsgBox ("alt key is down");
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long i_nKeyCode,
SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nVirtualKeyStatus & spyAltKeyMask)
        MsgBox ("alt key is down");
}
```

Declaration: altKey as Boolean

Description
True if the right ALT key is pressed.

29.3.3.1.2 altLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent 1639 (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent 1640 (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver 1638 (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyDown ()
// {
//     if (Application.ActiveDocument.DocEditView.event.altKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//         MsgBox ("alt key is down");
// }
// use now:
```

```
function On_AuthenticView_KeyDown (SPYKeyEvent i_eKeyEvent, long i_nKeyCode, SPYVirtualKeyMask
i_nVirtualKeyStatus)
{
    if (i_nVirtualKeyStatus & spyAltKeyMask)
        MsgBox ("alt key is down");
}
```

Declaration: altLeft as Boolean

Description
True if the left ALT key is pressed.

29.3.3.1.3 button (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent 1640 (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver 1638 (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditButtonDown ()
// {
//     if (Application.ActiveDocument.DocEditView.event.button == 1)
//         MsgBox ("left mouse button down detected");
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos, SPYMouseEvent i_eMouseEvent,
IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyLeftButtonDownMask)
        MsgBox ("left mouse button down detected");
}
```

Declaration: button as long

Description
Specifies which mouse button is pressed:

0	No button is pressed.
1	Left button is pressed.
2	Right button is pressed.
3	Left and right buttons are both pressed.

4 Middle button is pressed.
5 Left and middle buttons both are pressed.
6 Right and middle buttons are both pressed.
7 All three buttons are pressed.

29.3.3.1.4 cancelBubble (obsolete)

Superseded by the boolean return value of following event handler functions
AuthenticView.OnKeyboardEvent[1639] (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent[1640] (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver[1638] (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

Returning *true* from an event handler function signals that the event has beend handled and normal event handling should be aborted.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.keyCode == 0x20)
//     {
//       // cancel key processing, swallow spaces :-)
//       Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//     }
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long i_nKeyCode,
SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nKeyCode == 0x20)
        return true;  // cancel key processing, swallow spaces :-)
}
```

Declaration: cancelBubble as Boolean

Description
Set cancelBubble to TRUE if the default event handler should not be called.

(C) 2015-2021 Altova GmbH

29.3.3.1.5 clientX (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent 1640 (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop 1636 (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver 1638 (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     MsgBox ("moving over " + Application.ActiveDocument.DocEditView.event.clientX +
//             "/" + Application.ActiveDocument.DocEditView.event.clientY);
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos, SPYMouseEvent i_eMouseEvent,
IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyMouseMoveMask)
        MsgBox ("moving over " + i_nXPos + "/" + i_nYPos);
}
```

Declaration: clientX as long

Description
X value of the current mouse position in client coordinates.

29.3.3.1.6 clientY (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent 1640 (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop 1636 (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver 1638 (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
```

```
// function On_DocEditMouseMove ()
// {
//     MsgBox ("moving over " + Application.ActiveDocument.DocEditView.event.clientX +
//             "/" + Application.ActiveDocument.DocEditView.event.clientY);
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos, SPYMouseEvent i_eMouseEvent,
IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyMouseMoveMask)
        MsgBox ("moving over " + i_nXPos + "/" + i_nYPos);
}
```

Declaration: clientY as long

Description
Y value of the current mouse position in client coordinates.

29.3.3.1.7 ctrlKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent 1639 (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent 1640 (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver 1638 (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     if (Application.ActiveDocument.DocEditView.event.ctrlKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//         MsgBox ("control key is down");
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos, SPYMouseEvent i_eMouseEvent,
IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyCtrlKeyMask)
        MsgBox ("control key is down");
}
```

Declaration: ctrlKey as Boolean

(C) 2015-2021 Altova GmbH

Description
True if the right CTRL key is pressed.

29.3.3.1.8 ctrlLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent 1639 (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent 1640 (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver 1638 (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     if (Application.ActiveDocument.DocEditView.event.ctrlKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//       MsgBox ("control key is down");
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos, SPYMouseEvent i_eMouseEvent,
IAuthenticRange *i_ipRange)
{
   if (i_eMouseEvent & spyCtrlKeyMask)
      MsgBox ("control key is down");
}
```

Declaration: ctrlLeft as Boolean

Description
True if the left CTRL key is pressed.

29.3.3.1.9 dataTransfer (obsolete)

Superseded by parameters to
AuthenticView.OnBeforeDrop 1636 (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver 1638 (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDrop ()
// {
//     if (Application.ActiveDocument.DocEditView.event.dataTransfer != null)
//         if (! Application.ActiveDocument.DocEditView.event.dataTransfer.ownDrag)
//         {
//           // cancel key processing, don't drop foreign objects :-)
//           Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//         }
// }
// use now:
function On_AuthenticView_BeforeDrop (long i_nXPos, long i_nYPos,
                                      IAuthenticRange *i_ipRange,
                                      IAuthenticDataTransfer *i_ipData)
{
    if (i_ipRange != null)
        if (! i_ipRange.ownDrag)
            return true;  // cancel key processing, don't drop foreign objects :-)

    return false;
}
```

Declaration: dataTransfer as Variant

Description
Property dataTransfer.

29.3.3.1.10 fromElement (obsolete)

Not supported

Declaration: fromElement as Variant (not supported)

Description
Currently no event sets this property.

(C) 2015-2021 Altova GmbH

29.3.3.1.11 keyCode (obsolete)

Superseded by a parameter to AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.keyCode == 0x20)
//     {
//        // cancel key processing, swallow spaces :-)
//        Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//     }
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long i_nKeyCode,
SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nKeyCode == 0x20)
        return true;  // cancel key processing, swallow spaces :-)
}
```

Declaration: keyCode as long

Description
Keycode of the currently pressed key. This property is read-write.

29.3.3.1.12 propertyName (obsolete)

Not supported

Declaration: propertyName as String (not supported)

Description
Currently no event sets this property.

29.3.3.1.13 repeat (obsolete)

Not supported

Declaration: repeat as Boolean (not supported)

Description
True if the onkeydown event is repeated.

29.3.3.1.14 returnValue (obsolete)

No longer supported

Declaration: returnValue as Variant

Description
Use returnValue to set a return value for your event handler.

29.3.3.1.15 shiftKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent[1639] (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent[1640] (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver[1638] (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDragOver ()
// {
//     if (Application.ActiveDocument.DocEditView.event.shiftKey ||
//         Application.ActiveDocument.DocEditView.event.shiftLeft)
//       MsgBox ("shift key is down");
// }
// use now:
function On_AuthenticView_DragOver (long i_nXPos, long i_nYPos,
```

```
                                    SPYMouseEvent i_eMouseEvent,
                                    IAuthenticRange *i_ipRange,
                                    IAuthenticDataTransfer *i_ipData)
{
   if (i_eMouseEvent & spyShiftKeyMask)
      MsgBox ("shift key is down");
}
```

Declaration: shiftKey as Boolean

Description
True if the right SHIFT key is pressed.

29.3.3.1.16 shiftLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent[1639] (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent[1640] (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver[1638] (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDragOver ()
// {
//    if (Application.ActiveDocument.DocEditView.event.shiftKey ||
//        Application.ActiveDocument.DocEditView.event.shiftLeft)
//       MsgBox ("shift key is down");
// }
// use now:
function On_AuthenticView_DragOver (long i_nXPos, long i_nYPos,
                                    SPYMouseEvent i_eMouseEvent,
                                    IAuthenticRange *i_ipRange,
                                    IAuthenticDataTransfer *i_ipData)
{
   if (i_eMouseEvent & spyShiftKeyMask)
      MsgBox ("shift key is down");
}
```

Declaration: shiftLeft as Boolean

Description
True if the left SHIFT key is pressed.

29.3.3.1.17 srcElement (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent[1640] (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop[1636] (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver[1638] (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

With the new event handler function, a range object selecting this element is provided instead of the XMLData element currently below the mouse cursor.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     var objEvent = Application.ActiveDocument.DocEditView.event;
//     if (objEvent.srcElement != null)
//         MsgBox ("moving over " + objEvent.srcElement.Parent.Name);
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos, SPYMouseEvent i_eMouseEvent,
IAuthenticRange *i_ipRange)
{
    if ((i_eMouseEvent & spyMouseMoveMask) &&
        (i_ipRange != null))
        MsgBox ("moving over " + i_ipRange.FirstXMLData.Parent.Name);
}
```

Declaration: srcElement as Variant

Description
Element which fires the current event. This is usually an XMLData[1795] object.

29.3.3.1.18 type (obsolete)

Not supported

(C) 2015-2021 Altova GmbH

Declaration: type as String (not supported)

Description
Currently no event sets this property.

29.3.3.2 AuthenticSelection (obsolete)

Superseded by AuthenticRange[1606]

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interface. All functionality available up to now in DocEditView[1821], DocEditSelection[1819], DocEditEvent[1806] and DocEditDataTransfer[1601] is now available via AuthenticView[1634], AuthenticRange[1606] and AuthenticDataTransfer[1601]. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

Properties
Start[1820]
StartTextPosition[1820]
End[1819]
EndTextPosition[1820]

29.3.3.2.1 End (obsolete)

Superseded by AuthenticRange.LastXMLData[1625]

```
// ----- javascript sample -----
// instead of:
// var objXMLData = Application.ActiveDocument.DocEditView.CurrentSelection.End;
// use now:
var objXMLData = Application.ActiveDocument.AuthenticView.Selection.LastXMLData;
```

Declaration: End as XMLData[1795]

Description
XML element where the current selection ends.

29.3.3.2.2 EndTextPosition (obsolete)

Superseded by AuthenticRange.LastXMLDataOffset 1626

```
// ----- javascript sample -----
// instead of:
// var nOffset = Application.ActiveDocument.DocEditView.CurrentSelection.EndTextPosition;
// use now:
var nOffset = Application.ActiveDocument.AuthenticView.Selection.LastXMLDataOffset;
```

Declaration: EndTextPosition as long

Description
Position in DocEditSelection.End.TextValue 1819 where the selection ends.

29.3.3.2.3 Start (obsolete)

Superseded by AuthenticRange.FirstXMLData 1613

```
// ----- javascript sample -----
// instead of:
// var objXMLData = Application.ActiveDocument.DocEditView.CurrentSelection.Start;
// use now:
var objXMLData = Application.ActiveDocument.AuthenticView.Selection.FirstXMLData;
```

Declaration: Start as XMLData 1795

Description
XML element where the current selection starts.

29.3.3.2.4 StartTextPosition (obsolete)

Superseded by AuthenticRange.FirstXMLDataOffset 1614

```
// ----- javascript sample -----
// instead of:
// var nOffset = Application.ActiveDocument.DocEditView.CurrentSelection.StartTextPosition;
```


(C) 2015-2021 Altova GmbH

```
// use now:
var nOffset = Application.ActiveDocument.AuthenticView.Selection.FirstXMLDataOffset;
```

Declaration: StartTextPosition as long

Description
Position in DocEditSelection.Start.TextValue[1820] where the selection starts.

29.3.3.3 OldAuthentictView (obsolete)

Superseded by AuthenticView[1634] and AuthenticRange[1606]

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interfaces. All functionality available up to now in DocEditView[1821], DocEditSelection[1819], DocEditEvent[1806] and DocEditDataTransfer[1602] is now available via AuthenticView[1634], AuthenticRange[1606] and AuthenticDataTransfer[1601]. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

Methods

LoadXML[1837]
SaveXML[1841]

EditClear[1824]
EditCopy[1824]
EditCut[1825]
EditPaste[1825]
EditRedo[1826]
EditSelectAll[1826]
EditUndo[1826]

RowAppend[1838]
RowDelete[1839]
RowDuplicate[1839]
RowInsert[1840]
RowMoveDown[1840]
RowMoveUp[1841]

ApplyTextState[1822]

IsTextStateApplied 1836
IsTextStateEnabled 1837

MarkUpView 1838

SelectionSet 1842
SelectionMoveTabOrder 1842

GetNextVisible 1829
GetPreviousVisible 1830

GetAllowedElements 1827

Properties
CurrentSelection 1823

event 1827

XMLRoot 1843

IsEditClearEnabled 1830
IsEditCopyEnabled 1831
IsEditCutEnabled 1831
IsEditPasteEnabled 1832
IsEditRedoEnabled 1832
IsEditUndoEnabled 1833

IsRowAppendEnabled 1833
IsRowDeleteEnabled 1834
IsRowDuplicateEnabled 1834
IsRowInsertEnabled 1835
IsRowMoveDownEnabled 1835
IsRowMoveUpEnabled 1836

Description
Interface for Authentic View.

29.3.3.3.1 ApplyTextState (obsolete)

Superseded by AuthenticRange.PerformAction 1629

Use spyAuthenticApply for the eAction parameter. The PerformAction method allows to apply text state attributes to any range of the document, not only the current UI selection.

// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.ApplyTextState ("bold");
// use now:

```
if (! Application.ActiveDocument.AuthenticView.Selection.PerformAction (spyAuthenticApply, "bold"))
    MsgBox ("Error: can't set current selection to bold");
```

Declaration: ApplyTextState (*elementName* as String)

Description
Applies or removes the text state defined by the parameter elementName. Common examples for the parameter elementName would be strong and italic.

In an XML document there are segments of data, which may contain sub-elements. For example consider the following HTML:

```
<b>fragment</b>
```

The HTML tag <b> will cause the word fragment to be bold. However, this only happens because the HTML parser knows that the tag <b> is bold. With XML there is much more flexibility. It is possible to define any XML tag to do anything you desire. The point is that it is possible to apply a Text state using XML. But the Text state that is applied must be part of the schema. For example in the OrgChart.xml, OrgChart.sps, OrgChart.xsd example the tag is the same as bold. And to apply bold the method ApplyTextState() is called. But like the row and edit operations it is necessary to test if it is possible to apply the text state.

See also IsTextStateEnabled[1837] and IsTextStateApplied[1836].

29.3.3.3.2 CurrentSelection (obsolete)

Superseded by AuthenticView.Selection[1650]

The returned AuthenticRange[1606] object supports navigation via XMLData elements as well as navigation by document elements (e.g. characters, words, tags) or text cursor positions.

```
// ----- javascript sample -----
// instead of:
// var objDocEditSel = Application.ActiveDocument.DocEditView.CurrentSelection;
// use now:
    var objRange = Application.ActiveDocument.AuthenticView.Selection;
```

Declaration: CurrentSelection as DocEditSelection[1819]

Description
The property provides access to the current selection in the Authentic View.

29.3.3.3.3 EditClear (obsolete)

Superseded by AuthenticRange.Delete 1610

The Delete method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditClear();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Delete())
    MsgBox ("Error: can't delete current selection");
```

Declaration: EditClear

Description
Deletes the current selection.

29.3.3.3.4 EditCopy (obsolete)

Superseded by AuthenticRange.Copy 1610

The Copy method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditCopy();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Copy())
    MsgBox ("Error: can't copy current selection");
```

Declaration: EditCopy

Description
Copies the current selection to the clipboard.

(C) 2015-2021 Altova GmbH

29.3.3.3.5 EditCut (obsolete)

Superseded by AuthenticRange.Cut 1610

The Cut method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditCut();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Cut())
    MsgBox ("Error: can't cut out current selection");
```

Declaration: EditCut

Description
Cuts the current selection from the document and copies it to the clipboard.

29.3.3.3.6 EditPaste (obsolete)

Superseded by AuthenticRange.Paste 1628

The Paste method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditPaste();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Paste())
    MsgBox ("Error: can't paste to current selection");
```

Declaration: EditPaste

Description
Pastes the content from the clipboard into the document.

29.3.3.3.7 EditRedo (obsolete)

Superseded by AuthenticView.Redo 1650

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditRedo();
// use now:
if (! Application.ActiveDocument.AuthenticView.Redo())
    MsgBox ("Error: no redo step available");
```

Declaration: EditRedo

Description
Redo the last undo step.

29.3.3.3.8 EditSelectAll (obsolete)

Superseded by AuthenticView.WholeDocument 1652 and **AuthenticRange.Select** 1630

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditSelectAll();
// use now:
Application.ActiveDocument.AuthenticView.WholeDocument.Select();
```

Declaration: EditSelectAll

Description
The method selects the complete document.

29.3.3.3.9 EditUndo (obsolete)

Superseded by AuthenticView.Undo 1651

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditUndo();
// use now:
```


(C) 2015-2021 Altova GmbH

```
if (! Application.ActiveDocument.AuthenticView.Undo())
    MsgBox ("Error: no undo step available");
```

Declaration: EditUndo

Description
Undo the last action.

29.3.3.3.10 event (obsolete)

Superseded by parameters to AuthenticView events[1634].

Declaration: event as DocEditEvent[1806]

Description
The event property holds a DocEditEvent object which contains information about the current event.

29.3.3.3.11 GetAllowedElements (obsolete)

Superseded by AuthenticRange.CanPerformActionWith[1609]

AuthenticRange now supports all functionality of the 'elements' entry helper. Besides querying the elements that can be inserted, appended, etc., you can invoke the action as well. See AuthenticRange.PerformAction[1629] for more information.

```
// ----- javascript sample -----
// instead of:
// var arrElements = New Array();
// var objDocEditView = Application.ActiveDocument.DocEditView;
// var objStartElement = objDocEditView.CurrentSelection.Start;
// var objEndElement = objDocEditView.CurrentSelection.End;
// objDocEditView.GetAllowedElements(k_ActionInsertBefore, objStartElement, objEndElement,
arrElements);
// use now:
var arrElements = New Array();
Application.ActiveDocument.AuthenticView.Selection.CanPerformActionWith (spyAuthenticInsertBefore,
arrElements);
```

Declaration: GetAllowedElements (*nAction* as SpyAuthenticElementActions[1845], *pStartElement* as XMLData[1795], *pEndElement* as XMLData[1795], *pElements* as Variant)

Description
GetAllowedElements() returns the allowed elements for the various actions specified by nAction.

JavaScript example:

```
Function GetAllowed()
{
	var objView = Application.ActiveDocument.DocEditView;

	var arrElements = New Array(1);

	var objStart = objView.CurrentSelection.Start;
	var objEnd = objView.CurrentSelection.End;

	var strText;
	strText = "valid elements at current selection:\n\n";

	For(var i = 1;i <= 4;i++) {
		objPlugIn.GetAllowedElements(i,objStart,objEnd,arrElements);
		strText = strText + ListArray(arrElements) + "----------\n";
	}

	Return strText;
}

Function ListArray(arrIn)
{
	var strText = "";

	If(TypeOf(arrIn) == "object")	{
		For(var i = 0;i <= (arrIn.length - 1);i++)
			strText = strText + arrIn[i] + "\n";
	}

	Return strText;
}
```

VBScript example:

```
Sub DisplayAllowed
	Dim objView
	Set objView = Application.ActiveDocument.DocEditView

	Dim arrElements()

	Dim objStart
	Dim objEnd
	Set objStart = objView.CurrentSelection.Start
	Set objEnd = objView.CurrentSelection.End

	Dim strText
	strText = "valid elements at current selection:" & chr(13) & chr(13)
```

```
        Dim i

        For i = 1 To 4
            objView.GetAllowedElements i,objStart,objEnd,arrElements
            strText = strText & ListArray(arrElements) & "----------" & chr(13)
        Next

        msgbox strText
End Sub

Function ListArray(arrIn)
        Dim strText

        If IsArray(arrIn) Then
            Dim i

            For i = 0 To UBound(arrIn)
                strText = strText & arrIn(i) & chr(13)
            Next
        End If

        ListArray = strText
End Function
```

29.3.3.3.12 GetNextVisible (obsolete)

Superseded by AuthenticRange.SelectNext 1630

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the text passage that represents the content of the next XML element is just one of them.

```
// ----- javascript sample -----
// instead of:
// var objCurrXMLData = ...
// var objXMLData = Application.ActiveDocument.DocEditView.GetNextVisible(objCurrXMLData);
// Application.ActiveDocument.DocEditView.SelectionSet (objXMLData, 0, objXMLData, -1);
// use now:
var objRange = ...
try
   { objRange.SelectNext (spyAuthenticTag).Select(); }
catch (err)
{
   if ((err.number & 0xffff) == 2003)
      MsgBox ("end of document reached");
   else
      throw (err);
}
```

Declaration: GetNextVisible (*pElement* as XMLData[1795]) as XMLData[1795]

Description
The method gets the next visible XML element in the document.

29.3.3.3.13 GetPreviousVisible (obsolete)

Superseded by AuthenticRange.SelectPrevious[1631]

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the text passage that represents the content of the previous XML element is just one of them.

```
// ----- javascript sample -----
// instead of:
// var objCurrXMLData = ...
// var objXMLData = Application.ActiveDocument.DocEditView.GetPreviousVisible(objCurrXMLData);
// Application.ActiveDocument.DocEditView.SelectionSet (objXMLData, 0, objXMLData, -1);
// use now:
var objRange = ...
try
   { objRange.SelectPrevious (spyAuthenticTag).Select(); }
catch (err)
{
   if ((err.number & 0xffff) == 2004)
      MsgBox ("begin of document reached");
   else
      throw (err);
}
```

Declaration: GetPreviousVisible (*pElement* as XMLData[1795]) as XMLData[1795]

Description
The method gets the previous visible XML element in the document.

29.3.3.3.14 IsEditClearEnabled (obsolete)

Superseded by AuthenticRange.IsDeleteEnabled[1622]

The IsDeleteEnabled property is now supported for any range of the document, not only the current UI selection.

(C) 2015-2021 Altova GmbH

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditClearEnabled)
//    Application.ActiveDocument.DocEditView.EditClear();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsDeleteEnabled)
   objCurrSelection.Delete();
```

Declaration: IsEditClearEnabled as Boolean

Description
True if EditClear[1824] is possible. See also Editing operations[1578].

29.3.3.3.15 IsEditCopyEnabled (obsolete)

Superseded by AuthenticRange.IsCopyEnabled[1622]

The IsCopyEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditCopyEnabled)
//    Application.ActiveDocument.DocEditView.EditCopy();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsCopyEnabled)
   objCurrSelection.Copy();
```

Declaration: IsEditCopyEnabled as Boolean

Description
True if copy to clipboard is possible. See also EditCopy[1824] and Editing operations[1578].

29.3.3.3.16 IsEditCutEnabled (obsolete)

Superseded by AuthenticRange.IsCutEnabled[1622]

The IsCutEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditCutEnabled)
//    Application.ActiveDocument.DocEditView.EditCut();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsCutEnabled)
   objCurrSelection.Cut();
```

Declaration: IsEditCutEnabled as Boolean

Description
True if EditCut[1825] is currently possible. See also Editing operations[1578].

29.3.3.3.17 IsEditPasteEnabled (obsolete)

Superseded by AuthenticRange.IsPasteEnabled[1624]

The IsPasteEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditPasteEnabled)
//    Application.ActiveDocument.DocEditView.EditPaste();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsPasteEnabled)
   objCurrSelection.Paste();
```

Declaration: IsEditPasteEnabled as Boolean

Description
True if EditPaste[1825] is possible. See also Editing operations[1578].

29.3.3.3.18 IsEditRedoEnabled (obsolete)

Superseded by AuthenticView.IsRedoEnabled[1648]

(C) 2015-2021 Altova GmbH

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditRedoEnabled)
//    Application.ActiveDocument.DocEditView.EditRedo();
// use now:
if (Application.ActiveDocument.AuthenticView.IsRedoEnabled)
   Application.ActiveDocument.AuthenticView.Redo();
```

Declaration: IsEditRedoEnabled as Boolean

Description
True if EditRedo[1826] is currently possible. See also Editing operations[1578].

29.3.3.3.19 IsEditUndoEnabled (obsolete)

Superseded by AuthenticView.IsUndoEnabled[1648]

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditUndoEnabled)
//    Application.ActiveDocument.DocEditView.EditUndo();
// use now:
if (Application.ActiveDocument.AuthenticView.IsUndoEnabled)
   Application.ActiveDocument.AuthenticView.Undo();
```

Declaration: IsEditUndoEnabled as Boolean

Description
True if EditUndo[1826] is possible. See also Editing operations[1578].

29.3.3.3.20 IsRowAppendEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable[1623]

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
```

```
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowAppendEnabled)
//    Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.AppendRow();
```

Declaration: IsRowAppendEnabled as Boolean

Description
True if RowAppend[1838] is possible. See also Row operations[1577].

29.3.3.3.21 IsRowDeleteEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable[1623]

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowDeleteEnabled)
//    Application.ActiveDocument.DocEditView.Rowdelete();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.DeleteRow();
```

Declaration: IsRowDeleteEnabled as Boolean

Description
True if RowDelete[1839] is possible. See also Row operations[1577].

29.3.3.3.22 IsRowDuplicateEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable[1623]

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
```


(C) 2015-2021 Altova GmbH

```
// if (Application.ActiveDocument.DocEditView.IsRowDuplicateEnabled)
//    Application.ActiveDocument.DocEditView.RowDuplicate();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
   Application.ActiveDocument.AuthenticView.Selection.DuplicateRow();
```

Declaration: IsRowDuplicateEnabled as Boolean

Description
True if RowDuplicate[1839] is currently possible. See also Row operations[1577].

29.3.3.3.23 IsRowInsertEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable[1623]

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowInsertEnabled)
//    Application.ActiveDocument.DocEditView.RowInsert();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
   Application.ActiveDocument.AuthenticView.Selection.InsertRow();
```

Declaration: IsRowInsertEnabled as Boolean

Description
True if RowInsert[1840] is possible. See also Row operations[1577].

29.3.3.3.24 IsRowMoveDownEnabled (obsolete)

Superseded by AuthenticRange.IsLastRow[1623]

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.OldAuthenticView.IsRowMoveDownEnabled)
//    Application.ActiveDocument.DocEditView.RowMoveDown();
// use now:
```

```
if (!Application.ActiveDocument.AuthenticView.Selection.IsLastRow)
    Application.ActiveDocument.AuthenticView.Selection.MoveRowDown();
```

Declaration: IsRowMoveDownEnabled as Boolean

Description
True if RowMoveDown[1840] is currently possible. See also Row operations[1577].

29.3.3.3.25 IsRowMoveUpEnabled (obsolete)

Superseded by AuthenticRange.IsFirstRow[1623]

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowMoveUpEnabled)
//    Application.ActiveDocument.DocEditView.RowMoveUp();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.IsFirstRow)
    Application.ActiveDocument.AuthenticView.Selection.MoveRowUp();
```

Declaration: IsRowMoveUpEnabled as Boolean

Description
True if RowMoveUp[1841] is possible. See also Row operations[1577].

29.3.3.3.26 IsTextStateApplied (obsolete)

Superseded by AuthenticRange.IsTextStateApplied[1624]

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.IsTextStateApplied ("bold");
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsTextStateApplied ("bold"))
    MsgBox ("bold on");
else
    MsgBox ("bold off");
```

Declaration: IsTextStateApplied (*elementName* as String) as Boolean

(C) 2015-2021 Altova GmbH

Description
Checks to see if the it the text state has already been applied. Common examples for the parameter elementName would be strong and italic.

29.3.3.3.27 IsTextStateEnabled (obsolete)

Superseded by AuthenticRange.CanPerformAction 1608

Use spyAuthenticApply for the eAction parameter. The CanPerformAction method allows to operate on any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.IsTextStateEnabled ("bold");
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.CanPerformAction (spyAuthenticApply, "bold"))
    ... // e.g. enable 'bold' button
```

Declaration: IsTextStateEnabled (*i_strElementName* as String) as Boolean

Description
Checks to see if it is possible to apply a text state. Common examples for the parameter elementName would be strong and italic.

29.3.3.3.28 LoadXML (obsolete)

Superseded by AuthenticView.AsXMLString 1644

AuthenticView now supports the property AsXMLString that can be used to directly access and replace the document content as an XMLString.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.LoadXML (strDocAsXMLString);
// use now:
try
    { Application.ActiveDocument.AuthenticView.AsXMLString = strDocAsXMLString; }
catch (err)
    { MsgBox ("Error: invalid XML string"); }
```

Declaration: LoadXML (*xmlString* as String)

Description
Loads the current XML document with the XML string applied. The new content is displayed immediately. The *xmlString* parameter must begin with the XML declaration, e.g.,
objPlugIn.LoadXML("<?xml version='1.0' encoding='UTF-8'?><root></root>");

29.3.3.3.29 MarkUpView (obsolete)

Superseded by AuthenticView.MarkupVisibility 1649

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.MarkuUpView = 2;
// use now:
Application.ActiveDocument.AuthenticView.MarkupVisibility = spyAuthenticMarkupLarge;
```

Declaration: MarkUpView (*kind* as long)

Description
By default the document displayed is using HTML techniques. But sometimes it is desirable to show the editing tags. Using this method it is possible to display three different types of markup tags:

0	hide the markup tags
2	show the large markup tags
3	show the mixed markup tags.

29.3.3.3.30 RowAppend (obsolete)

Superseded by AuthenticRange.AppendRow 1607

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.AppendRow())
    MsgBox ("Error: can't append row");
```

Declaration: RowAppend

Description
Appends a row at the current position.

See also Row operations[1577].

29.3.3.3.31 RowDelete (obsolete)

Superseded by AuthenticRange.DeleteRow[1611]

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowDelete();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.DeleteRow())
    MsgBox ("Error: can't delete row");
```

Declaration: RowDelete

Description
Deletes the currently selected row(s).

See also Row operations[1577].

29.3.3.3.32 RowDuplicate (obsolete)

Superseded by AuthenticRange.DuplicateRow[1611]

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowDuplicate();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.DuplicateRow())
    MsgBox ("Error: can't duplicate row");
```

Declaration: RowDuplicate

Description
The method duplicates the currently selected rows.

See also Row operations 1577.

29.3.3.3.33 RowInsert (obsolete)

Superseded by AuthenticRange.InsertRow 1621

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowInsert();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.InsertRow())
    MsgBox ("Error: can't insert row");
```

Declaration: RowInsert

Description
Inserts a new row immediately above the current selection.

See also Row operations 1577.

29.3.3.3.34 RowMoveDown (obsolete)

Superseded by AuthenticRange.MoveRowDown 1628

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowMoveDown();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.MoveRowDown())
    MsgBox ("Error: can't move row down");
```

Declaration: RowMoveDown

(C) 2015-2021 Altova GmbH

Description
Moves the current row one position down.

See also Row operations[1577].

29.3.3.3.35 RowMoveUp (obsolete)

Superseded by AuthenticRange.MoveRowUp[1628]

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.MoveRowUp())
    MsgBox ("Error: can't move row up");
```

Declaration: RowMoveUp

Description
Moves the current row one position up.

See also Row operations[1577].

29.3.3.3.36 SaveXML (obsolete)

Superseded by AuthenticView.AsXMLString[1644]

AuthenticView now supports the property XMLString that can be used to directly access and replace the document content as an XMLString.

```
// ----- javascript sample -----
// instead of:
// var strDocAsXMLString = Application.ActiveDocument.DocEditView.SaveXML();
// use now:
try
{
    var strDocAsXMLString = Application.ActiveDocument.AuthenticView.AsXMLString;
    ... // do something here
}
```

```
catch (err)
   { MsgBox ("Error: invalid XML string"); }
```

Declaration: SaveXML as String

Return Value
XML structure as string

Description
Saves the current XML data to a string that is returned to the caller.

29.3.3.3.37 SelectionMoveTabOrder (obsolete)

Superseded by AuthenticRange.SelectNext 1630

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the next paragraph is just one of them, and navigation is not necessarily bound to the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.SelectionMoveTabOrder(true, true);
// use now:
Application.ActiveDocument.AuthenticView.Selection.SelectNext (spyAuthenticParagraph).Select();
// to append a row to a table use AuthenticRange.AppendRow
```

Declaration: SelectionMoveTabOrder (*bForward* as Boolean, *bTag* as Boolean)

Description
SelectionMoveTabOrder() moves the current selection forwards or backwards.

If bTag is false and the current selection is at the last cell of a table a new line will be added.

29.3.3.3.38 SelectionSet (obsolete)

Superseded by AuthenticRange.FirstXMLData 1613 and related properties

AuthenticRange supports navigation via XMLData elements as well as navigation by document elements (e.g. characters, words, tags) or text cursor positions.

```
// ----- javascript sample -----
// instead of:
```

```
// if (! Application.ActiveDocument.DocEditView.SelectionSet(varXMLData1, 0, varXMLData2, -1))
//    MsgBox ("Error: invalid data position");
// use now:
try
{
    var objSelection = Application.ActiveDocument.AuthenticView.Selection;
    objSelection.FirstXMLData = varXMLData1;
    objSelection.FirstXMLdataOffset = 0;
    objSelection.LastXMLData = varXMLData2;
    objSelection.LastXMLDataOffset = -1;
    objSelection.Select();
}
catch (err)
    { MsgBox ("Error: invalid data position"); }
// to select all text between varXMLData1 and varXMLdata2, inclusive
```

Declaration: SelectionSet (*pStartElement* as XMLData[1795], *nStartPos* as long, *pEndElement* as XMLData[1795], *nEndPos* as long) as Boolean

Description
Use SelectionSet() to set a new selection in the Authentic View. Its possible to set pEndElement to null (nothing) if the selection should be just over one (pStartElement) XML element.

29.3.3.3.39 XMLRoot (obsolete)

Superseded by AuthenticView.XMLDataRoot[1652]

```
// ----- javascript sample -----
// instead of:
// var objXMLData = Application.ActiveDocument.DocEditView.XMLRoot;
// use now:
var objXMLData = Application.ActiveDocument.AuthenticView.XMLDataRoot;
```

Declaration: XMLRoot as XMLData[1795]

Description
XMLRoot is the parent element of the currently displayed XML structure. Using the XMLData[1795] interface you have full access to the complete content of the file.

29.3.4 Enumerations

This is a list of all enumerations used by the XMLSpy API. If your scripting environment does not support enumerations use the number-values instead.

29.3.4.1 ENUMApplicationStatus

Description
Enumeration to specify the current Application status.

Possible values:

eApplicationRunning	= 0
eApplicationAfterLicenseCheck	= 1
eApplicationBeforeLicenseCheck	= 2
eApplicationConcurrentLicenseCheckFailed	= 3
eApplicationProcessingCommandLine	= 4

29.3.4.2 SPYAttributeTypeDefinition

Description
Attribute type definition that can be selected for generation of Sample XML.
This type is used with the method GenerateDTDOrSchema[1683] and GenerateDTDOrSchemaEx[1683].

Possible values:

spyMergedGlobal	= 0
spyDistinctGlobal	= 1
spyLocal	= 2

29.3.4.3 SPYAuthenticActions

Description
Actions that can be performed on AuthenticRange[1606] objects.

Possible values:

spyAuthenticInsertAt	= 0
spyAuthenticApply	= 1
spyAuthenticClearSurr	= 2
spyAuthenticAppend	= 3
spyAuthenticInsertBefore	= 4
spyAuthenticRemove	= 5

29.3.4.4 SPYAuthenticDocumentPosition

Description
Relative and absolute positions used for navigating with AuthenticRange[1606] objects.

Possible values:

spyAuthenticDocumentBegin = 0
spyAuthenticDocumentEnd = 1
spyAuthenticRangeBegin = 2
spyAuthenticRangeEnd = 3

29.3.4.5 SPYAuthenticElementActions

Description
Actions that can be used with the obsolete object GetAllowedElements (superseded by AuthenticRange.CanPerformActionWith[1609]).

Possible values:
k_ActionInsertAt = 0
k_ActionApply = 1
k_ActionClearSurr = 2
k_ActionAppend = 3
k_ActionInsertBefore = 4
k_ActionRemove = 5

29.3.4.6 SPYAuthenticElementKind

Description
Enumeration of the different kinds of elements used for navigation and selection within the AuthenticRange[1606] and AuthenticView[1634] objects.

Possible values:
spyAuthenticChar = 0
spyAuthenticWord = 1
spyAuthenticLine = 3
spyAuthenticParagraph = 4
spyAuthenticTag = 6
spyAuthenticDocument = 8
spyAuthenticTable = 9
spyAuthenticTableRow = 10
spyAuthenticTableColumn = 11

29.3.4.7 SPYAuthenticMarkupVisibility

Description
Enumeration values to customize the visibility of markup with MarkupVisibility[1649].

Possible values:
spyAuthenticMarkupHidden = 0
spyAuthenticMarkupSmall = 1
spyAuthenticMarkupLarge = 2

spyAuthenticMarkupMixed = 3

29.3.4.8 SPYAuthenticToolbarButtonState

Description
Authentic toolbar button states are given by the following enumeration:

Possible values:

authenticToolbarButtonDefault	= 0
authenticToolbarButtonEnabled	= 1
authenticToolbarButtonDisabled	= 2

29.3.4.9 SPYDatabaseKind

Description
Values to select different kinds of databases for import. See DatabaseConnection.DatabaseKind[1660] for its use.

Possible values:

spyDB_Access	= 0
spyDB_SQLServer	= 1
spyDB_Oracle	= 2
spyDB_Sybase	= 3
spyDB_MySQL	= 4
spyDB_DB2	= 5
spyDB_Other	= 6
spyDB_Unspecified	= 7
spyDB_PostgreSQL	= 8
spyDB_iSeries	= 9

29.3.4.10 SPYDialogAction

Description
Values to simulate different interactions on dialogs. See Dialogs[1664] for all dialogs available.

Possible values:

spyDialogOK	= 0	// simulate click on OK button
spyDialogCancel	= 1	// simulate click on Cancel button
spyDialogUserInput	= 2	// show dialog and allow user interaction

(C) 2015-2021 Altova GmbH

29.3.4.11 SPYDOMType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:

spyDOMType_msxml4	= 0	Obsolete
spyDOMType_xerces	= 1	
spyDOMType_xerces3	= 2	
spyDOMType_msxml6	= 3	

`spyDOMType_xerces` indicates Xerces 2.x usage; `spyDOMType_xerces3` indicates Xerces 3.x usage.

29.3.4.12 SPYDTDSchemaFormat

Description
Enumeration to identify the different schema formats.

Possible values:

spyDTD	= 0
spyW3C	= 1

29.3.4.13 SPYEncodingByteOrder

Description
Enumeration values to specify encoding byte ordering for text import and export.

Possible values:

spyNONE	= 0
spyLITTLE_ENDIAN	= 1
spyBIG_ENDIAN	= 2

29.3.4.14 SPYExportNamespace

Description
Enumeration type to configure handling of namespace identifiers during export.

Possible values:

spyNoNamespace	= 0
spyReplaceColonWithUnderscore	= 1

29.3.4.15 SPYFindInFilesSearchLocation

Description
The different locations where a search can be performed. This type is used with the FindInFilesDlg[1712] dialog.

Possible values:
spyFindInFiles_Documents = 0
spyFindInFiles_Project = 1
spyFindInFiles_Folder = 2

29.3.4.16 SPYFrequentElements

Description
Enumeration value to parameterize schema generation.

Possible values:
spyGlobalElements = 0
spyGlobalComplexType = 1

29.3.4.17 SPYImageKind

Description
Enumeration values to parameterize image type of the generated documentation. These values are used in SchemaDocumentationDialog.DiagramFormat[1735] and WSDLDocumentationDlg.DiagramFormat[1763].

Possible values:
spyImageType_PNG = 0
spyImageType_EMF = 1

29.3.4.18 SPYImportColumnsType

Description
Enumeration to specify different Import columns types.

Possible values:

spyImportColumns_Element = 0
spyImportColumns_Attribute = 1

(C) 2015-2021 Altova GmbH

29.3.4.19 SPYKeyEvent

Description
Enumeration type to identify the different key events. These events correspond with the equally named windows messages.

Possible values:

spyKeyDown	= 0
spyKeyUp	= 1
spyKeyPressed	= 2

29.3.4.20 SPYKeyStatus

Description
Enumeration type to identify the key status.

Possible values:

spyLeftShiftKeyMask	= 1
spyRightShiftKeyMask	= 2
spyLeftCtrlKeyMask	= 4
spyRightCtrlKeyMask	= 8
spyLeftAltKeyMask	= 16
spyRightAltKeyMask	= 32

29.3.4.21 SPYLibType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:

spyLibType_static	= 0
spyLibType_dll	= 1

29.3.4.22 SPYLoading

Description
Enumeration values to define loading behaviour of URL files.

Possible values:

spyUseCacheProxy	= 0
spyReload	= 1

29.3.4.23 SPYMouseEvent

Description
Enumeration type that defines the mouse status during a mouse event. Use the enumeration values as bitmasks rather then directly comparing with them.

Examples

```
' to check for ctrl-leftbutton-down in VB
If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or XMLSpyLib.spyCtrlKeyDownMask)) Then
        ' react on ctrl-leftbutton-down
End If

' to check for double-click with any button in VBScript
If (((i_eMouseEvent And spyDoubleClickMask) <> 0) Then
        ' react on double-click
End If
```

Possible values:

```
spyNoButtonMask          = 0
spyMouseMoveMask         = 1
spyLeftButtonMask        = 2
spyMiddleButtonMask      = 4
spyRightButtonMask       = 8
spyButtonUpMask          = 16
spyButtonDownMask        = 32
spyDoubleClickMask       = 64
spyShiftKeyDownMask      = 128
spyCtrlKeyDownMask       = 256
spyLeftButtonDownMask    = 34   // spyLeftButtonMask | spyButtonDownMask
spyMiddleButtonDownMask  = 36   // spyMiddleButtonMask | spyButtonDownMask
spyRightButtonDownMask   = 40   // spyRightButtonMask | spyButtonDownMask
spyLeftButtonUpMask      = 18   // spyLeftButtonMask | spyButtonUpMask
spyMiddleButtonUpMask    = 20   // spyMiddleButtonMask | spyButtonUpMask
spyRightButtonUpMask     = 24   // spyRightButtonMask | spyButtonUpMask
spyLeftDoubleClickMask   = 66   // spyRightButtonMask | spyButtonUpMask
spyMiddleDoubleClickMask = 68   // spyMiddleButtonMask | spyDoubleClickMask
spyRightDoubleClickMask  = 72   // spyRightButtonMask | spyDoubleClickMask
```

29.3.4.24 SPYNumberDateTimeFormat

Description
Enumeration value to configure database connections.

Possible values:

```
spySystemLocale      = 0
spySchemaCompatible  = 1
```


(C) 2015-2021 Altova GmbH

29.3.4.25 SPYProgrammingLanguage

Description
Enumeration values to select the programming language for code generation from schema definitions.

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

Possible values:

spyUndefinedLanguage	= -1
spyJava	= 0
spyCpp	= 1
spyCSharp	= 2

29.3.4.26 SPYProjectItemTypes

Description
Enumeration values to identify the different elements in project item lists. See SpyProjectItem.ItemType[1749].

Possible values:

spyUnknownItem	= 0
spyFileItem	= 1
spyFolderItem	= 2
spyURLItem	= 3

29.3.4.27 SPYProjectType

Description
Enumeration values to parameterize generation of C# from schema definitions.

Possible values:

spyVisualStudioProject	= 0	Obsolete
spyVisualStudio2003Project	= 1	Obsolete
spyBorlandProject	= 2	Obsolete
spyMonoMakefile	= 3	Obsolete
spyVisualStudio2005Project	= 4	Obsolete
spyVisualStudio2008Project	= 5	Obsolete
spyVisualStudio2010Project	= 6	For C++ code also
spyVisualStudio2013Project	= 7	For C++ code also
spyVisualStudio2015Project	= 8	For C++ code also
spyVisualStudio2017Project	= 9	For C++ code also
spyVisualStudio2019Project	=10	For C++ code also

29.3.4.28 SpySampleXMLGenerationChoiceMode

Description
This enumeration is used in GenerateSampleXMLDlg.ChoiceMode[1725]:

spySampleXMLGen_FirstBranch = 0
spySampleXMLGen_AllBranches = 1
spySampleXMLGen_ShortestBranch = 2

29.3.4.29 SPYSampleXMLGenerationOptimization (Obsolete)

This enumeration is OBSOLETE since v2014.

Description
Specify the elements that will be generated in the Sample XML.
This enumeration is used in GenerateSampleXMLDlg[1724].

Possible values:

spySampleXMLGen_Optimized	= 0
spySampleXMLGen_NonMandatoryElements	= 1
spySampleXMLGen_Everything	= 2

29.3.4.30 SpySampleXMLGenerationSampleValueHints

Description
This enumeration is used in GenerateSampleXMLDlg.SampleValueHints[1729]

spySampleXMLGen_FirstFit = 0
spySampleXMLGen_RandomFit = 1
spySampleXMLGen_CycleThrough = 2

29.3.4.31 SPYSampleXMLGenerationSchemaOrDTDAssignment

Description
Specifies what kind of reference to the schema/DTD should be added to the generated Sample XML.
This enumeration is used in GenerateSampleXMLDlg[1724].

(C) 2015-2021 Altova GmbH

Possible values:

spySampleXMLGen_AssignRelatively	= 0
spySampleXMLGen_AssignAbsolutely	= 1
spySampleXMLGen_DoNotAssign	= 2

29.3.4.32 SPYSchemaDefKind

Description
Enumeration type to select schema diagram types.

Possible values:

spyKindElement	= 0
spyKindComplexType	= 1
spyKindSimpleType	= 2
spyKindGroup	= 3
spyKindModel	= 4
spyKindAny	= 5
spyKindAttr	= 6
spyKindAttrGroup	= 7
spyKindAttrAny	= 8
spyKindIdentityUnique	= 9
spyKindIdentityKey	= 10
spyKindIdentityKeyRef	= 11
spyKindIdentitySelector	= 12
spyKindIdentityField	= 13
spyKindNotation	= 14
spyKindInclude	= 15
spyKindImport	= 16
spyKindRedefine	= 17
spyKindFacet	= 18
spyKindSchema	= 19
spyKindCount	= 20

29.3.4.33 SPYSchemaDocumentationFormat

Description
Enumeration values to parameterize generation of schema documentation. These values are used in SchemaDocumentationDialog.OutputFormat[1741] and WSDLDocumentationDlg.OutputFormat[1767].

Possible values:

spySchemaDoc_HTML	= 0
spySchemaDoc_MSWord	= 1
spySchemaDoc_RTF	= 2
spySchemaDoc_PDF	= 3

29.3.4.34 SPYSchemaExtensionType

Description
Enumeration to specify different Schema Extension types.

Possible values:

spySchemaExtension_None = 0
spySchemaExtension_SQL_XML = 1
spySchemaExtension_MS_SQL_Server = 2
spySchemaExtension_Oracle = 3

29.3.4.35 SPYSchemaFormat

Description
Enumeration to specify different Schema Format types.

Possible values:

spySchemaFormat_Hierarchical = 0
spySchemaFormat_Flat = 1

29.3.4.36 SPYTextDelimiters

Description
Enumeration values to specify text delimiters for text export.

Possible values:
spyTabulator = 0
spySemicolon = 1
spyComma = 2
spySpace = 3

29.3.4.37 SPYTextEnclosing

Description
Enumeration value to specify text enclosing characters for text import and export.

Possible values:
spyNoEnclosing = 0
spySingleQuote = 1
spyDoubleQuote = 2

(C) 2015-2021 Altova GmbH

29.3.4.38 SPYTypeDetection

Description
Enumeration to select how type detection works during GenerateDTDOrSchema[1683] and GenerateDTDOrSchemaEx[1683].

Possible values:
spyBestPossible = 0
spyNumbersOnly = 1
spyNoDetection = 2

29.3.4.39 SPYURLTypes

Description
Enumeration to specify different URL types.

Possible values:
spyURLTypeAuto = -1
spyURLTypeXML = 0
spyURLTypeDTD = 1

29.3.4.40 SPYValidateXSDVersion

Description
Enumeration values that select what XSD version to use. The XSD version that is selected depends on both (i) the presence/absence—and, if present, the value—of the `/xs:schema/@vc:minVersion` attribute of the XSD document, and (ii) the value of this enumeration.

`spyValidateXSDVersion_1_0` selects `XSD 1.0` if `vc:minVersion` is absent, or is present with any value.
`spyValidateXSDVersion_1_1` selects `XSD 1.1` if `vc:minVersion` is absent, or is present with any value.
`spyValidateXSDVersion_AutoDetect` selects `XSD 1.1` if `vc:minVersion=1.1`. If the `vc:minVersion` attribute is absent, or is present with a value other than `1.1`, then `XSD 1.0` is selected.

Possible values

```
spyValidateXSDVersion_AutoDetect      = 0
spyValidateXSDVersion_1_1             = 1
spyValidateXSDVersion_1_0             = 2
```

29.3.4.41 SPYValidateErrorFormat

Description
Enumeration values that select the format of the error message.

Possible values

```
spyValidateErrorFormat_Text          = 0
spyValidateErrorFormat_ShortXML      = 1
spyValidateErrorFormat_LongXML       = 2
```

29.3.4.42 SPYViewModes

Description
Enumeration values that define the different view modes for XML documents. The mode *spyViewAuthentic(4)* identifies the mode that was intermediately called DocEdit mode and is now called Authentic mode. The mode *spyViewJsonSchema* identifies a mode which is mapped to the Schema Design View on the GUI but is distinguished internally.

Possible values:

```
spyViewGrid          = 0
spyViewText          = 1
spyViewBrowser       = 2
spyViewSchema        = 3
spyViewContent       = 4     // obsolete
spyViewAuthentic     = 4
spyViewWSDL          = 5
spyViewZIP           = 6
spyViewEditionInfo   = 7
spyViewXBRL          = 8
spyViewJsonSchema    = 9
```

29.3.4.43 SPYVirtualKeyMask

Description
Enumeration type for the most frequently used key masks that identify the status of the virtual keys. Use these values as bitmasks rather then directly comparing with them. When necessary, you can create further masks by using the 'logical or' operator.

Examples

```
'  VBScript sample: check if ctrl-key is pressed
If ((i_nVirtualKeyStatus And spyCtrlKeyMask) <> 0)) Then
        ' ctrl-key is pressed
End If

'  VBScript sample: check if ONLY ctrl-key is pressed
If (i_nVirtualKeyStatus == spyCtrlKeyMask) Then
        ' exactly ctrl-key is pressed
End If

// JScript sample: check if any of the right virtual keys is pressed
if ((i_nVirtualKeyStatus & (spyRightShiftKeyMask | spyRightCtrlKeyMask | spyRightAltKeyMask)) != 0)
```


(C) 2015-2021 Altova GmbH

```
{
        ; ' right virtual key is pressed
}
```

Possible values:

```
spyNoVirtualKeyMask     = 0
spyLeftShiftKeyMask     = 1
spyRightShiftKeyMask    = 2
spyLeftCtrlKeyMask      = 4
spyRightCtrlKeyMask     = 8
spyLeftAltKeyMask       = 16
spyRightAltKeyMask      = 32
spyShiftKeyMask         = 3     // spyLeftShiftKeyMask | spyRightShiftKeyMask
spyCtrlKeyMask          = 12    // spyLeftCtrlKeyMask | spyRightCtrlKeyMask
spyAltKeyMask           = 48    // spyLeftAltKeyMask | spyRightAltKeyMask
```

29.3.4.44 SPYXMLDataKind

Description
The different types of XMLData elements available for XML documents.

Possible values:

```
spyXMLDataXMLDocStruct          = 0
spyXMLDataXMLEntityDocStruct    = 1
spyXMLDataDTDDocStruct          = 2
spyXMLDataXML                   = 3
spyXMLDataElement               = 4
spyXMLDataAttr                  = 5
spyXMLDataText                  = 6
spyXMLDataCData                 = 7
spyXMLDataComment               = 8
spyXMLDataPI                    = 9
spyXMLDataDefDoctype            = 10
spyXMLDataDefExternalID         = 11
spyXMLDataDefElement            = 12
spyXMLDataDefAttlist            = 13
spyXMLDataDefEntity             = 14
spyXMLDataDefNotation           = 15
spyXMLDataKindsCount            = 16
```

29.3.5 Application API for Java (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

The Application API in Java has an interface built up of Java classes, each of which corresponds to an object in the Application API[1544]. Developers can use these Java classes to interact with the COM API. These classes are listed below and described in subsequent sections. For a description of the Application API[1544] objects themselves, see the Application API documentation[1544]. Bear in mind that some API features are only available in scripting environments; these have therefore not been ported to Java.

Java classes

- SpyApplication[1861]
  - SpyProject[1873]
    - SpyProjectItems[1873]
      - SpyProjectItem[1873]
  - SpyDocuments[1866]
    - SpyDoc[1864]
      - SpyAuthenticView[1886]
        - SpyAuthenticRange[1884]
      - SpyDocEditView[1887]
        - SpyDocEditSelection[1887]
      - SpyGridView[1872]
      - SpyTextView[1877]
      - SpyXMLData[1884]
  - SpyDialogs[1864]
    - SpyCodeGeneratorDlg[1862]
    - SpyDTDSchemaGeneratorDlg[1866]
    - SpyFileSelectionDlg[1869]
    - SpyFindInFilesDlg[1869]
    - SpyGenerateSampleXMLDlg[1871]
    - SpySchemaDocumentationDlg[1874]
    - SpyWSDL20DocumentationDlg[1877]
    - SpyWSDLDocumentationDlg[1879]
    - SpyXBRLDocumentationDlg[1881]
  - SpyDatabaseConnection[1862]
  - SpyElementList[1867]
  - SpyElementListItem[1868]
  - SpyExportSettings[1868]
  - SpyFindInFilesResults[1871]
    - SpyFindInFilesResult[1871]
      - SpyFindInFilesMatch[1870]
  - SpyTextImportExportSettings[1876]

Implementation of COM properties in Java

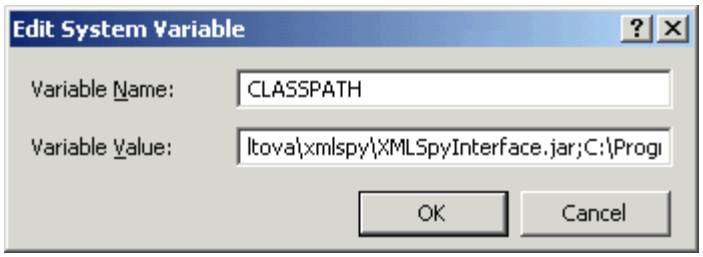
Properties in Java have been defined to include both a **set** and **get** method (`set` if it is allowed by the COM implementation). For example, the COM class `Document` contains the **GridView** property. In Java the method is called `SpyDoc` and the property is defined as a **GetGridView** method.

If you encounter compiling problems, please check the following points:

- The xmlspylib.dll must be available in ..¥windows¥system32.
- The XMLSpyInterface.jar file must be inserted in the ClassPath environment variable.

Setting the `ClassPath` variable in Windows XP

1. Click **Start | Settings | Control panel | System | Advanced | Environment Variables**. This opens the Environment Variables dialog box.
2. If a ClassPath entry already exists in the **System variables** group, select the ClassPath entry, and click the **Edit** button. Edit the path to: "C:¥Program Files¥Altova¥xmlspy¥XMLSpyInterface.jar".



If a ClassPath entry does not exist in the System variables group, click the **New** button. The New System Variable dialog pops up. Enter CLASSPATH as the variable name, and "C:¥Program Files¥Altova¥xmlspy¥XMLSpyInterface.jar" as the ClassPath variable (alter the path to match your installation, if necessary).

29.3.5.1 Sample source code (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1567.

The "`SpyDoc doc = app.GetDocuments().OpenFile(...)`" command parameter must be altered to suit your environment.

What the sample does:

- Starts a new XMLSpy instance
- Opens the Datasheet.xml file (alter the path here...)
- Switches to the Enhanced Grid view
- Appends a new child element called "NewChild" with the text value "NewValuE" element to the root element
- Checks if the document is valid and outputs a message to the Java console
- Quits and releases the XMLSpy application

```
import XMLSpyInterface.*;

public class TestSpyInterface
{
    public TestSpyInterface() {}

    public static void main(String[] args)
    {
        SpyApplication app = null;
        SpyDoc oDoc = null;
```

```
        SpyXMLData oData = null;
        SpyXMLData oNewChild = null;

        try
        {
            app = new SpyApplication();
            app.ShowApplication( true );

            oDoc = app.GetDocuments().OpenFile("C:¥¥FilePath¥¥OrgChart.xml", true );

            // OrgChart.xml is in the folder C:¥Documents and Settings¥<ユーザー名>¥マイドキュメント
            ¥Altova¥XMLSpy2021. The filepath should be in
            // the form: C:¥¥Documents and Settings¥¥Username¥¥Folder¥¥Filename.xml

            if ( oDoc != null )
            {
                oDoc.SwitchViewMode(SPYViewModes.spyViewGrid);
                oData = oDoc.GetRootElement();
                oNewChild = oDoc.CreateChild(SPYXMLDataKind.spyXMLDataElement);

                oNewChild.SetName( "NewChild" );
                oNewChild.SetTextValue("newVaLuE");
                oData.AppendChild(oNewChild);

                if ( oDoc.IsValid() == false )
                {
                    // is to be expected after above insertion
                    System.out.println( "!!!!!!validation error: " + oDoc.GetErrorString() );
                    System.out.println( "!!!!!!validation error: " + oDoc.GetErrorPos() );
                    System.out.println( "!!!!!!validation error: " + oDoc.GetBadData() );
                }
            }

            app.Quit();
        }
        finally
        {
            // Free any allocated resources by calling ReleaseInstance().
            if ( oNewChild != null )
                oNewChild.ReleaseInstance();

            if ( oData != null )
                oData.ReleaseInstance();

            if ( oDoc != null )
                oDoc.ReleaseInstance();

            if ( app != null )
                app.ReleaseInstance();
        }
    }
}
```

If you have difficulties compiling this sample, please try the following commands on the (**Start | Run | cmd**) command line. Please make sure you are currently in the folder that contains the sample java file.

compilation
javac -classpath c:*yourpathhere*\XMLSpyInterface.jar *testspyinterface.java*

Execution
java -classpath c:*yourpathhere*\XMLSpyInterface.jar *testspyinterface*

29.3.5.2 SpyApplication (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1567].

```
public class SpyApplication
{
   public void ReleaseInstance[1859]();
   public void ShowApplication[1597]( boolean bShow );
   public void Quit[1595]();
   public void AddMacroMenuItem[1583]( String sMacro, String sDisplayText );
   public void ClearMacroMenu[1584]();
   public SpyDoc[1864] GetActiveDocument[1583]();
   public SpyProject[1873] GetCurrentProject[1585]();
   public SpyDocuments[1866] GetDocuments[1585]();
   public SpyElementList[1867] GetDatabaseImportElementList[1586]( SpyDatabaseConnection[1862] oImportSettings );
   public SpyDatabaseConnection[1862] GetDatabaseSettings[1587]();
   public SpyElementList[1867] GetDatabaseTables[1587]( SpyDatabaseConnection[1862] oImportSettings );
   public SpyExportSettings[1868] GetExportSettings[1588]();
   public SpyElementList[1867] GetTextImportElementList[1588]( SpyTextImportExportSettings[1876] oImportSettings );
   public SpyTextImportExportSettings[1876] GetTextImportExportSettings[1589]();
   public SpyDoc[1864] ImportFromDatabase[1590]( SpyDatabaseConnection[1862] oImportSettings, SpyElementList[1867]
   oElementList );
   public SpyDoc[1864] ImportFromSchema[1591]( SpyDatabaseConnection[1862] oImportSettings, String strTable, SpyDoc[1864]
   oSchemaDoc );
   public SpyDoc[1864] ImportFromText[1592]( SpyTextImportExportSettings[1876] oImportSettings, SpyElementList[1867]
   oElementList );
   public SpyDoc[1864] ImportFromWord[1593]( String sFile );
   public void NewProject[1594]( String sPath, boolean bDiscardCurrent );
   public void OpenProject[1594](String sPath , boolean bDiscardCurrent, boolean bDialog );
   public long ShowForm[1597]( String sName );
   public void URLDelete[1593]( String sURL, String sUser, String sPassword );
   public void URLMakeDirectory[1598]( String sURL, String sUser, String sPassword );
   public int GetWarningNumber[1599]();
   public String GetWarningText[1599]();

   // since Version 2004R4
   public SpyApplication[1861] GetApplication[1584]();
   public SpyApplication[1861] GetParent[1594]();
   public SpyDialogs[1864] GetDialogs[1585]();
   public boolean GetVisible[1598]();
   public void SetVisible[1598]( boolean i_bVisibility );
   public long GetWindowHandle();

   public void ReloadSettings[1595]();
   public SpyFindInFilesResults[1871] FindInFiles[1586]( SpyFindInFilesDlg[1869] dlgSettings );
   public boolean ShowFindInFiles[1597]( SpyFindInFilesDlg[1869] dlgSettings );
   public void Selection( String sVal );

   public long[1888] Status[1598]();
   public int MajorVersion[1593]();
   public int MinorVersion[1593]();
   public String Edition[1586]();
   public boolean IsAPISupported[1593]();
```

```
   public long ServicePackVersion();
   public void CreateXMLSchemaFromDBStructure( SpyDatabaseConnection oConnection, SpyElementList
oTables );
}
```

29.3.5.3 SpyCodeGeneratorDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

Only available/enabled in the Enterprise edition. An error is returned, if accessed by any other version.

```
// since version 2004R4
public class SpyCodeGeneratorDlg
{
   public void ReleaseInstance();
   public SpyApplication GetApplication();
   public SpyDialogs GetParent();
   public long GetProgrammingLanguage();
   public void SetProgrammingLanguage( long i_eVal );
   public String GetTemplateFileName();
   public void SetTemplateFileName( String i_strVal );
   public String GetOutputPath();
   public void SetOutputPath( String i_strVal );
   public long GetOutputPathDialogAction();
   public void SetOutputPathDialogAction( long i_eVal );
   public long GetPropertySheetDialogAction();
   public void SetPropertySheetDialogAction( long i_eVal );
   public long GetOutputResultDialogAction();
   public void SetOutputResultDialogAction( long i_eVal );
   public long GetCPPSettings_DOMType();
   public void SetCPPSettings_DOMType( long i_eVal );
   public long GetCPPSettings_LibraryType();
   public void SetCPPSettings_LibraryType( long i_eVal );
   public boolean GetCPPSettings_UseMFC();
   public void SetCPPSettings_UseMFC( boolean i_bVal );
   public long GetCSharpSettings_ProjectType();
   public void SetCSharpSettings_ProjectType( long i_eVal );
}
```

29.3.5.4 SpyDatabaseConnection (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

(C) 2015-2021 Altova GmbH

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SpyDatabaseConnection
{
   public void ReleaseInstance[1859]();
   public String GetADOConnection[1659]();
   public void SetADOConnection[1659]( String sValue );
   public boolean GetAsAttributes[1659]();
   public void SetAsAttributes[1659]( boolean bValue );
   public boolean GetCreateMissingTables[1660]();
   public void SetCreateMissingTables[1660]( boolean bValue );
   public boolean GetCreateNew[1660]();
   public void SetCreateNew[1660]( boolean bValue );
   public boolean GetExcludeKeys[1661]();
   public void SetExcludeKeys[1661]( boolean bValue );
   public String GetFile[1661]();
   public void SetFile[1661]( String sValue );
   public boolean GetIncludeEmptyElements[1662]();
   public void SetIncludeEmptyElements[1662]( boolean bValue );
   public long[1896] GetNumberDateTimeFormat[1662]();
   public void SetNumberDateTimeFormat[1662]( long[1896] nValue );
   public String GetODBCConnection[1662]();
   public void SetODBCConnection[1662]( String sValue );
   public String GetSQLSelect[1663]();
   public void SetSQLSelect[1663]( String sValue );
   public long GetTextFieldLen[1664]();
   public void SetTextFieldLen[1664]( long nValue );

   // since version 2004R4
   public long[1891] GetDatabaseKind[1660]();
   public void SetDatabaseKind[1660]( long[1891] nValue );

   // since version 2008R2
   public boolean GetCommentIncluded[1659]();
   public void SetCommentIncluded[1659]( boolean bValue );
   public String GetNullReplacement[1662]();
   public void SetNullReplacement[1662]( String sValue );
   public String GetDatabaseSchema[1660]();
   public void SetDatabaseSchema[1660]( String sValue );

   // since version 2010r3
   public boolean GetPrimaryKeys[1663]()
   public void SetPrimaryKeys[1663]( boolean bValue )
   public boolean GetForeignKeys[1661]()
   public void SetForeignKeys[1661]( boolean bValue )
   public boolean GetUniqueKeys[1664]()
   public void SetUniqueKeys[1664]( boolean bValue )
   public long[1900] GetSchemaExtensionType[1663]()
   public void SetSchemaExtensionType[1663]( long[1900] nValue )
   public long[1900] GetSchemaFormat[1663]()
   public void SetSchemaFormat[1663]( long[1900] nValue )
   public long[1895] GetImportColumnsType[1661]()
   public void SetImportColumnsType[1661]( long[1895] nValue )
}
```

29.3.5.5 SpyDialogs (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
// Since version 2004R4
public class SpyDialogs
{
   public SpyApplication[1861] GetApplication[1665]();
   public SpyApplication[1861] GetParent[1665]();
   public SpyCodeGeneratorDlg[1862] GetCodeGeneratorDlg[1665]();
   public SpyFileSelectionDlg[1869] GetFileSelectionDlg[1665]();
   public SpySchemaDocumentationDlg[1874] GetSchemaDocumentationDlg[1666]();
   public SpyGenerateSampleXMLDlg[1871] GetGenerateSampleXMLDlg[1666]();
   public SpyDTDSchemaGeneratorDlg[1866] GetDTDSchemaGeneratorDlg[1666]();
   public SpyFindInFilesDlg[1869] GetFindInFilesDlg[1667]();
   public SpyWSDLDocumentationDlg[1879] GetWSDLDocumentationDlg[1667]();

   // Since version 2010
   public SpyWSDL20DocumentationDlg[1877] GetWSDL20DocumentationDlg[1667]();
   public SpyXBRLDocumentationDlg[1881] GetXBRLDocumentationDlg[1667]();
}
```

29.3.5.6 SpyDoc (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SpyDoc
{
   public void ReleaseInstance[1859]();
   public void SetEncoding[1679]( String strEncoding );
   public void SetPathName[1679]( String strPath );
   public String GetPathName[1687]();
   public String GetTitle[1699]();
   public boolean IsModified[1687]();
   public void Save[1691]();
   public void Close[1675]( boolean bDiscardChanges );
   public void UpdateViews[1697]();
   public long[1902] GetCurrentViewMode[1678]();
   public boolean SwitchViewMode[1695]( long[1902] nMode );
   public SpyGridView[1872] GetGridView[1687]();
```

```
public void SetActiveDocument();
public void StartChanges();
public void EndChanges();
public void TransformXSL();
public void AssignDTD( String sDTDFile, boolean bDialog );
public void AssignSchema( String sSchemaFile, boolean bDialog );
public void AssignXSL( String sXSLFile, boolean bDialog );
public void ConvertDTDOrSchema( long nFormat, long nFrequentElements );
public SpyXMLData CreateChild( long nKind );
public void CreateSchemaDiagram( long nKind, String sName, String sFile );
public SpyDocEditView GetDocEditView();
public void ExportToDatabase( SpyXMLData oFromChild, SpyExportSettings oExportSettings,
SpyDatabaseConnection oDatabaseConnection );
public void ExportToText( SpyXMLData oFromChild, SpyExportSettings oExportSettings,
SpyTextImportExportSettings oTextSettings );
public void GenerateDTDOrSchema( long nFormat, int nValuesList, long nDetection, long
nFrequentElements );
public SpyElementList GetExportElementList( SpyXMLData oFromChild, SpyExportSettings
oExportSettings );
public SpyXMLData GetRootElement();
public String SaveInString( SpyXMLData oData, boolean bMarked );
public void SaveToURL( String sUrl, String sUser, String sPassword );
public String GetErrorString(); // See IsValid() or IsWellFormed()
public int GetErrorPos(); // See IsValid() or IsWellFormed()
public SpyXMLData GetBadData(); // See IsValid() or IsWellFormed()
public boolean IsValid();
public boolean IsWellFormed( SpyXMLData oData, boolean bWithChildren );

// Since version 2004R3
public SpyAuthenticView GetAuthenticView()

// Since version 2004R4
public SpyApplication GetApplication();
public SpyDocuments GetParent();
public String GetFullName();
public void SetFullName( String i_strName );
public String GetName();
public String GetPath();
public boolean GetSaved();
public void SaveAs( String i_strFileNameOrPath );
public String GetEncoding();
public SpyXMLData GetDataRoot();
public void GenerateProgramCode( SpyCodeGeneratorDlg i_dlg );
public void AssignXSLFO( String i_strFile, boolean i_bUseDialog );
public void TransformXSLFO();
public void GenerateSchemaDocumentation( SpySchemaDocumentationDlg i_dlg );

public void ExecuteXQuery( String i_strXMLSourceFile );
public void SetExternalIsValid( boolean bIsValid );
public SpyDoc GenerateSampleXML( SpyGenerateSampleXMLDlg ipGenerateXMLDlg );
public boolean UpdateXMLData();
public String GetAsXMLString();
public void SetAsXMLString( String newVal );
public SpyDoc GenerateDTDOrSchemaEx( SpyDTDSchemaGeneratorDlg ipDTDSchemaGeneratorDlg );
public SpyDoc ConvertDTDOrSchemaEx( long nFormat, long nFrequentElements, String sOutputPath,
long nOutputPathDialogAction );
public SpyTextView GetTextView();
public String[] GetSuggestions();
public void SetSuggestions( String[] aList );
public void SetSelection( String sVal );

// Since version 2009
public void GenerateWSDLDocumentation( SpyWSDLDocumentationDlg ipWSDLDocumenationDlg );
```

```
   public void TransformXSLEx( long nDialogAction );

   // Since version 2010
   public void GenerateWSDL20Documentation( SpyWSDL20DocumentationDlg ipWSD20DocumenationDlg );
   public void GenerateXBRLDocumentation( SpyXBRLDocumentationDlg ipXBRLDocumentationDlg );
   public SpyDoc ConvertToWSDL20( String sFilePath, boolean bShowDialogs );

   // Since version 2010r3
   public String CreateDBStructureFromXMLSchema( SpyDatabaseConnection oConnection, SpyElementList
   oTables, boolean bDropTableWithExistingName );
   public SpyElementList GetDBStructureList( SpyDatabaseConnection oConnection );
}
```

29.3.5.7 SpyDocuments (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyDocuments
{
   public void ReleaseInstance();
   public long Count();
   public SpyDoc GetItem( long nNo );
   public SpyDoc NewFile( String strFile, String strType );
   public SpyDoc NewFileFromText( String nSource, String strType );
   public SpyDoc OpenFile( String sPath, boolean bDialog );
   public SpyDoc OpenURL( String sUrl, long nURLType, long nLoading, String sUser, String sPassword );
   public SpyDoc OpenURLDialog(String sURL, long nURLType, long nLoading, String sUser, String
   sPassword );
   // Since version 2011r2
   public SpyDoc NewAuthenticFile( String strSPSPath, String strXMLPath );
   public SpyDoc OpenAuthenticFile( String strSPSPath, String strXMLPath );
}
```

29.3.5.8 SpyDTDSchemaGeneratorDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyDTDSchemaGeneratorDlg
{
  public void ReleaseInstance();
```

```
  public SpyApplication[1861] GetApplication[1702]();
  public long[1892] GetDTDSchemaFormat[1703]();
  public void SetDTDSchemaFormat[1703]( long[1892] newVal );
  public short GetValueList[1706]();
  public void SetValueList[1706]( short newVal );
  public long[1901] GetTypeDetection[1706]();
  public void SetTypeDetection[1706]( long[1901] newVal );
  public long[1894] GetFrequentElements[1703]();
  public void SetFrequentElements[1703]( long[1894] newVal );
  public boolean GetMergeAllEqualNamed[1704]();
  public void SetMergeAllEqualNamed[1704]( boolean newVal );
  public boolean GetResolveEntities[1705]();
  public void SetResolveEntities[1705]( boolean newVal );
  public long[1889] GetAttributeTypeDefinition[1703]();
  public void SetAttributeTypeDefinition[1703]( long[1889] newVal );
  public boolean GetGlobalAttributes[1704]();
  public void SetGlobalAttributes[1704]( boolean newVal );
  public boolean GetOnlyStringEnums[1704]();
  public void SetOnlyStringEnums[1704]( boolean newVal );
  public long GetMaxEnumLength[1704]();
  public void SetMaxEnumLength[1704]( long newVal );
  public String GetOutputPath[1705]();
  public void SetOutputPath[1705]( String newVal );
  public long[1892] GetOutputPathDialogAction[1705]();
  public void SetOutputPathDialogAction[1705]( long[1892] newVal );
}
```

29.3.5.9 SpyElementList (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SpyElementList
{
  public void ReleaseInstance[1859]();
  public long GetCount[1707]();
  public SpyElementListItem[1868] GetItem[1707]( long nIndex );
  public void RemoveElement[1707]( long nIndex );
}
```

29.3.5.10 SpyElementListItem (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1567.

```
public class SpyElementListItem
{
   public void ReleaseInstance 1859();
   public long 1903 GetElementKind 1707();
   public void SetElementKind 1707( long 1903 nKind );
   public long GetFieldCount 1708();
   public String GetName 1708();
   public long GetRecordCount 1708();
}
```

29.3.5.11 SpyExportSettings (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1567.

```
public class SpyExportSettings
{
   public void ReleaseInstance 1859();
   public boolean GetCreateKeys 1709();
   public void SetCreateKeys 1709( boolean bValue );
   public SpyElementList 1867 GetElementList 1709();
   public void SetElementList 1709( SpyElementList 1867 obj );
   public boolean GetEntitiesToText 1709 ();
   public void SetEntitiesToText 1709( boolean bValue );
   public boolean GetExportAllElements 1709();
   public void SetExportAllElements 1709( boolean bValue );
   public boolean GetFromAttributes 1710();
   public void SetFromAttributes 1710( boolean bValue );
   public boolean GetFromSingleSubElements 1710();
   public void SetFromSingleSubElements 1710( boolean bValue );
   public boolean GetFromTextValues 1710();
   public void SetFromTextValues 1710( boolean bValue );
   public boolean GetIndependentPrimaryKey 1710();
   public void SetIndependentPrimaryKey 1710( boolean bValue );
   public long 1893 GetNamespace 1710();
   public void SetNamespace 1710( long 1893 nValue );
   public int GetSubLevelLimit 1711();
```


(C) 2015-2021 Altova GmbH

```
   public void SetSubLevelLimit( int nValue );
}
```

29.3.5.12 SpyFileSelectionDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2004R4
public class SpyFileSelectionDlg
{
   public void ReleaseInstance();
   public SpyApplication GetApplication();
   public SpyDialogs GetParent();
   public String GetFullName();
   public void SetFullName( String i_strName );
   public long GetDialogAction();
   public void SetDialogAction( long i_eAction );
}
```

29.3.5.13 SpyFindInFilesDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyFindInFilesDlg
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public String GetFind();
  public void SetFind( String sNewVal );
  public boolean GetRegularExpression();
  public void SetRegularExpression( boolean bNewVal );
  public String GetReplace();
  public void SetReplace( String sNewVal );
  public boolean GetReplaceOnDisk();
  public void SetReplaceOnDisk( boolean bNewVal );
  public boolean GetDoReplace();
  public void SetDoReplace( boolean bNewVal );
  public boolean GetMatchWholeWord();
```

```
  public void SetMatchWholeWord( boolean bNewVal );
  public boolean GetMatchCase();
  public void SetMatchCase( boolean bNewVal );
  public long GetSearchLocation();
  public void SetSearchLocation( long nPosition );
  public String GetStartFolder();
  public void SetStartFolder( String sNewVal );
  public boolean GetIncludeSubfolders();
  public void SetIncludeSubfolders( boolean bNewVal );
  public boolean GetSearchInProjectFilesDoExternal();
  public void SetSearchInProjectFilesDoExternal( boolean bNewVal );
  public String GetFileExtension();
  public void SetFileExtension( String sNewVal );
  public boolean GetAdvancedXMLSearch();
  public void SetAdvancedXMLSearch( boolean bNewVal );
  public boolean GetXMLElementNames();
  public void SetXMLElementNames( boolean bNewVal );
  public boolean GetXMLElementContents();
  public void SetXMLElementContents( boolean bNewVal );
  public boolean GetXMLAttributeNames();
  public void SetXMLAttributeNames( boolean bNewVal );
  public boolean GetXMLAttributeContents();
  public void SetXMLAttributeContents( boolean bNewVal );
  public boolean GetXMLComments();
  public void SetXMLComments( boolean bNewVal );
  public boolean GetXMLCData();
  public void SetXMLCData( boolean bNewVal );
  public boolean GetXMLPI();
  public void SetXMLPI( boolean bNewVal );
  public boolean GetXMLRest();
  public void SetXMLRest( boolean bNewVal );
  public boolean GetShowResult();
  public void SetShowResult( boolean bNewVal );
}
```

29.3.5.14 SpyFindInFilesMatch (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyFindInFilesMatch
{
  public void ReleaseInstance();
  public long Line();
  public long Position();
```

```
  public long Length();
  public String LineText();
  public boolean Replaced();
}
```

29.3.5.15 SpyFindInFilesResult (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyFindInFilesResult
{
  public void ReleaseInstance();
  public long Count();
  public SpyFindInFilesMatch GetItem( long nNo );
  public String GetPath();
  public SpyDoc GetDocument();
}
```

29.3.5.16 SpyFindInFilesResults (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyFindInFilesResults
{
  public void ReleaseInstance();
  public long Count();
  public SpyFindInFilesResult GetItem( long nNo );
}
```

29.3.5.17 SpyGenerateSampleXMLDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012

onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SpyGenerateSampleXMLDlg
{
  public void ReleaseInstance[1859]();
  public SpyApplication[1861] GetApplication[1725]();
  public boolean GetNonMandatoryAttributes[1727]();
  public void SetNonMandatoryAttributes[1727]( boolean newVal );
  public boolean GetNonMandatoryElements[1727]();
  public void SetNonMandatoryElements[1727]( boolean newVal );
  public boolean GetTakeFirstChoice[1729]();
  public void SetTakeFirstChoice[1729]( boolean newVal );
  public long GetRepeatCount[1728]();
  public void SetRepeatCount[1728]( long newVal );
  public boolean GetFillWithSampleData[1726]();
  public void SetFillWithSampleData[1726]( boolean newVal );
  public boolean GetFillElementsWithSampleData[1726]();
  public void SetFillElementsWithSampleData[1726]( boolean newVal );
  public boolean GetFillAttributesWithSampleData[1726]();
  public void SetFillAttributesWithSampleData[1726]( boolean newVal );
  public boolean GetContentOfNillableElementsIsNonMandatory[1726]();
  public void SetContentOfNillableElementsIsNonMandatory[1726]( boolean newVal );
  public boolean GetTryToUseNonAbstractTypes[1729]();
  public void SetTryToUseNonAbstractTypes[1729]( boolean newVal );
  public long[1898] GetOptimization[1728]();
  public void SetOptimization[1728]( long[1898] newVal );
  public long[1898] GetSchemaOrDTDAssignment[1729]();
  public void SetSchemaOrDTDAssignment[1729]( long[1898] newVal );
  public String GetLocalNameOfRootElement[1727]();
  public void SetLocalNameOfRootElement[1727]( String newVal );
  public String GetNamespaceURIOfRootElement[1727]();
  public void SetNamespaceURIOfRootElement[1727]( String newVal );
  public long[1892] GetOptionsDialogAction[1728]();
  public void SetOptionsDialogAction[1728]( long[1892] newVal );
}
```

29.3.5.18 SpyGridView (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

(C) 2015-2021 Altova GmbH

```
public class SpyGridView
{
   public void ReleaseInstance();
   public SpyXMLData GetCurrentFocus();
   public void Deselect( SpyXMLData oData );
   public boolean GetIsVisible();
   public void Select( SpyXMLData oData );
   public void SetFocus( SpyXMLData oData );
}
```

29.3.5.19 SpyProject (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyProject
{
   public void ReleaseInstance();
   public void CloseProject( boolean bDiscardChanges, boolean bCloseFiles, boolean bDialog );
   public String GetProjectFile();
   public void SetProjectFile( String sFile );
   public SpyProjectItems GetRootItems();
   public void SaveProject();
   public void SaveProjectAs( String sPath, boolean bDialog );
}
```

29.3.5.20 SpyProjectItem (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyProjectItem
{
   public void ReleaseInstance();
   public SpyProjectItems GetChildItems();
   public String GetFileExtensions();
   public void SetFileExtensions( String sExtensions );
   public long GetItemType();
   public String GetName();
   public SpyDoc Open();
   public SpyProjectItem GetParentItem();
```

```
   public String GetPath();
   public String GetValidateWith();
   public void SetValidateWith( String sVal );
   public String GetXMLForXSLTransformation();
   public void SetXMLForXSLTransformation( String sVal );
   public String GetXSLForXMLTransformation();
   public void SetXSLForXMLTransformation( String sVal );
   public String GetXSLTransformationFileExtension();
   public void SetXSLTransformationFileExtension( String sVal );
   public String GetXSLTransformationFolder();
   public void SetXSLTransformationFolder( String sVal );
}
```

29.3.5.21 SpyProjectItems (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyProjectItems
{
   public void ReleaseInstance();
   public void AddFile( String sPath );
   public void AddFolder( String sName );
   public void AddURL( String sURL, long nURLType, String sUser, String sPassword, boolean bSave );
   public long Count();
   public SpyProjectItem GetItem( long nNumber );
   public void RemoveItem( SpyProjectItem oItemToRemove );
}
```

29.3.5.22 SpySchemaDocumentationDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2004R4
public class SpySchemaDocumentationDlg
{
   public void ReleaseInstance();
   public SpyApplication GetApplication();
   public SpyDialogs GetParent();

   public String GetOutputFile();
```

```
public void SetOutputFile( String i_strVal );
public long GetOutputFormat();
public void SetOutputFormat( long i_eVal );

public boolean GetShowResult();
public void SetShowResult( boolean i_bVal );
public long GetOptionsDialogAction();
public void SetOptionsDialogAction( long i_eVal );
public long GetOutputFileDialogAction();
public void SetOutputFileDialogAction( long i_eVal );
public boolean GetShowProgressBar();
public void SetShowProgressBar( boolean i_bVal );

public void IncludeAll( boolean i_bInclude );
public boolean GetIncludeIndex();
public void SetIncludeIndex( boolean i_bVal );
public boolean GetIncludeGlobalElements();
public void SetIncludeGlobalElements( boolean i_bVal );
public boolean GetIncludeLocalElements();
public void SetIncludeLocalElements( boolean i_bVal );
public boolean GetIncludeGroups();
public void SetIncludeGroups( boolean i_bVal );
public boolean GetIncludeComplexTypes();
public void SetIncludeComplexTypes( boolean i_bVal );
public boolean GetIncludeSimpleTypes();
public void SetIncludeSimpleTypes( boolean i_bVal );
public boolean GetIncludeAttributeGroups();
public void SetIncludeAttributeGroups( boolean i_bVal );
public boolean GetIncludeRedefines();
public void SetIncludeRedefines( boolean i_bVal );

public void AllDetails( boolean i_bDetailsOn );
public boolean GetShowDiagram();
public void SetShowDiagram( boolean i_bVal );
public boolean GetShowNamespace();
public void SetShowNamespace( boolean i_bVal );
public boolean GetShowType();
public void SetShowType( boolean i_bVal );
public boolean GetShowChildren();
public void SetShowChildren( boolean i_bVal );
public boolean GetShowUsedBy();
public void SetShowUsedBy( boolean i_bVal );
public boolean GetShowProperties();
public void SetShowProperties( boolean i_bVal );
public boolean GetShowSingleFacets();
public void SetShowSingleFacets( boolean i_bVal );
public boolean GetShowPatterns();
public void SetShowPatterns( boolean i_bVal );
public boolean GetShowEnumerations();
public void SetShowEnumerations( boolean i_bVal );
public boolean GetShowAttributes();
public void SetShowAttributes( boolean i_bVal );
public boolean GetShowIdentityConstraints();
public void SetShowIdentityConstraints( boolean i_bVal );
public boolean GetShowAnnotations();
public void SetShowAnnotations( boolean i_bVal );
public boolean GetShowSourceCode();
public void SetShowSourceCode( boolean i_bVal );

// Since version 2009
public boolean GetEmbedDiagrams();
public void SetEmbedDiagrams( boolean i_bVal );
```

```
   public long GetDiagramFormat();
   public void SetDiagramFormat( long i_nVal );
   public boolean GetIncludeGlobalAttributes();
   public void SetIncludeGlobalAttributes( boolean i_bVal );
   public boolean GetIncludeLocalAttributes();
   public void SetIncludeLocalAttributes( boolean i_bVal );
   public boolean GetIncludeReferencedSchemas();
   public void SetIncludeReferencedSchemas( boolean i_bVal );
   public boolean GetMultipleOutputFiles();
   public void SetMultipleOutputFiles( boolean i_bVal );

   // Since version 2010
   public boolean GetEmbedCSSInHTML();
   public void SetEmbedCSSInHTML( boolean i_bVal );
   public boolean GetCreateDiagramsFolder();
   public void SetCreateDiagramsFolder( boolean i_bVal );

   // Since version 2010r3
   public boolean GetGenerateRelativeLinks();
   public void SetGenerateRelativeLinks( boolean i_bVal );

   // Since version 2011r2
   public boolean GetUseFixedDesign();
   public void SetUseFixedDesign( boolean i_bVal );
   public String GetSPSFile();
   public void SetSPSFile( String i_strVal );
   }
```

29.3.5.23 SpyTextImportExportSettings (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SpyTextImportExportSettings
{
   public void ReleaseInstance();
   public String GetDestinationFolder();
   public void SetDestinationFolder( String sVal );
   public long GetEnclosingCharacter();
   public void SetEnclosingCharacter( long nEnclosing );
   public String GetEncoding();
   public void SetEncoding( String sVal );
   public long GetEncodingByteOrder();
   public void SetEncodingByteOrder( long nByteOrder );
   public long GetFieldDelimiter();
   public void SetFieldDelimiter( long nDelimiter );
   public String GetFileExtension ();
   public void SetFileExtension( String sVal );
   public boolean GetHeaderRow();
   public void SetHeaderRow( boolean bVal );
```

```
  public String GetImportFile();
  public void SetImportFile( String sVal );
}
```

29.3.5.24 SpyTextView (obsolete)

```
public class SpyTextView
{
  public void ReleaseInstance();
  public SpyApplication GetApplication();
  public SpyDoc GetParent();
  public long LineFromPosition( long nCharPos );
  public long PositionFromLine( long nLine );
  public long LineLength( long nLine );
  public String GetSelText();
  public void SetSelText( String sText );
  public String GetRangeText( long nPosFrom, long nPosTill );
  public void ReplaceText( long nPosFrom, long nPosTill, String sText );
  public void MoveCaret( long nDiff );
  public void GoToLineChar( long nLine, long nChar );
  public void SelectText( long nPosFrom, long nPosTill );
  public long GetSelectionStart();
  public void SetSelectionStart( long nNewVal );
  public long GetSelectionEnd();
  public void SetSelectionEnd( long nNewVal );
  public String GetText();
  public void SetText( String sText );
  public long LineCount();
  public long Length();
}
```

29.3.5.25 SpyWSDL20DocumentationDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2010
public class SpyWSDL20DocumentationDlg
{
	public void ReleaseInstance();
	public SpyApplication GetApplication();

	public long GetOptionsDialogAction();
	public void SetOptionsDialogAction( long nNewVal );
```

```
public long GetOutputFileDialogAction();
public void SetOutputFileDialogAction( long nNewVal );

public boolean GetShowProgressBar();
public void SetShowProgressBar( boolean bNewVal );

public String GetOutputFile();
public void SetOutputFile( String sNewVal );

public long GetOutputFormat();
public void SetOutputFormat( long nNewVal );

public boolean GetMultipleOutputFiles();
public void SetMultipleOutputFiles( boolean bNewVal );

public boolean GetEmbedCSSInHTML();
public void SetEmbedCSSInHTML( boolean bNewVal );

public long GetDiagramFormat();
public void SetDiagramFormat( long nNewVal );

public boolean GetEmbedDiagrams();
public void SetEmbedDiagrams( boolean bNewVal );

public boolean GetCreateDiagramsFolder();
public void SetCreateDiagramsFolder( boolean bNewVal );

public boolean GetShowResult();
public void SetShowResult( boolean bNewVal );

public void IncludeAll( boolean bNewVal );
public void AllDetails( boolean bNewVal );

public boolean GetIncludeOverview();
public void SetIncludeOverview( boolean bNewVal );

public boolean GetIncludeService();
public void SetIncludeService( boolean bNewVal );

public boolean GetIncludeBinding();
public void SetIncludeBinding( boolean bNewVal );

public boolean GetIncludeInterface();
public void SetIncludeInterface( boolean bNewVal );

public boolean GetIncludeTypes();
public void SetIncludeTypes( boolean bNewVal );

public boolean GetIncludeImportedWSDLFiles();
public void SetIncludeImportedWSDLFiles( boolean bNewVal );

public boolean GetShowServiceDiagram();
public void SetShowServiceDiagram( boolean bNewVal );
```

```
        public boolean GetShowBindingDiagram();
        public void SetShowBindingDiagram( boolean bNewVal );

        public boolean GetShowInterfaceDiagram();
        public void SetShowInterfaceDiagram( boolean bNewVal );

        public boolean GetShowTypesDiagram();
        public void SetShowTypesDiagram( boolean bNewVal );

        public boolean GetShowEndpoint();
        public void SetShowEndpoint( boolean bNewVal );

        public boolean GetShowSourceCode();
        public void SetShowSourceCode( boolean bNewVal );

        public boolean GetShowExtensibility();
        public void SetShowExtensibility( boolean bNewVal );

        public boolean GetShowUsedBy();
        public void SetShowUsedBy( boolean bNewVal );

        public boolean GetShowOperation();
        public void SetShowOperation( boolean bNewVal );

        public boolean GetShowFault();
        public void SetShowFault( boolean bNewVal );

        // Since version 2011r2
        public boolean GetUseFixedDesign();
        public void SetUseFixedDesign( boolean i_bVal );

        public String GetSPSFile();
        public void SetSPSFile( String i_strVal );

}
```

29.3.5.26 SpyWSDLDocumentationDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2008r2sp1
public class SpyWSDLDocumentationDlg
{
        public void ReleaseInstance();
```

```
public SpyApplication GetApplication ();

public String GetOutputFile ();
public void SetOutputFile ( String sNewVal );

public long GetOutputFileDialogAction ();
public void SetOutputFileDialogAction ( long nNewVal );

public long GetOptionsDialogAction ();
public void SetOptionsDialogAction ( long nNewVal );

public boolean GetShowProgressBar ();
public void SetShowProgressBar ( boolean bNewVal );

public boolean GetShowResult ();
public void SetShowResult ( boolean bNewVal );

public long GetOutputFormat ();
public void SetOutputFormat ( long nNewVal );

public boolean GetEmbedDiagrams ();
public void SetEmbedDiagrams ( boolean bNewVal );

public long GetDiagramFormat ();
public void SetDiagramFormat ( long nNewVal );

public boolean GetMultipleOutputFiles ();
public void SetMultipleOutputFiles ( boolean bNewVal );

public void IncludeAll ( boolean bNewVal );

public boolean GetIncludeBinding ();
public void SetIncludeBinding ( boolean bNewVal );

public boolean GetIncludeImportedWSDLFiles ();
public void SetIncludeImportedWSDLFiles ( boolean bNewVal );

public boolean GetIncludeMessages ();
public void SetIncludeMessages ( boolean bNewVal );

public boolean GetIncludeOverview ();
public void SetIncludeOverview ( boolean bNewVal );

public boolean GetIncludePortType ();
public void SetIncludePortType ( boolean bNewVal );

public boolean GetIncludeService ();
public void SetIncludeService ( boolean bNewVal );

public boolean GetIncludeTypes ();
public void SetIncludeTypes ( boolean bNewVal );

public void AllDetails ( boolean bNewVal );
```

```
        public boolean GetShowBindingDiagram1768();
        public void SetShowBindingDiagram1768( boolean bNewVal );

        public boolean GetShowExtensibility1769();
        public void SetShowExtensibility1769( boolean bNewVal );

        public boolean GetShowMessageParts1769();
        public void SetShowMessageParts1769( boolean bNewVal );

        public boolean GetShowPort1769();
        public void SetShowPort1769( boolean bNewVal );

        public boolean GetShowPortTypeDiagram1769();
        public void SetShowPortTypeDiagram1769( boolean bNewVal );

        public boolean GetShowPortTypeOperations1770();
        public void SetShowPortTypeOperations1770( boolean bNewVal );

        public boolean GetShowServiceDiagram1771();
        public void SetShowServiceDiagram1771( boolean bNewVal );

        public boolean GetShowSourceCode1771();
        public void SetShowSourceCode1771( boolean bNewVal );

        public boolean GetShowTypesDiagram1771();
        public void SetShowTypesDiagram1771( boolean bNewVal );

        public boolean GetShowUsedBy1772();
        public void SetShowUsedBy1772( boolean bNewVal );

        // Since version 2010
        public boolean GetEmbedCSSInHTML1763();
        public void SetEmbedCSSInHTML1763( boolean i_bVal );

        public boolean GetCreateDiagramsFolder1762();
        public void SetCreateDiagramsFolder1762( boolean i_bVal );

        // Since version 2011r2
        public boolean GetUseFixedDesign1772();
        public void SetUseFixedDesign1772( boolean i_bVal );

        public String GetSPSFile1772();
        public void SetSPSFile1772( String i_strVal );

}
```

29.3.5.27 SpyXBRLDocumentationDlg (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
// Since version 2010
public class SpyXBRLDocumentationDlg
{
	public void ReleaseInstance[1859]();
	public SpyApplication[1861] GetApplication[1785]();

	public long[1892] GetOptionsDialogAction[1789]();
	public void SetOptionsDialogAction[1789]( long[1892] nNewVal );

	public long[1892] GetOutputDialogAction[1789]();
	public void SetOutputDialogAction[1789]( long[1892] nNewVal );

	public boolean GetShowProgressBar[1793]();
	public void SetShowProgressBar[1793]( boolean bNewVal );

	public String GetOutputFile[1789]();
	public void SetOutputFile[1789]( String sNewVal );

	public long[1899] GetOutputFormat[1789]();
	public void SetOutputFormat[1789]( long[1899] nNewVal );

	public boolean GetEmbedCSSInHTML[1786]();
	public void SetEmbedCSSInHTML[1786]( boolean bNewVal );

	public long[1894] GetDiagramFormat[1786]();
	public void SetDiagramFormat[1786]( long[1894] nNewVal );

	public boolean GetEmbedDiagrams[1786]();
	public void SetEmbedDiagrams[1786]( boolean bNewVal );

	public boolean GetCreateDiagramsFolder[1785]();
	public void SetCreateDiagramsFolder[1785]( boolean bNewVal );

	public boolean GetShowResult[1794]();
	public void SetShowResult[1794]( boolean bNewVal );

	public void IncludeAll[1787]( boolean bNewVal );
	public void AllDetails[1785]( boolean bNewVal );

	public boolean GetIncludeOverview[1788]();
	public void SetIncludeOverview[1788]( boolean bNewVal );

	public boolean GetIncludeNamespacePrefixes[1788]();
	public void SetIncludeNamespacePrefixes[1788]( boolean bNewVal );

	public boolean GetIncludeGlobalElements[1787]();
	public void SetIncludeGlobalElements[1787]( boolean bNewVal );

	public boolean GetIncludeDefinitionLinkroles[1787]();
```

```
        public void SetIncludeDefinitionLinkroles(1787)( boolean bNewVal );

        public boolean GetIncludePresentationLinkroles(1788)();
        public void SetIncludePresentationLinkroles(1788)( boolean bNewVal );

        public boolean GetIncludeCalculationLinkroles(1787)();
        public void SetIncludeCalculationLinkroles(1787)( boolean bNewVal );

        public boolean GetShowDiagram(1791)();
        public void SetShowDiagram(1791)( boolean bNewVal );

        public boolean GetShowSubstitutiongroup(1794)();
        public void SetShowSubstitutiongroup(1794)( boolean bNewVal );

        public boolean GetShowItemtype(1792)();
        public void SetShowItemtype(1792)( boolean bNewVal );

        public boolean GetShowBalance(1791)();
        public void SetShowBalance(1791)( boolean bNewVal );

        public boolean GetShowPeriod(1793)();
        public void SetShowPeriod(1793)( boolean bNewVal );

        public boolean GetShowAbstract(1790)();
        public void SetShowAbstract(1790)( boolean bNewVal );

        public boolean GetShowNillable(1792)();
        public void SetShowNillable(1792)( boolean bNewVal );

        public boolean GetShowLabels(1792)();
        public void SetShowLabels(1792)( boolean bNewVal );

        public boolean GetShowReferences(1793)();
        public void SetShowReferences(1793)( boolean bNewVal );

        public boolean GetShowLinkbaseReferences(1792)();
        public void SetShowLinkbaseReferences(1792)( boolean bNewVal );

        public boolean GetShortQualifiedName(1790)();
        public void SetShortQualifiedName(1790)( boolean bNewVal );

        public boolean GetShowImportedElements(1791)();
        public void SetShowImportedElements(1791)( boolean bNewVal );

        // Since version 2011r2
        public boolean GetUseFixedDesign(1795)();
        public void SetUseFixedDesign(1795)( boolean i_bVal );

        public String GetSPSFile(1794)();
        public void SetSPSFile(1794)( String i_strVal );
};
```

29.3.5.28 SpyXMLData (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SpyXMLData
{
   public void ReleaseInstance[1859]();
   public void AppendChild[1795]( SpyXMLData[1884] oNewData );
   public void EraseAllChildren[1797]();
   public void EraseCurrentChild[1798]();
   public SpyXMLData[1884] GetCurrentChild[1800]();
   public SpyXMLData[1884] GetFirstChild[1800]( long[1903] nKind );
   public SpyXMLData[1884] GetNextChild[1801]();
   public boolean GetHasChildren[1803]();
   public void InsertChild[1803]( SpyXMLData[1884] oNewData );
   public boolean IsSameNode[1804]( SpyXMLData[1884] oToComp);
   public long[1903] GetKind[1805]();
   public boolean GetMayHaveChildren[1805]();
   public String GetName[1805]();
   public void SetName[1805]( String sValue );
   public SpyXMLData[1884] GetParent[1805]();
   public String GetTextValue[1806]();
   public void SetTextValue[1806]( String sValue );
}
```

29.3.5.29 Authentic (obsolete)

29.3.5.29.1 SpyAuthenticRange (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
// Since version 2004R3
public class SpyAuthenticRange
{
    public void ReleaseInstance[1859]();
    public SpyApplication[1861] GetApplication[1608]();
    public SpyAuthenticView[1886] GetParent[1628]();
```

```
    public SpyAuthenticRange[1884] GotoNext[1618]( long[1890] eKind );
    public SpyAuthenticRange[1884] GotoPrevious[1619]( long[1890] eKind );
    public void Select[1630]();
    public long GetFirstTextPosition[1613]();
    public void SetFirstTextPosition[1613]( long nTextPosition );
    public long GetLastTextPosition[1624]();
    public void SetLastTextPosition[1624]( long nTextPosition );
    public String GetText[1633]();
    public void SetText[1633]( String strText );
    public boolean PerformAction[1629]( long[1889] eAction, String strElementName );
    public boolean CanPerformAction[1608]( long[1889] eAction, String strElementName );
    public String[] CanPerformActionWith[1609]( long[1889] eAction );
    public SpyAuthenticRange[1884] GoTo[1618]( long[1890] eKind, long nCount, long[1890] nFrom );
    public SpyAuthenticRange[1884] SelectNext[1630]( long[1890] eKind );
    public SpyAuthenticRange[1884] SelectPrevious[1631]( long[1890] eKind );
    public SpyAuthenticRange[1884] MoveBegin[1627]( long[1890] eKind, long nCount );
    public SpyAuthenticRange[1884] MoveEnd[1627]( long[1890] eKind, long nCount );
    public SpyAuthenticRange[1884] ExpandTo[1612]( long[1890] eKind );
    public SpyAuthenticRange[1884] CollapsToBegin[1609]();
    public SpyAuthenticRange[1884] CollapsToEnd[1610]();
    public SpyAuthenticRange[1884] GotoNextCursorPosition[1619]();
    public SpyAuthenticRange[1884] GotoPreviousCursorPosition[1620]();
    public boolean IsEmpty[1622]();
    public boolean IsEqual[1623]( SpyAuthenticRange[1884] ipCmp );
    public SpyAuthenticRange[1884] Clone[1609]();
    public SpyAuthenticRange[1884] SetFromRange[1633]( SpyAuthenticRange[1884] ipSrc );
    public boolean Delete[1610]();
    public boolean Cut[1610]();
    public boolean Copy[1610]();
    public boolean Paste[1628]();
    public SpyXMLData[1884] GetFirstXMLData[1613]();
    public void SetFirstXMLData[1613]( SpyXMLData[1884] objXMLDataPtr );
    public long GetFirstXMLDataOffset[1614]();
    public void SetFirstXMLDataOffset[1614]( long nOffset );
    public SpyXMLData[1884] GetLastXMLData[1625]();
    public void SetLastXMLData[1625]( SpyXMLData[1902] objXMLDataPtr );
    public long GetLastXMLDataOffset[1626]();
    public void SetLastXMLDataOffset[1626]( long nOffset );
    public String[] GetElementHierarchy[1616]();
    public String[] GetElementAttributeNames[1615]( String strElementName );
    public boolean HasElementAttribute[1620]( String strElementName, String strAttributeName
);
    public String GetElementAttributeValue[1616]( String strElementName, String
strAttributeName );
    public void SetElementAttributeValue[1632]( String strElementName, String
strAttributeName, String strNewValue );
    public String[] GetEntityNames[1617]();
    public void InsertEntity[1620]( String strEntityName );
    public boolean IsInDynamicTable[1623]();
    public boolean AppendRow[1607]();
    public boolean InsertRow[1621]();
    public boolean DuplicateRow[1611]();
    public boolean DeleteRow[1611]();
```

```
    public boolean MoveRowUp();
    public boolean MoveRowDown();

    // Since version 2004R4
    public boolean IsCopyEnabled();
    public boolean IsCutEnabled();
    public boolean IsPasteEnabled();
    public boolean IsDeleteEnabled();
    public boolean IsTextStateApplied( String i_strElementName );
    public boolean IsFirstRow();
    public boolean IsLastRow();
}
```

29.3.5.29.2 SpyAuthenticView (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
// Since version 2004R3
public class SpyAuthenticView
{
    public void ReleaseInstance();
    public SpyApplication GetApplication();
    public SpyDoc GetParent();
    public SpyAuthenticRange GetSelection();
    public void SetSelection( SpyAuthenticRange obj );
    public SpyAuthenticRange GetDocumentBegin();
    public SpyAuthenticRange GetDocumentEnd();
    public SpyAuthenticRange GetWholeDocument();
    public long GetMarkupVisibility();
    public void SetMarkupVisibility( long eSpyAuthenticMarkupVisibility );
    public SpyAuthenticRange GoTo( long eKind, long nCount, long nFrom );
    public void Print( boolean bWithPreview, boolean bPromptUser );
    public boolean Undo();
    public boolean Redo();
    public void UpdateXMLInstanceEntities();

    // Since version 2004R4
    public String GetAsXMLString();
    public void SetAsXMLString( String i_strXML );
    public SpyXMLData GetXMLDataRoot();
    public boolean IsUndoEnabled();
    public boolean IsRedoEnabled();
}
```

29.3.5.29.3 SpyDocEditSelection (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1567].

```
public class SpyDocEditSelection
{
   public void ReleaseInstance [1859]();
   public SpyXMLData [1884] GetEnd [1819]();
   public long GetEndTextPosition [1820]();
   public SpyXMLData [1884] GetStart [1820]();
   public long GetStartTextPosition [1820]();
}
```

29.3.5.29.4 SpyDocEditView (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1567].

```
public class SpyDocEditView
{
   public void ReleaseInstance [1859]();
   public void ApplyTextState [1822]( String sElementName );
   public SpyDocEditSelection [1887] GetCurrentSelection [1823]();
   public void EditClear [1824]();
   public void EditCopy [1824]();
   public void EditCut [1825]();
   public void EditPaste [1825]();
   public void EditRedo [1826]();
   public void EditSelectAll [1826]();
   public void EditUndo [1826]();
   public SpyXMLData [1884] GetNextVisible [1829]( SpyXMLData [1884] oElement );
   public SpyXMLData [1884] GetPreviousVisible [1830]( SpyXMLData [1884] oElement );
   public boolean GetIsEditClearEnabled [1830]();
   public boolean GetIsEditCopyEnabled [1831]();
   public boolean GetIsEditCutEnabled [1831]();
   public boolean GetIsEditPasteEnabled [1832]();
   public boolean GetIsEditRedoEnabled [1832]();
   public boolean GetIsEditUndoEnabled [1833]();
   public boolean GetIsRowAppendEnabled [1833]();
   public boolean GetIsRowDeleteEnabled [1834]();
   public boolean GetIsRowDuplicateEnabled [1834]();
```

```
   public boolean GetIsRowInsertEnabled();
   public boolean GetIsRowMoveDownEnabled();
   public boolean GetIsRowMoveUpEnabled();
   public boolean IsTextStateApplied( String sElementName );
   public boolean IsTextStateEnabled( String sElementName );
   public void LoadXML( String sXML );
   public void MarkUpView( long nKind );
   public void RowAppend();
   public void RowDelete();
   public void RowDuplicate();
   public void RowInsert();
   public void RowMoveDown();
   public void RowMoveUp();
   public String SaveXML();
   public void SelectionMoveTabOrder( boolean bForward, boolean bTag );
   public boolean SelectionSet( SpyXMLData oStart, long nStartPos, SpyXMLData oEndElement, long
   nEndPos );
   public SpyXMLData GetXMLRoot();
   public String[] GetAllowedElements( long nAction, SpyXMLData oStartPtr, SpyXMLData oEndPtr );
}
```

29.3.5.30 Predefined constants (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

This section lists all classes that define the predefined constants used by the Java interface.

29.3.5.30.1 SPYApplicationStatus (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java.

```
public class SPYApplicationStatus
  {
      public final static long spyApplicationStatus_Running                 = 0;
      public final static long spyApplicationStatus_AfterLicenseCheck       = 1;
      public final static long spyApplicationStatus_BeforeLicenseCheck      = 2;
      public final static long                                              = 3;
      spyApplicationStatus_ConcurrentLicenseCheckFailed
      public final static long spyApplicationStatus_ProcessingCommandLine   = 4;
  }
```

29.3.5.30.2 SPYAttributeTypeDefinition (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYAttributeTypeDefinition
{
   public final static long              = 0;
   spyMergedGlobal
   public final static long              = 1;
   spyDistinctGlobal
   public final static long spyLocal     = 2;
}
```

29.3.5.30.3 SPYAuthenticActions (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYAuthenticActions
{
    public final static long spyAuthenticInsertAt   = 0;
    public final static long spyAuthenticApply      = 1;
    public final static long spyAuthenticClearSurr  = 2;
    public final static long spyAuthenticAppend     = 3;
    public final static long                        = 4;
    spyAuthenticInsertBefore
    public final static long spyAuthenticRemove     = 5;
}
```

29.3.5.30.4 SPYAuthenticDocumentPosition (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYAuthenticDocumentPosition
{
    public final static long                          = 0;
    spyAuthenticDocumentBegin
    public final static long                          = 1;
    spyAuthenticDocumentEnd
    public final static long spyAuthenticRangeBegin = 2;
    public final static long spyAuthenticRangeEnd   = 3;
}
```

29.3.5.30.5 SPYAuthenticElementKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYAuthenticElementKind
{
   public final static long              = 0;
   spyAuthenticChar
   public final static long              = 1;
   spyAuthenticWord
   public final static long              = 3;
   spyAuthenticLine
   public final static long              = 4;
   spyAuthenticParagraph
   public final static long              = 6;
   spyAuthenticTag
   public final static long              = 8;
   spyAuthenticDocument
   public final static long              = 9;
   spyAuthenticTable
```


(C) 2015-2021 Altova GmbH

```
    public final static long                = 10;
    spyAuthenticTableRow
    public final static long                = 11;
    spyAuthenticTableColumn
}
```

29.3.5.30.6 SPYAuthenticMarkupVisibility (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1567.

```
public class SPYAuthenticMarkupVisibility
{
    public final static long                          = 0;
    spyAuthenticMarkupHidden
    public final static long spyAuthenticMarkupSmall= 1;
    public final static long spyAuthenticMarkupLarge= 2;
    public final static long spyAuthenticMarkupMixed= 3;
}
```

29.3.5.30.7 SPYDatabaseKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1567.

```
public class SPYLoading
 {
    public final static long spyDB_Access   = 0;
    public final static long                = 1;
    spyDB_SQLServer
    public final static long spyDB_Oracle   = 2;
    public final static long spyDB_Sybase   = 3;
    public final static long spyDB_MySQL    = 4;
    public final static long spyDB_DB2      = 5;
    public final static long spyDB_Other    = 6;
    public final static long                = 7;
    spyDB_Unspecified
```

```
}
```

29.3.5.30.8 SPYDialogAction (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYDialogAction
 {
    public final static long spyDialogOK    = 0;
    public final static long                = 1;
    spyDialogCancel
    public final static long                = 2;
    spyDialogUserInput
 }
```

29.3.5.30.9 SPYDOMType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYDOMType
 {
    public final static long               = 0;
    spyDOMType_msxml4
    public final static long               = 1;
    spyDOMType_xerces
 }
```

29.3.5.30.10 SPYDTDSchemaFormat (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

(C) 2015-2021 Altova GmbH

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYDTDSchemaFormat
{
    public final static long spyDTD     = 0;
    public final static long spyDCD     = 1;
    public final static long spyXMLData = 2;
    public final static long spyBizTalk = 3;
    public final static long spyW3C     = 4;
}
```

29.3.5.30.11 SPYEncodingByteOrder (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYEncodingByteOrder
{
    public final static long spyNONE          = 0;
    public final static long spyLITTLE_ENDIAN = 1;
    public final static long spyBIG_ENDIAN    = 2;
}
```

29.3.5.30.12 SPYExportNamespace (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYExportNamespace
{
    public final static long spyNoNamespace                = 0;
    public final static long spyReplaceColonWithUnderscore = 1;
}
```

29.3.5.30.13 SPYFindInFilesSearchLocation (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1567].

```
public class SPYFindInFilesSearchLocation
{
   public final static long                    = 0;
   spyFindInFiles_Documents
   public final static long                    = 1;
   spyFindInFiles_Project
   public final static long                    = 2;
   spyFindInFiles_Folder
}
```

29.3.5.30.14 SPYFrequentElements (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1567].

```
public class SPYFrequentElements
{
   public final static long                 = 0;
   spyGlobalElements
   public final static long                 = 1;
   spyGlobalComplexType
}
```

29.3.5.30.15 SPYImageKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

(C) 2015-2021 Altova GmbH

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYImageKind
 {
    public final static long               = 0;
    spyImageType_PNG
    public final static long               = 1;
    spyImageType_EMF
 }
```

29.3.5.30.16 SPYImportColumnsType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

29.3.5.30.17 SPYLibType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYLibType
 {
    public final static long               = 0;
    spyLibType_static
    public final static long spyLibType_dll= 1;
 }
```

29.3.5.30.18 SPYLoading (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYLoading
{
    public final static long               = 0;
    spyUseCacheProxy
    public final static long spyReload     = 1;
}
```

29.3.5.30.19 SPYNumberDateTimeFormat (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class  SPYNumberDateTimeFormat
{
    public final static long spySystemLocale = 0;
    public final static long                 = 1;
    spySchemaCompatible
}
```

29.3.5.30.20 SPYProgrammingLanguage (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYLoading
 {
```


(C) 2015-2021 Altova GmbH

```
    public final static long spyUndefinedLanguage = -1;
    public final static long spyJava              = 0;
    public final static long spyCpp               = 1;
    public final static long spyCSharp            = 2;
}
```

29.3.5.30.21 SPYProjectItemTypes (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1567.

```
public class SPYProjectItemTypes
{
    public final static long            = 0;
    spyUnknownItem
    public final static long spyFileItem= 1;
    public final static long            = 2;
    spyFolderItem
    public final static long spyURLItem = 3;
}
```

29.3.5.30.22 SPYProjectType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java 1567.

```
public class SPYProjectType
 {
    public final static long                  = 0;
    spyVisualStudioProject
    public final static long                  = 1;
    spyVisualStudio2003Project
    public final static long spyBorlandProject = 2;
    public final static long spyMonoMakefile   = 3;
 }
```

29.3.5.30.23 SPYSampleXMLGenerationOptimization (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYSampleXMLGenerationOptimization
{
   public final static long spySampleXMLGen_Optimized   = 0;
   public final static long                             = 1;
   spySampleXMLGen_NonMandatoryElements
   public final static long spySampleXMLGen_Everything  = 2;
}
```

29.3.5.30.24 SPYSampleXMLGenerationSchemaOrDTDAssignment (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYSampleXMLGenerationOptimization
{
   public final static long                                = 0;
   spySampleXMLGen_AssignRelatively
   public final static long                                = 1;
   spySampleXMLGen_AssignAbsolutely
   public final static long spySampleXMLGen_DoNotAssign= 2;
}
```

29.3.5.30.25 SPYSchemaDefKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

(C) 2015-2021 Altova GmbH

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYSchemaDefKind
{
    public final static long spyKindElement       = 0;
    public final static long spyKindComplexType   = 1;
    public final static long spyKindSimpleType    = 2;
    public final static long spyKindGroup         = 3;
    public final static long spyKindModel         = 4;
    public final static long spyKindAny           = 5;
    public final static long spyKindAttr          = 6;
    public final static long spyKindAttrGroup     = 7;
    public final static long spyKindAttrAny       = 8;
    public final static long spyKindIdentityUnique= 9;
    public final static long spyKindIdentityKey   = 10;
    public final static long spyKindIdentityKeyRef= 11;
    public final static long                      = 12;
    spyKindIdentitySelector
    public final static long spyKindIdentityField = 13;
    public final static long spyKindNotation      = 14;
    public final static long spyKindInclude       = 15;
    public final static long spyKindImport        = 16;
    public final static long spyKindRedefine      = 17;
    public final static long spyKindFacet         = 18;
    public final static long spyKindSchema        = 19;
    public final static long spyKindCount         = 20;
}
```

29.3.5.30.26 SPYSchemaDocumentationFormat (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYSchemaDocumentationFormat
 {
    public final static long                = 0;
    spySchemaDoc_HTML
    public final static long                = 1;
    spySchemaDoc_MSWord
    public final static long spySchemaDoc_RTF        = 2;
    public final static long spySchemaDoc_PDF        = 3;
 }
```

29.3.5.30.27 SPYSchemaExtensionType (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1567].

```
public class SPYSchemaExtensionType
 {
    public final static long spySchemaExtension_None    = 0;
    public final static long spySchemaExtension_SQL_XML = 1;
    public final static long                            = 2;
    spySchemaExtension_MS_SQL_Server
    public final static long spySchemaExtension_Oracle  = 3;
 }
```

29.3.5.30.28 SPYSchemaFormat (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1567].

```
public class SPYSchemaFormat
 {
    public final static long                        = 0;
    spySchemaFormat_Hierarchical
    public final static long spySchemaFormat_Flat   = 1;
 }
```

29.3.5.30.29 SPYTextDelimiters (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1567].

(C) 2015-2021 Altova GmbH

```
public class SPYTextDelimiters
{
    public final static long spyTabulator = 0;
    public final static long spySemicolon = 1;
    public final static long spyComma     = 2;
    public final static long spySpace     = 3;
}
```

29.3.5.30.30 SPYTextEnclosing (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYTextEnclosing
{
    public final static long           = 0;
    spyNoEnclosing
    public final static long           = 1;
    spySingleQuote
    public final static long           = 2;
    spyDoubleQuote
}
```

29.3.5.30.31 SPYTypeDetection (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYTypeDetection
{
    public final static long               = 0;
    spyBestPossible
    public final static long               = 1;
    spyNumbersOnly
    public final static long               = 2;
    spyNoDetection
}
```

29.3.5.30.32 SPYURLTypes (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYURLTypes
{
    public final static long              = (-1);
    spyURLTypeAuto
    public final static long              = 0;
    spyURLTypeXML
    public final static long              = 1;
    spyURLTypeDTD
}
```

29.3.5.30.33 SpyViewModes (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java[1567].

```
public class SPYViewModes
{
    public final static long spyViewGrid     = 0;
    public final static long spyViewText     = 1;
    public final static long spyViewBrowser  = 2;
    public final static long spyViewSchema   = 3;
    public final static long spyViewContent  = 4;
    public final static long                 = 4;
    spyViewAuthentic
    public final static long spyViewWSDL     = 5;
    public final static long spyViewZIP      = 6;
    public final static long                 = 7;
    spyViewEditionInfo
}
```


(C) 2015-2021 Altova GmbH

29.3.5.30.34 SPYWhitespaceComparison (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1567].

```
public class SPYWhitespaceComparison
 {
    public final static long spyCompareAsIs  = 0;
    public final static long                 = 1;
    spyCompareNormalized
    public final static long spyStripAll     = 2;
 }
```

29.3.5.30.35 SPYXMLDataKind (obsolete)

The objects described in this section (Application API for Java) are obsolete from v2012 onwards.

For information about how to access the Application API from Java code, see the section: Programming Languages | Java [1567].

```
public class SPYXMLDataKind
{
   public final static long                  = 0;
   spyXMLDataXMLDocStruct
   public final static long                  = 1;
   spyXMLDataXMLEntityDocStruct
   public final static long                  = 2;
   spyXMLDataDTDDocStruct
   public final static long spyXMLDataXML    = 3;
   public final static long                  = 4;
   spyXMLDataElement
   public final static long spyXMLDataAttr   = 5;
   public final static long spyXMLDataText   = 6;
   public final static long spyXMLDataCData  = 7;
   public final static long                  = 8;
   spyXMLDataComment
   public final static long spyXMLDataPI     = 9;
   public final static long                  = 10;
   spyXMLDataDefDoctype
```

```
    public final static long                = 11;
    spyXMLDataDefExternalID
    public final static long                = 12;
    spyXMLDataDefElement
    public final static long                = 13;
    spyXMLDataDefAttlist
    public final static long                = 14;
    spyXMLDataDefEntity
    public final static long                = 15;
    spyXMLDataDefNotation
    public final static long                = 16;
    spyXMLDataKindsCount
}
```


(C) 2015-2021 Altova GmbH

29.4 ActiveX Integration

このセクションで説明されているXMLSpy ユーザーインターフェイスとActiveX コントロール機能を使用するアプリケーションに統合することができます。 ActiveX 技術によりC++、C#、VB.NET、HTML などの統合のために使用される広範囲の言語が有効化されます(HTML 内に統合されているActiveX コンポーネントは、Microsoft Internet Explorer バージョンとActiveX をサポートすると共に作動する必要があります)。全てのコンポーネントはフルOLE コントロールです。Java への統合はラッパークラスにより与えられます。

ActiveX コントロールを、カスタムコードに統合するには、XMLSpy Integration Package をインストールしてください(次を参照してください: http://www.altova.com/ide_integration.html)。すべてのXMLSpy を最初に、そして、XMLSpy Integration Package そして他の必要条件を言語とプラットフォームに応じてインストールしてください(次を参照してください: 必要条件[1905])。

統合の2つの異なるレベル間を柔軟に選択することができます: アプリケーションレベルとドキュメントレベル.

アプリケーションレベルの統合は、(メニュー、ツールバー、ペインを含む) XMLSpy のフルインターフェイスを ActiveX コントロールをカスタムアプリケーションとして埋め込むことを意味します。例えば、多くのシナリオでは、カスタムアプリケーションを、XMLSpy グラフィカル ユーザーインターフェイスを埋め込む1つのフォームから構成することができます。

ドキュメントレベルの統合よりもこのアプローチより簡単に実装することができますが、カスタム必要条件に従いを柔軟的にXMLSpy グラフィカル ユーザーインターフェイスを構成するためには適切ではない可能性があります。ドキュメントレベルの統合は、アプリケーションにXMLSpy を1つ埋め込むことを意味します。メインXMLSpy コントロールのみではなく、メインドキュメント エディターウィンドウの実装も含まれます。また、オプションで、追加ウィンドウの実装も含まれます。このアプローチはGUI の構成に柔軟性を与えますが、選択された言語によりActiveX コントロールとの高度なインタラクションが必要とされます。

セクションアプリケーションレベルの統合[1908]とドキュメントレベルの統合[1910] は、それぞれのレベルの主要なステップに関して説明しています。ActiveX 統合のサンプル[1913] セクションは、C#、HTML、および、Java のサンプルを提供します。これらのサンプルを確認することにより、正しい選択が直ぐに行うことができます。オブジェクトレファレンス[1951] セクションは、プロパティとメソッドに加え統合のために使用できるすべてのCOM オブジェクトに付いて説明しています。

XMLSpy をVisual Studio プラグインとして使用するための情報に関しては、次を参照してください: Visual Studio 内のXMLSpy[1005]。

29.4.1 必要条件

XMLSpy ActiveX コントロールをカスタムアプリケーションに統合するには、以下が使用中のコンピューターにインストールされている必要があります:

- XMLSpy
- XMLSpy Integration Package は、次からダウンロードできます: http://www.altova.com/ide_integration.html

64 ビット ActiveX コントロールを統合するには、XMLSpy の64 ビット バージョンとXMLSpy Integration Package をインストールする必要があります。Visual Studio を使用して、Microsoft .NET プラットフォームの下で開発されたアプリケーションに関しては、XMLSpy の32 ビット と64 ビット バージョンとXMLSpy Integration Package について下で説明されているとおりインストールします。

メモ Windows XP とWindows Server 2003 で統合パッケージが正確に作動するためには、再頒布可能パッケージを手動でインストールしてください: https://www.microsoft.com/en-US/download/details.aspx?id=48145.

Microsoft .NET (C#, VB.NET) with Visual Studio

XMLSpy ActiveX コントロールを、Microsoft .NET の下で開発された 32 ビット アプリケーションに統合するには、以下を使用中のコンピューターにインストールしてください:

- Microsoft .NET Framework 4.0 または以降
- Visual Studio 2010/2012/2013/2015/2017/2019
- XMLSpy 32 ビットとXMLSpy Integration Package 32 ビット
- ActiveX コントロールがVisual Studio ツールボックスに追加されている必要があります(次を参照してください: ActiveX コントロールをツールボックスに追加する 1906).

64 ビット ActiveX コントロールを統合する場合、次の必要条件が上記に加え満たされる必要があります:

- XMLSpy 32 ビットとXMLSpy Integration Package 32 ビット をインストールしてください(Visual Studio は32 ビットで作動するため、32 ビット ActiveX コントロールを Visual Studio デザイナーに与える必要があります。)
- XMLSpy 64 ビットとXMLSpy Integration Package 64 ビット をインストールしてください(実際の64 ビット ActiveX コントロールをランタイムにカスタムアプリケーションに与えます。)
- Visual Studio 内で、64 ビットビルド更新を作成し、この更新を使用してアプリケーションをビルドします。サンプルに関しては、次を参照してください: サンプルC# ソリューションの実行 1913。

Java

Eclipse 開発環境を使用して、XMLSpy ActiveX コントロールを Java アプリケーションに統合するには、次を使用中のコンピューターにインストールしてください:

- Java Runtime Environment(JRE) または Java Development Kit(JDK) 7 または以降
- Eclipse
- XMLSpy and XMLSpy Integration Package

メモ XMLSpy ActiveX コントロールの64 ビット バージョンを作動するには、Eclipse の64 ビット バージョン、および、XMLSpy とXMLSpy Integration Package の64 ビット バージョン ofを使用してください。

クライアントコンピューター上でのXMLSpy 統合とデプロイ

.NET アプリケーションを作成し他のクライアントに配布する場合は、クライアントコンピューター上に次をインストールしてください:

- XMLSpy
- XMLSpy Integration Package
- カスタムの統合コード、または、アプリケーション。

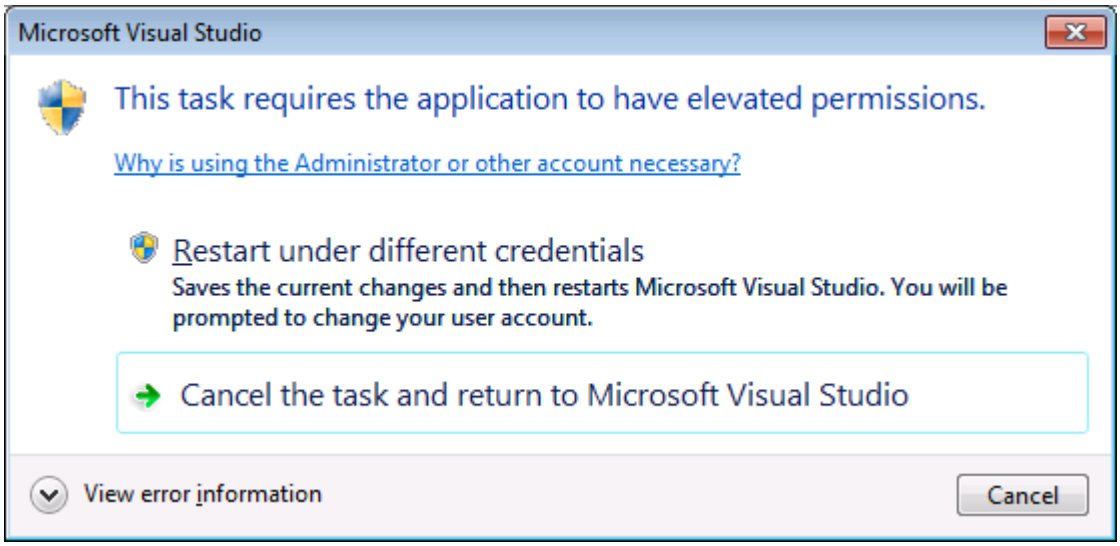
29.4.2 ActiveX コントロールをツールボックスに追加する

Visual Studio を使用して開発されたアプリケーション内で、XMLSpy ActiveX コントロールを使用するには、コントロールが最初に Visual Studio ツールボックスに追加される必要があります:

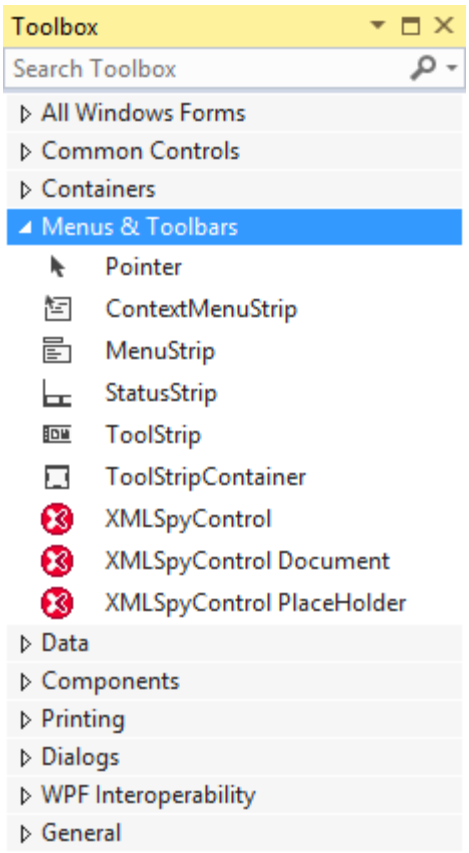
1. Visual Studio の「ツール」メニューから、「ツールボックスアイテムの選択」をクリックします。
2. 「COM コンポーネント」タブ、から、XMLSpyControl, XMLSpyControl ドキュメント、および、XMLSpyControl プレースホルダーの横のチェックボックスを選択します。

上記のコントロールが使用することができない場合、下のステップに従います:

1. 「COM コンポーネント」タブから「参照」をクリックします。**XMLSpyControl.ocx** をXMLSpy インストールからフォルダーを選択します。XMLSpy Integration Package をインストールすることを忘れないでください。それ以外の場合、このファイルを使用することはできません。次を参照してください: 必要条件[1905].
2. を更に高いパーミッションと共に再起動するようにプロンプトされると、「異なる資格情報下で再起動する」をクリックします。



上記のステップが完了すると、XMLSpy ActiveX コントロールは、Visual Studio ツールボックス内で使用できるようになります。



メモ アプリケーションレベルの統合では、**XMLSpyControl** ActiveX コントロールのみが使用されます（次を参照してください: アプリケーションレベルの統合[1908]）。 XMLSpyControl ドキュメント とXMLSpyControl プレースホルダー コントロールは、ドキュメントレベルの統合のために使用されます（次を参照してください: ドキュメントレベルの統合[1910]）。

29.4.3 アプリケーションレベルの統合

アプリケーションレベルの統合により使用中のアプリケーションのウィンドウにXMLSpy のインターフェイスを埋め込むことができます。 この種類の統合により、すべてのメニュー、ツールバー、ステータスバー、ドキュメントウィンドウ、および、ヘルパーウィンドウを含むXMLSpy のユーザーインターフェイスを使用することができます。アプリケーションのユーザーインターフェイスのカスタム化はXMLSpy が提供する機能に限られています。これには、ヘルパーウィンドウの整理とサイズ調整、及びメニューとツールバーのカスタム化が含まれています。

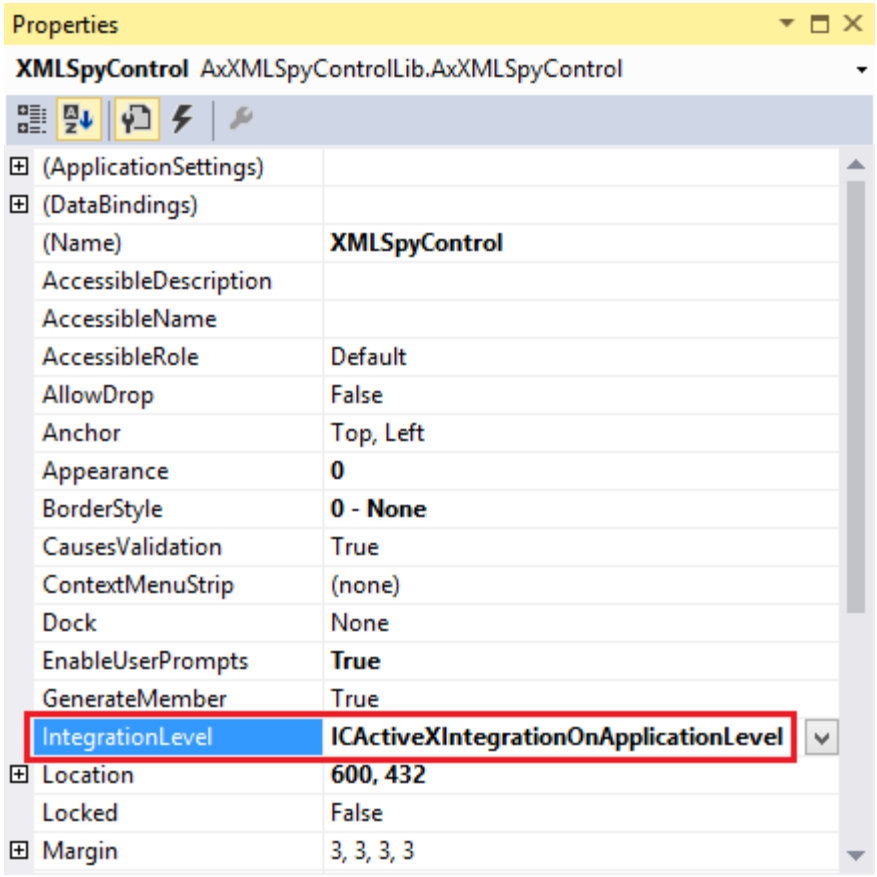
アプリケーションレベルで統合する場合、統合する必要のある唯一のActiveX コントロールは`XMLSpyControl`[1954] です。`XMLSpyControlDocument`[1962] または`XMLSpyControlPlaceHolder`[1968] ActiveX コントロールをインスタンス化、または、アクセス使用としないでください。

初期化、または、XMLSpy の振る舞いを自動化するには、`XMLSpyControl`[1954] のために説明されているプロパティ、メソッド、とイベントを使用してください。XMLSpy 機能への複雑なアクセスのために`XMLSpyControl.application`[1955] を使用することを考慮してください。

XMLSpy アプリケーションをHTML ページ、に埋め込むサンプルに関しては、次を参照してください: アプリケーションレベルのHTML 統合[1916]。

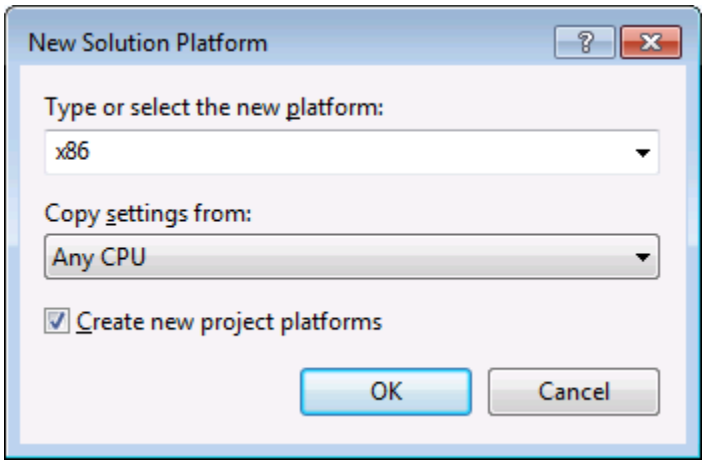
Visual Studio を使用する C# または VB.NET での XMLSpy ActiveX コントロールをアプリケーションレベルで統合する基本的なアプリケーションを作成するステップは、以下のとおりです:

1. すべての必要条件が満たされていることを確認してください（次を参照してください: 必要条件[1905]）.
2. 新規の Visual Studio Windows Forms プロジェクトを新規の空のフォーム共に作成してください。
3. まだ行っていない場合、ActiveX コントロールをツールボックスに追加します（次を参照してください: ActiveX コントロールをツールボックスに追加する[1906]）。
4. ツールボックスから、**XMLSpyControl** をドラッグして、新規のフォームにドロップしてください。
5. フォームで **XMLSpyControl** を選択し、プロパティウインドウ内で **IntegrationLevel** プロパティを **ICActiveXIntegrationOnApplicationLevel** に設定します。



6. ビルド（x86、x64）するプラットフォームに一致するビルドプラットフォームの更新を作成します。ビルドの更新の作成の方法は以下のとおりです:

   a. Visual Studio 内のソリューションを右クリックし、「**更新 マネージャー**」を選択します。
   b. 「**アクティブなソリューションプラットフォーム**」から、「**新規作成...**」を選択し、x86 または x64 更新を選択します（このサンプルでは、**x86**）。



Visual Studio 内でソリューションをビルドして実行する準備が整うと、ターゲットプラットフォーム(x86、x64)に一致する更新をしようしてビルドすることを確認してください。

29.4.4 ドキュメントレベルの統合

アプリケーションレベルの統合に比べ、ドキュメントレベルの統合複雑ですが、ActiveX コントロールを使用することによりアプリケーションに XMLSpy 機能を柔軟的に埋め込むことができます。このアプローチにより、コードは XMLSpy ユーザーインターフェイスの次のパーツに部分的にアクセスすることができます:

- ドキュメントの編集ウィンドウ
- プロジェクトウィンドウ
- 入力ヘルパーウィンドウ
- バリデーター出力ウィンドウ
- XPath プロファイラーウィンドウ
- XPath ダイアログウィンドウ
- XSLT/XQuery デバッガーウィンドウ
- SOAP デバッガーウィンドウ

アプリケーションレベルの統合[1908] で説明されているとおり、ActiveX アプリケーションレベルの統合のためには、**XMLSpyControl** のみが必要とされる唯一のコントロールです。しかしながら、ActiveX ドキュメントレベルの統合の場合は、機能 XMLSpy が次の ActiveX コントロールにより与えられます:

1. **XMLSpyControl**[1954]
2. XMLSpyControl ドキュメント[1962]
3. XMLSpyControl プレースホルダ[1968]

これらのコントロールは XMLSpy のアプリケーション インストール フォルダーで使用することのできる **XMLSpyControl.ocx** ファイルにより与えられます。Visual Studio を使用して ActiveX 統合を開発すると、これらのコントロールを Visual Studio ツールボックスに追加する必要があります(次を参照してください: ActiveX コントロールをツールボックスに追加する[1906])。

ドキュメントレベルでの ActiveX コントロールのアプリケーションへの統合の基本のステップは、以下のとおりです:

1. 最初に、アプリケーション内で **XMLSpyControl** をインスタンス化します。「**XMLSpyControl ドキュメント**」と「**XMLSpyControl プレースホルダ**」コントロールへのサポートが有効化されるために、このコントロールの統合は必須です。`IntegrationLevel`[1956] プロパティを「ICActiveXIntegrationOnDocumentLevel」(または「1」)に設定することは重要です。ユーザーに対してコントロールを非表示にするには、「Visible」プロパティを「False」に設定します。

メモ ドキュメントレベルで統合する場合、未知の結果が生成される可能性があるため、**XMLSpyControl** の「開く」メソッドを使用しないでください。**XMLSpyControl ドキュメント** と**XMLSpyControl プレースホルダ** の対応するメソッドを代わりに使用して開いてください。

2. **XMLSpyControl ドキュメント** のインスタンスを少なくとも1つアプリケーション内に作成します。このコントロールは、アプリケーションにXMLSpy のドキュメントの編集ウィンドウを与え、必要であれば、複数回インスタンス化することができます。

   既存のファイルを開くために、メソッド「開く」を使用します。ドキュメントに関連した機能にアクセスするには、メソッドとプロパティ「ドキュメント」を介してアクセスすることのできる「パス」と「保存」を使用します。

メモ コントロールは、読み取り専用モードをサポートしません。プロパティ「読み取り専用」の値は無視されます。

3. 任意で、アプリケーションに、アプリケーションで使用することのできる(ドキュメント ウィンドウ以外の)各追加ウィンドウのために **XMLSpyControl プレースホルダ** コントロールを追加することができます。

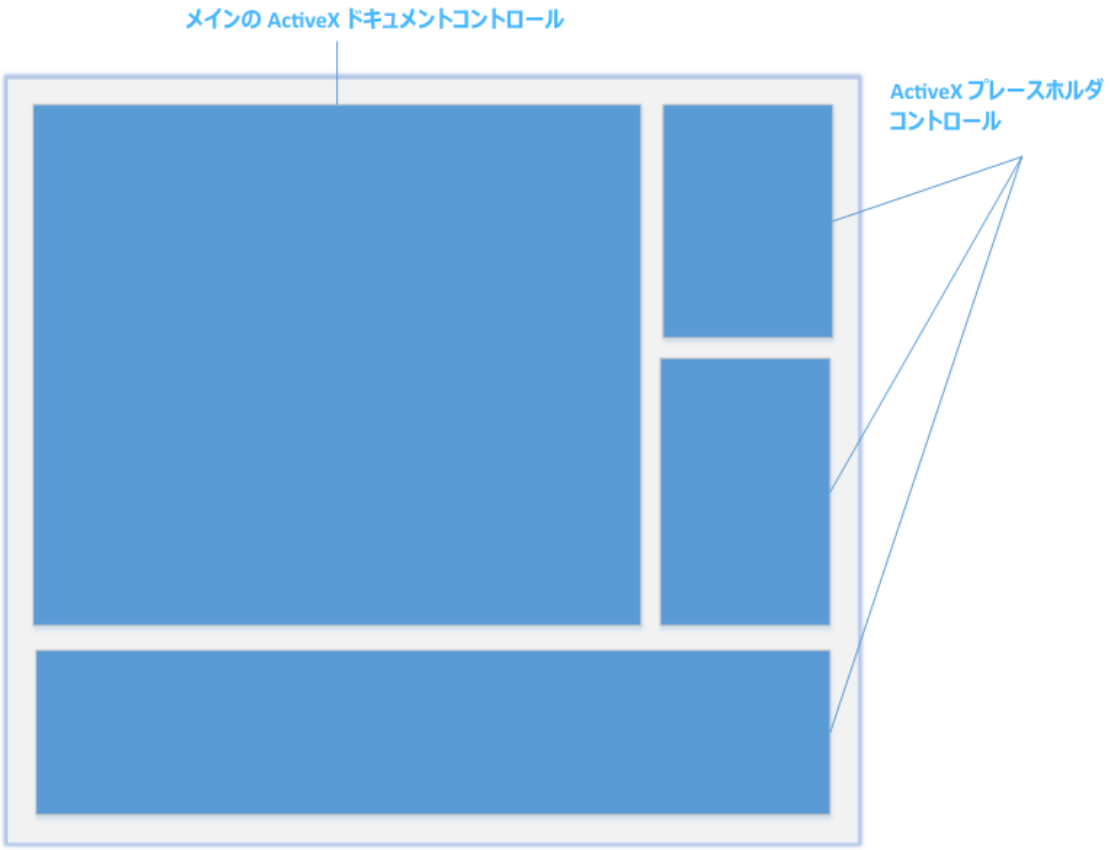
   XMLSpyControl プレースホルダ のインスタンスによりアプリケーションウィンドウに選択して XMLSpy の追加ウィンドウを埋め込むことができます。ウィンドウの種類(例えば、プロジェクト ウィンドウ) はプロパティ「PlaceholderW indowID」により定義することができます。有効なウィンドウの識別子のためには、次を参照してください: XMLSpyControlPlaceholderWindow[1971].

メモ 各ウィンドウ識別子のために **XMLSpyControl プレースホルダ**」のみを使用してください。

   XMLSpy プロジェクト ウィンドウを選択するプレースホルダーコントロールのために、追加メソッドを使用することができます。「OpenProject」を使用して、XMLSpy プロジェクトをロードすることができます。XMLSpy 自動化インターフェイスからプロパティプロジェクトとメソッドを使用して、他のプロジェクトに関連したオペレーションを操作します。

例えば、Visual Studio を使用したC# またはVB.NET 内では、XMLSpy ActiveX コントロールをドキュメントレベルで統合する基本的なアプリケーションの作成ステップは、以下にリストされるものに類似しています。必要に応じて適用は複雑になる可能性があります。しかしながら、ActiveX ドキュメントレベルの統合のための必要条件として、下の命令を理解することは重要です。

1. 新規の空のフォームと共に新規のVisual Studio W indows Forms プロジェクトを作成します。
2. まだ行っていない場合、ActiveX コントロールを ツールボックスに追加します(次を参照してください: ActiveX コントロールをツールボックスに追加する[1906])。
3. ツールボックスから **XMLSpyControl**[1954] をドラッグして新規のフォームにドラッグします。
4. **XMLSpyControl** の「IntegrationLevel」プロパティを「ICActiveXIntegrationOnDocumentLevel」に、と「Visible」プロパティを「False」に設定します。コードから、または、「プロパティ」ウィンドウから行うことができます。
5. XMLSpyControl ドキュメント[1962] をツールボックスからフォームにドラッグします。 このコントロールは、使用中のアプリケーションにXMLSpy のメインウィンドウを与え、ドキュメントのためにサイズを調整することができます。
6. 代わりに、フォームに1つまたは複数のXMLSpyControl プレースホルダ[1968] コントロールを追加することができます(アプリケーションが必要とする各追加ウィンドウの型など。例えば、「プロジェクト」ウィンドウなど)。このような追加プレースホルダーコントロールをメインドキュメントコントロールの下、または、左右に通常配置します。例:



7. 各「XMLSpyControl プレースホルダ」コントロールの「PlaceholderWindowID」プロパティを有効なウィンドウ識別子に設定してください。有効な値のリストに関しては、次を参照してください: XMLSpyControlPlaceholderWindow[1971]。
8. アプリケーションにコマンドを以下のように追加します(少なくとも、ドキュメントを開き、保存し、閉じる必要があります)。

XMLSpy コマンドをクエリする

ドキュメントレベルで統合する場合は、XMLSpy メニューまたはツールバーをアプリケーションで使用することはできません。代わりに、必要とするコマンドを取得し、状態を確認し、プログラム的に実行することができます:

- 使用することのできるすべてのコマンドを取得するには、**XMLSpyControl** の CommandsList[1956] プロパティを使用します。
- メニュー構造により整理されているコマンドを取得するには、メイン Menu[1957] プロパティを使用します。
- 表示される順番に整理されているコマンドを取得するには、Toolbars[1957] プロパティを使用します。
- XMLSpy にコマンドを送信するには、Exec[1958] メソッドを使用します。
- コマンドが現在有効化されているか、または、無効化されているかクエリするために、QueryStatus[1959] メソッドを使用します。

これにより XMLSpy コマンドをアプリケーションのメニューとツールバーに柔軟的に統合することができます。

XMLSpy のインストールは、XMLSpy 内で使用されているコマンドラベルを与えます。GIF 書式内のアイコンのために XMLSpy インストールのフォルダー「<ApplicationFolder>¥Examples¥ActiveX¥Images」を参照してください。fコマンド名に対応するファイル名は、コマンド参照[1932] セクション内にリストされています。

一般的な注意点

XMLSpy の振る舞いを自動化するには、**XMLSpyControl**[1954]、XMLSpyControl ドキュメント[1962] とXMLSpyControl プレースホルダ[1968] 内で説明されている プロパティ、メソッド、とイベントを使用してください。

XMLSpy 機能への高度なアクセスに関しては、次のプロパティを考慮してください:

- XMLSpyControl.Application[1955]
- XMLSpyControlDocument.Document[1963]
- XMLSpyControlPlaceHolder.Project[1969]

これらのプロパティはXMLSpy 自動化インターフェイス(XMLSpyAPI)へのアクセスを提供します

メモ ドキュメントを開くには、適切なドキュメントコントロールのXMLSpyControlDocument.Open[1965] または XMLSpyControlDocument.New[1965] を使用してください。 プロジェクトを開くには、XMLSpy プロジェクトウィンドウを埋め込むプレースホルダ コントロール上のXMLSpyControlPlaceHolder.OpenProject[1970] を常に使用してください。

異なるプログラミング環境内で必要なコントロールをインスタント化しアクセスする方法を説明するサンプルに関しては、次を参照してください: ActiveX 統合のサンプル[1913]。

29.4.5 ActiveX 統合のサンプル

このセクションは、異なるコンテナー環境とプログラム言語を使用するXMLSpy ドキュメントレベルの統合のサンプル、を含みます(HTML セクションには、追加でアプリケーションレベルの統合のサンプルが含まれます)。全てのサンプルのソースコードは、XMLSpy インストール内のフォルダー<ApplicationFolder>\Examples\ActiveX で使用することができます。

29.4.5.1 C#

C# and Visual Studio のための基本のActiveX 統合サンプルソリューションは、フォルダー <ApplicationFolder>\Examples\ActiveX\C# 内にあります。ソースコードをコンパイルする前に、サンプルを実行して、全ての必要条件が満たされていることを確認してください(次を参照してください: サンプルC# ソリューションの実行[1913])。

29.4.5.1.1 サンプル C# ソリューションの実行

フォルダー**<ApplicationFolder>\Examples\ActiveX\C#** 内で使用することのできるサンプルVisual Studio ソリューションは、XMLSpy ActiveX コントロールの使用方法について説明しています。このソリューションをビルドして実行する前に、次のステップに注意してください:

ステップ 1: 必要条件をチェックする

Visual Studio 2010 または以降は、サンプルソリューションを開く必要があります。必要条件の完全なリストは、次を参照してください: 必要条件[1905]。

ステップ 2: 書き込みのパーミッションを保有するディレクトリにサンプルをコピーする

Visual Studio を管理者として作動することを回避するには、デフォルトの場所から実行する代わりに、書き込みのパーミッションを持つディレクトリへのソースコードをコピーします。

ステップ 3: すべての必要とされるコントロールプロパティをチェックし設定する

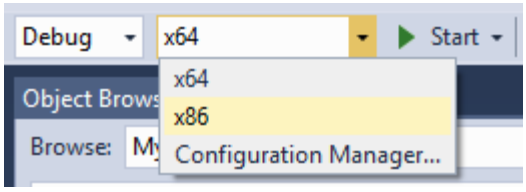
サンプルアプリケーションは、XMLSpyControlDocument[1962] の 1 つのインスタンスと XMLSpyControlPlaceHolder[1968] コントロールの 1 つのインスタンスを含んでいます。下のテーブルに表示されているとおりにこれらのコントロールの次のプロパティが設定されていることを確認してください:

コントロール名	プロパティ	プロパティ値
XMLSpyControl	IntegrationLevel	ICActiveXIntegrationOnDocumentLevel
XPathDialog	PlaceholderWindowID	16

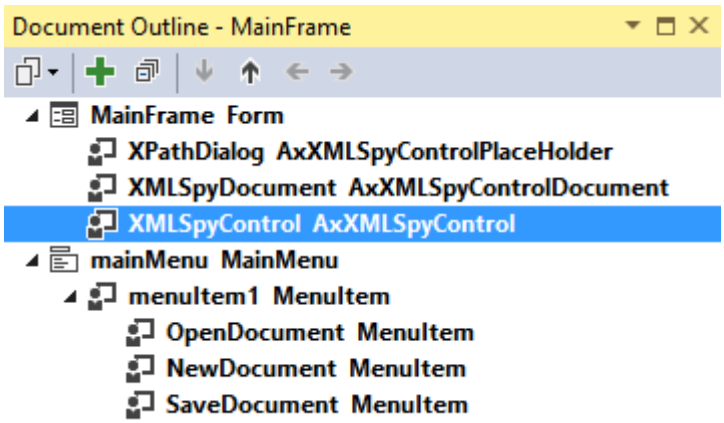
ActiveX コントロールのプロパティをビュー、または、設定する方法は、以下のとおりです:

1. デザイナーウィンドウ内で **MDIMain.cs** フォームを開きます。

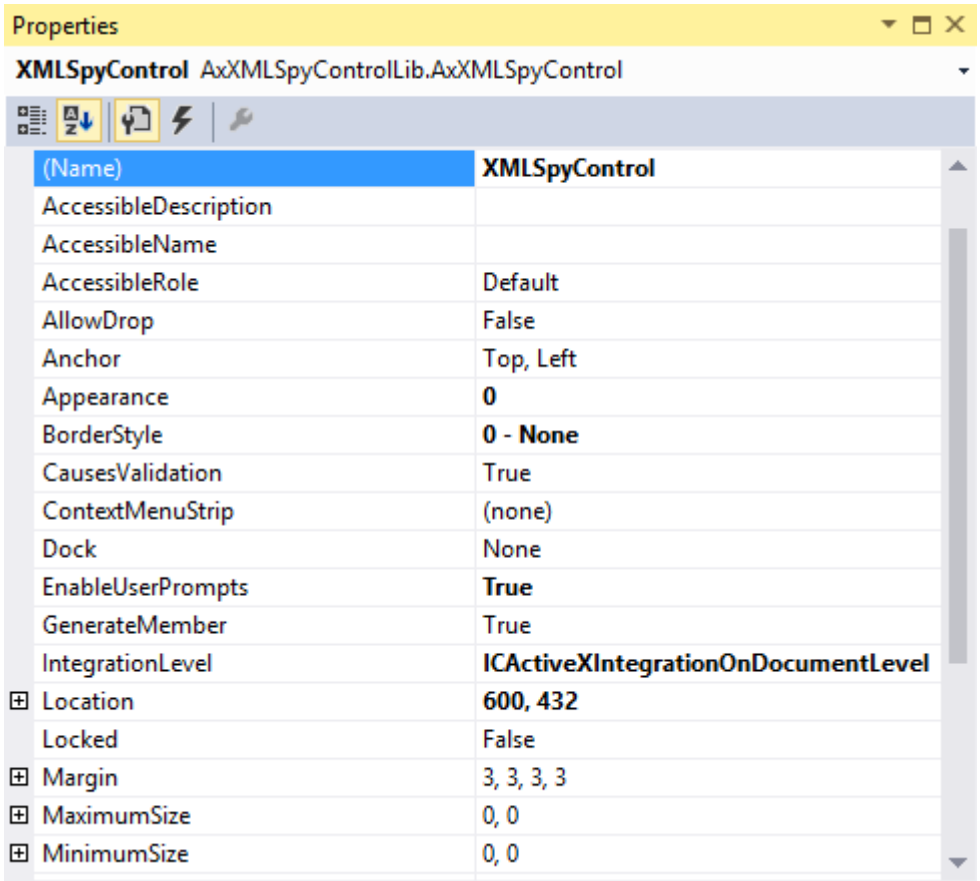
メモ 64 ビット Windows では、デザイナーウィンドウで開く前に Visual Studio ソリューションを「x86」にビルドを更新する必要がある可能性があります。64 ビットアプリケーションとしてサンプルをビルドする場合は、次を参照してください: 必要条件[1905].



2. Visual Studio の**ドキュメント概要** ウィンドウを開きます(「**表示**」メニューから、「**他のウィンドウ | ドキュメントの概要**」をクリックします)。



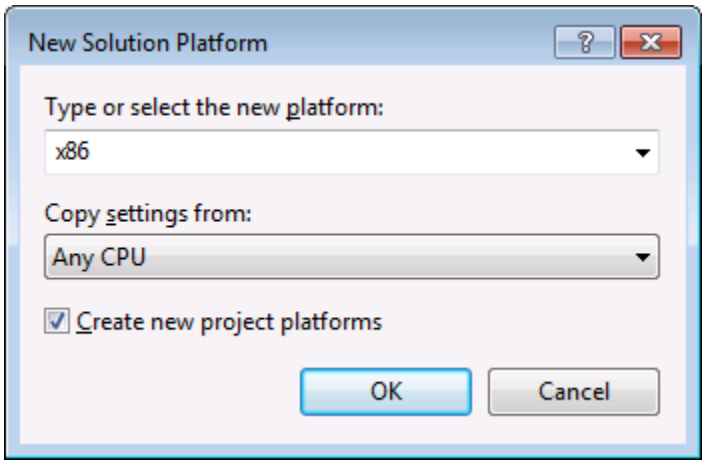
3. ドキュメント概要 ウィンドウ内のActiveX コントロールをクリックし、プロパティ ウィンドウ内の必要とされるプロパティを編集します。例:



ステップ 4: ビルドプラットフォームを設定する

- ビルド(x86、x64)するプラットフォームに一致するビルドプラットフォームの更新を作成します。ビルドの更新の作成の方法は以下のとおりです:

a. Visual Studio 内のソリューションを右クリックし、「**更新マネージャー**」を選択します。
b. 「**アクティブなソリューションプラットフォーム**」から、「**新規作成...**」を選択し、x86 または x64 更新を選択します（このサンプルでは、**x86**）。



Visual Studio 内でソリューションをビルドして実行する準備が整うと、ターゲットプラットフォーム（x86、x64）に一致するように更新をしようしてビルドすることを忘れないでください。それ以外の場合、ランタイムエラーが起こる可能性があります。

サンプルを実行するには、メイン MDI フレームウィンドウが作成され、この中には空の XML ドキュメントを持つ編集ウィンドウと XMLSpy の XPath ダイアログウィンドウが含まれています。「**ファイル | 開く**」を使用して、XMLSpy サンプルフォルダーから XML ファイルを開きます。ファイルはロードされ表示されます。ドキュメントのロード後、XPath ダイアログの使用を開始します。描きなおしコンテンツを表示するためにフォームの下の角をドラッグする必要があります。

29.4.5.2 HTML

このセクションのコードリストは、アプリケーションレベルとドキュメントレベルでの XMLSpyControl の統合方法を示しています。全てのサンプルのためのソースコードは XMLSpy インストールのフォルダー `<ApplicationFolder>\Examples\ActiveX\HTML` 内にあります。サンプルは、32 ビット アプリケーションとして作動する場合、Internet Explorer 内でのみ作動することができます。

メモ 64 ビット モード上で作動する場合、nternet Explorer 10 または以降は、ActiveX コントロールをロードしません。

29.4.5.2.1 アプリケーションレベルの HTML 統合

このサンプルは、アプリケーションレベルでの HTML ページへの XMLSpy コントロールの簡単な統合について説明しています。統合に関しては、次のセクションで説明されています:

- HTML コード内で XMLSpyControl を開始します。
- ボタンがドキュメントをロードするように実装し、コード生成タスクを自動化します。
- アプリケーションイベントのためのアクションを定義する。

このサンプルのためのコードは、XMLSpy インストール内の次の場所で使用することができます:
`<ApplicationFolder>\Examples\ActiveX\HTML\XMLSpyActiveX_ApplicationLevel.htm`.

29.4.5.2.1.1 コントロールインスタンスを作成する

HTML `Object` タグは、XMLSpyControl のインスタンスを作成するために使用されます。`Classid` は、XMLSpyControl です。幅と高さがウィンドウのサイズを指定します。アプリケーションレベルがデフォルトのため、追加のパラメーターは必要ありません。

```
<OBJECT id="objXMLSpyControl"
        Classid="clsid:a258bba2-3835-4c16-8590-72b44f52c471"
        width="1000"
        height="700"
        VIEWASTEXT>
</OBJECT>
```

29.4.5.2.1.2 デフォルトのドキュメントを開くボタンを追加する

一部のタスクの自動化のシンプルなサンプルとして、ページにボタンを追加します:

```
<input type="button" value="Open Marketing Expenses" onclick="BtnOpenMEFile()">
```

クリックされると、定義済みドキュメントが XMLSpyControl 内で開かれます。異なるインストール上でサンプルを作動するため、XMLSpyControl に対して相対的なファイルを検索するためにメソッドを使用します。

```
<SCRIPT ID=Javahandlers LANGUAGE=javascript>
// --------------------------------
// open a pre-defined document
function BtnOpenMEFile()
{

             objXMLSpyControl.Open("C:\Documents and Settings\username\My
Documents\Altova\XMLSpy2021\Examples/OrgChart.xml");

}
</SCRIPT>
```

29.4.5.2.1.3 コード生成のためにボタンを追加する

直接アクセスするには、現在のドキュメントを検証するボタンが必要です。メソッドは前のセクションで使用されたメソッドに類似しています。

最初はボタンを与えます:

```
<input type="button" value="Validate" onclick="BtnValidate()">
```

現在のドキュメントを検証するスクリプトを与えます。

```
<SCRIPT ID=Javahandlers LANGUAGE=javascript>
// ---------------------------------------------------------------------
// check validity of current document.
// if validation fails, show validation result in alert box .
function BtnValidate()
{
       // get top-level object of automation interface
       var objApp = objXMLSpyControl.Application;

       // get the active document
```

```
	var objDocument = objApp.ActiveDocument;

	if ( objDocument == null )
		alert( "no active document found" );
	else
	{
		// define as arrays to support their usage as return parameters
		var errorText = new Array(1);
		var errorPos = new Array(1);
		var badData = new Array(1);

		var valid = objDocument.IsValid(errorText, errorPos, badData);

		if (! valid)
		{
			// compose the error description
			var text = errorText;

			// access that XMLData object only if filled in
			if (badData[0] != null)
				text += "(" + badData[0].Name + "/" + badData[0].TextValue
+ ")";


			alert("Validation error[" + errorPos + "]: " + text);
		}
		else
		       alert("Docuent is valid"); }
}
</SCRIPT>
```

29.4.5.2.1.4 カスタムイベントへの接続

サンプルは、プリンシパルを表示するためにXMLSpyControl カスタムイベントのための2つのイベントのコールバックを実装します:

```
<!-- ------------------------------------------------------------ -->
<!--  custom event 'OnDocumentOpened" of XMLSpyControl object  -->
<SCRIPT LANGUAGE="javascript">
	function objXMLSpyControl::OnDocumentOpened( objDocument )
	{
	// alert("Document '" + objDocument.Name + "' opened!");
	}
</SCRIPT>

<!-- ------------------------------------------------------------ -->
<!--  custom event 'OnDocumentClosed" of XMLSpyControl object  -->
<SCRIPT LANGUAGE="javascript">
	function objXMLSpyControl::OnDocumentClosed( objDocument )
	{
	// alert("Document '" + objDocument.Name + "' closed!");
	}
</SCRIPT>
```

29.4.5.2.2 ドキュメントレベルの HTML 統合

このサンプルは、ドキュメントレベルで HTML ページに XMLSpy コントロールを統合する方法について説明しています。次のトピックについて説明されています:

- HTML コード内で XMLSpyControl ActiveX コントロールオブジェクトをインスタンス化します。
- XMLSpy ファイルの編集を許可する XMLSpyControlDocument ActiveX コントロールをインスタンス化します。
- XMLSpyControl プロジェクトウィンドウのための XMLSpyControlPlaceHolder をインスタンス化します。
- XMLSpy ヘルパーウィンドウをホストするために XMLSpyControlPlaceHolder をインスタンス化します。
- XMLSpy 検証出力ウィンドウを表示するために XMLSpyControlPlaceHolder ActiveX コントロールをインスタンス化します。
- 頻繁に使用される XMLSpy コマンドのためのシンプルなカスタマーツールバーを作成します。
- XMLSpy の COM 自動化インターフェイスを使用するボタンを追加します。
- コマンドボタンを更新するためのイベントハンドラーを使用します。

このサンプルは、XMLSpy インストールの `<ApplicationFolder>\Examples\ActiveX\HTML\` フォルダー内のファイル `XMLSpyActiveX_ApplicationLevel.htm` 内で使用することがあります。

29.4.5.2.2.1 XMLSpyControl をインスタンス化する

XMLSpyControl HTML `OBJECT` タグは、XMLSpyControl のインスタンスを作成するために使用されます。クラス ID は XMLSpyControl です。ユーザーインターフェイスのためのコントロール間ジェージャーとして使用される幅と高さは、0 に設定されています。統合レベルは `OBJECT` タグ内でパラメーターとして指定されています。

```
<OBJECT id="objXMLSpyXXMLSpyControl"
        Classid="clsid:a258bba2-3835-4c16-8590-72b44f52c471"
        width="0"
        height="0"
        VIEWASTEXT>
   <PARAM NAME="IntegrationLevel" VALUE="1">
</OBJECT>
```

29.4.5.2.2.2 エディターウィンドウの作成

HTML `OBJECT` タグは、編集ウィンドウを埋め込むために使用されます。

```
<OBJECT id="objDoc1"
        Classid="clsid:DFBB0871-DAFE-4502-BB66-08CEB7DF5255"
        width="600"
        height="500"
        VIEWASTEXT>
</OBJECT>
```

29.4.5.2.2.3 プロジェクトウィンドウの作成

HTML OBJECT タグは、XMLSpyControlPlaceHolder ウィンドウを作成するために使用されます。最初の追加カスタムパラメータは、XMLSpy プロジェクトウィンドウを表示するためのプレースホルダを定義します。2番目のパラメータは、XMLSpy インストールで提供されたサンプルプロジェクトの1つをロードします（以下の場所で見つけることができます: <yourusername>/MyDocuments フォルダ）。

```
<OBJECT id="objProjectWindow"
        Classid="clsid:FDEC3B04-05F2-427d-988C-F03A85DE53C2"
        width="200"
        height="200"
        VIEWASTEXT>
    <PARAM name="PlaceholderWindowID" value="3">
          <PARAM name="FileName" value="Examples/Examples.spp">
</OBJECT>
```

29.4.5.2.2.4 ヘルパーウィンドウのためにプレースホルダを作成する

HTML OBJECT タグは、異なる XMLSpy ヘルパーウィンドウをホストすることのできる XMLSpyControlPlaceHolder ActiveX コントロールをインスタンス化するために使用されます。最初から、ヘルパーウィンドウは表示されません。サンプルファイルを参照してください。

```
<OBJECT id="objEHWindow"
        Classid="clsid:135DEEF4-6DF0-47c2-8F8C-F145F5F3F672"
        width="200"
        height="200"
        VIEWASTEXT>
      <PARAM name="PlaceholderWindowID" value="0">
</OBJECT>
```

3つのボタンにより表示される実際のウィンドウを切り替えることができます。JavaScript 実行ボタンをクリックすることにより、プロパティ PlaceHolderWindowID を定義されている対応する値に設定します。XMLSpyControlPlaceholderWindow[1971].

```
<input type="button" value="EH - Elements" onclick="BtnHelperWindow(0)">
<input type="button" value="EH - Attributes" onclick="BtnHelperWindow(1)">
<input type="button" value="EH - Entities" onclick="BtnHelperWindow(2)">

<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
// ---------------------------------------------------------------------------
// specify which of the windows shall be shown in the placeholder control.
function BtnHelperWindow(i_ePlaceholderWindowID)
{
      objEHWindow.PlaceholderWindowID = i_ePlaceholderWindowID;
}
</SCRIPT>
```

29.4.5.2.2.5 カスタムツールバーの作成

カスタムツールバーは、XMLSpy コマンドのイメージを持つボタンにより構成されています。コマンド ID 番号は、表示されるボタン要素内で見つけることができます コマンドテーブル[1932]。

```
<button id="btnWellFormed" title="Check Well-formedness" onclick="BtnDoCommand(34049)">
       <img src="..\Images\IDC_CHECK_WELL_FORM.gif" width="16" height="16" />
</button>
<button id="btnValidate" title="Validate" onclick="BtnDoCommand(34174)">
       <img src="..\Images\IDC_VALIDATE.gif" width="16" height="16" />
</button>
```

ボタンをクリックすると、対応するコマンド ID がマネジャーコントロールに送信されます。

```
<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
// ---------------------------------------------------------------
// perform any command specified by cmdID.
// command routing includes application, active document and view.
function BtnDoCommand(cmdID)
{
       objXMLSpyX.Exec( cmdID );
       msgtext.innerText = "Command " + cmdID + " performed.";
}
</SCRIPT>
```

29.4.5.2.2.6 ボタンを追加する

サンプルでは、自動化コードを表示するボタンを追加します。

```
<p>
       <input type="button" value="New File" onclick="BtnNewFile(objDoc1)">
       <input type="button" value="Save File" onclick="BtnSaveFile(objDoc1)">
       <input type="text" title="Path" id="strPath" width="150">
       <input type="button" value="Open OrgChart.xml" onclick="BtnOpenFile(objDoc1,
'OrgChart.xml')">
       <input type="button" value="Open OrgChart.xsd" onclick="BtnOpenFile(objDoc1,
'OrgChart.xsd')">
<p>
```

対応する JavaScript 修正は以下のようになります:

```
<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
// ---------------------------------------------------------
// open a document in the specified document control window.
function BtnOpenFile(objDocCtrl, strFileName)
{
       // do not use XMLSpyX.Application.OpenDocument(...) to open a document,
       // since then XMLSpyControl wouldn't know a control window to show
       // the document in. Instead:

       objDocCtrl.Open("C:\Documents and Settings\username\My
Documents\Altova\XMLSpy2021\Examples/" + strFileName);
       objDocCtrl.setActive();

}

// ---------------------------------------------------------------
// open a new empty document in the specified document control window.
function BtnNewFile(objDocCtrl)
{
```

```
	objDocCtrl.Open("");
	objDocCtrl.setActive();
}

// ----------------------------------------------------------------------
// Saves the current file in the specified document control window.
function BtnSaveFile(objDocCtrl)
{
	if(objDocCtrl.Path.length > 0)
		objDocCtrl.SaveDocument();
	else
	{
		if(strPath.value.length > 0)
		{
			objDocCtrl.Path = strPath.value;
			objDocCtrl.Save();
		}
		else
		{
			alert("Please set path for the document first!");
			strPath.focus();
		}
	}

	objDocCtrl.setActive();
}

// ------------------------------------------------------------------------
// check validity of current document.
// if validation fails, show validation result in alert box .
function BtnValidate()
{
	// get top-level object of automation interface
	var objApp = objXMLSpyControl.Application;

	// get the active document
	var objDocument = objApp.ActiveDocument;

	if ( objDocument == null )
		alert( "no active document found" );
	else
	{
		// define as arrays to support their usage as return parameters
		var errorText = new Array(1);
		var errorPos = new Array(1);
		var badData = new Array(1);

		var valid = objDocument.IsValid(errorText, errorPos, badData);

		if (! valid)
		{
			// compose the error description
			var text = errorText;

			// access that XMLData object only if filled in
			if (badData[0] != null)
```

```
                                text += "(" + badData[0].Name + "/" + badData[0].TextValue
+ ")";


                        alert("Validation error[" + errorPos + "]: " + text);
                }
                else
                        alert("Docuent is valid");
        }
}
</SCRIPT>
```

29.4.5.2.2.7 ボタンステータスを更新するためのイベントハンドラーを作成する

コマンドの可用性はマウスクリック、または、キーボード操作により異なる場合があります。XMLSpyControl のカスタムイベント OnUpdateCmdUI は、XMLSpy コマンドに関連するボタンの有効化された/無効化された状態を更新することができます。コマンドの有効化、無効化に関わらず、メソッド XMLSpyControl.QueryStatus[1959] はクエリのために使用されます。

```
<SCRIPT LANGUAGE="javascript">

        function objXMLSpyX::OnUpdateCmdUI()
        {
        if ( document.readyState == "complete" )              // 'complete'
        {
                // update status of buttons
                btnWellFormed.disabled =  ! (objDoc1.QueryStatus(34049) & 0x02);   // not
enabled
                btnValidate.disabled =  ! (objDoc1.QueryStatus(34174) & 0x02);     // not
enabled
        }
        }

        // set activity status of simulated toolbar
</SCRIPT>
```

29.4.5.3 Java

XMLSpy ActiveX コンポーネントに Java コードからアクセスすることができます。Java 統合は下にリストされるライブラリで提供されています。これらのライブラリは、XMLSpy と XMLSpy Integration Package の両方をインストールした後、XMLSpy インストール内のフォルダー <ApplicationFolder>\Examples\JavaAPI で使用することができます(次を参照してください: 必要条件[1905])。

- AltovaAutomation.dll: Altova 自動サーバーのための JNI ラッパー(XMLSpy の 32 ビット インストールの場合)
- AltovaAutomation_x64.dll: Altova 自動サーバーのための JNI ラッパー(XMLSpy の 64 ビット インストールの場合)
- AltovaAutomation.jar: Altova 自動サーバーにアクセスします
- XMLSpyActiveX.jar: XMLSpy ActiveX インターフェイスをラップする Java クラス
- XMLSpyActiveX_JavaDoc.zip: Javadoc インターフェイスのためのヘルプドキュメントを含む Javadoc ファイル

メモ Java ActiveX 統合を使用するには、.dll と .jar ファイルが Java クラス検索パス内に含まれている必要があります。

サンプルJava プロジェクト
サンプルJava プロジェクトは製品のインストールにより提供されます。Java プロジェクト をテストし、使用用途に応じて変更することができます。詳細に関しては、次を参照してください: サンプルJava プロジェクト[1925]。

JavaRules にActiveX コントロール名をマッピングする際のルール
ActiveX コントロールのドキュメントに関しては、次を参照してください: オブジェクトレファレンス[1951]。オブジェクトの名前変換はJava の場合、他の言語と比較して、若干異なります。具体的には、ActiveX コントロールとJava ラッパー間のマッピングのためのルールは、以下のとおりです:

- *クラスとクラス名*
  XMLSpy ActiveX インターフェイスの全てのコンポーネントのために、コンポーネントの名前を持つJava クラスが存在します。

- *メソッド名*
  Javadoc インターフェイス上のメソッド名は、COM インターフェイス上で使用される名前と同じですが、Java 名前変換に準拠する小文字で開始します。COM プロパティにアクセスするには、プロパティ名をプレフィックスするgetとsetから始まるJava メソッド使用することができます。ふプロパティが書き込みをサポートしない場合、setter メソッドを使用することはできません。サンプル: XMLSpyControl の`IntegrationLevel` プロパティ、Java メソッド `getIntegrationLevel` と `setIntegrationLevel` を使用することができます。

- *列挙*
  ActiveX インターフェイス内で定義される全ての列挙のために、Java 列挙は同じ名前と値を使用して定義されます。

- *イベント とイベントハンドラー*
  イベントをサポートする自動化インターフェイス内の全てのインターフェイスのために、同じ名前と'イベント' を使用したJavadoc インターフェイスを使用することができます。単一のイベントのオーバーロードを簡素化するために、全てのイベントのためのデフォルトの実装を使用するJava クラスが提供されます。このJava クラスの名前は、イベント インターフェイスの名前と '`DefaultHandler`' です。例:
  `XMLSpyControl`:アプリケーションにアクセスするためのJava クラス
  `XMLSpyControlEvents`:XMLSpyControl のためのイベント インターフェイス
  `XMLSpyControlEventsDefaultHandler`:`XMLSpyControlEvents` のためのデフォルトのハンドラー

マッピングルールの例外
上記のルールの例外の一部は、以下のとおりです。これらは下にリストされています:

インターフェイス	Java 名
`XMLSpyControlDocument`, method `New`	`newDocument`
`Document`, method `SetEncoding`	`setFileEncoding`
`AuthenticView`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Clone`	`cloneRange`

このセクション
このセクションは、Java コードからアクセスすることのできる基本のXMLSpy ActiveX 機能の一部を紹介しています。次のサブセクションに整理されています:

- サンプルJava プロジェクト[1925]
- ActiveX コントロールの作成[1927]

(C) 2015-2021 Altova GmbH

- コントロール内のデータのロード[1927]
- 基本のイベントハンドリング[1928]
- メニュー[1928]
- UI アップデートイベントハンドリング[1930]
- XML ツリーの作成[1930]

29.4.5.3.1 サンプル Java プロジェクト

XMLSpy インストールパッケージには、アプリケーションフォルダー: `<ApplicationFolder>\Examples\ActiveX\Java\` の ActiveX サンプルフォルダーにあるサンプル Java プロジェクトが含まれています。

Java サンプルは、Java. を使用して作成された共有されるデスクトップアプリケーション内に XMLSpyControl を統合する方法を示しています。

バッチファイル `BuildAndRun.bat` を使用してコマンドラインからテストすることができ、また、Eclipse 内からサンプルプロジェクトをコンパイルして実行することもできます。これらのプロシージャの使用方法を参照してください。

ファイルリスト

Java サンプルフォルダーには、サンプルプロジェクトを作動するために必要な全てのファイルが含まれています。これらのファイルは下にリストされています:

`.classpath`	Eclipse プロジェクトヘルパーファイル
`.project`	Eclipse プロジェクトファイル
`AltovaAutomation.dll`	Java-COM ブリッジ: DLL パート (32-ビットインストール用)
`AltovaAutomation_x64.dll`	Java-COM ブリッジ: DLL パート (64-ビットインストール用)
`AltovaAutomation.jar`	Java-COM ブリッジ: Java ライブラリパート
`BuildAndRun.bat`	コンパイルするバッチファイル、および、コマンドラインプロンプトからサンプルコードを実行します。Java 仮想マシンがパラメータとして存在する箇所にフォルダーが期待されます。
`XMLSpyActiveX.jar`	XMLSpy ActiveX コントロールの Java クラス
`XMLSpyActiveX_JavaDoc.zip`	Java API のためのヘルプドキュメントを含む Javadoc ファイル
`XMLSpyContainer.java`	Java サンプルソースコード
`XMLSpyContainerEventHandler.java`	Java サンプルソースコード
`XMLTreeDialog.java`	Java サンプルソースコード

サンプルの内容

サンプルplaces one XMLSpy ドキュメントエディターウィンドウ、XMLSpy プロジェクトウィンドウ、XMLSpy XPath ウィンドウと XMLSpy 入力ヘルパーin an AWT フレームウィンドウ。XMLSpy のために定義されているファイルメニューを読み取り、同じ構造を持つ AWT メニューを作成します。このメニューまたはプロジェクトを使用して、ドキュメントエディター内でファイルを開き作業することができます。

サンプルを必要に応じて変更することができます.

コードリスト内で次の特定の機能が説明されています:

- ActiveX コントロールの作成[1927]: 自動サーバーとして考えられているXMLSpy を返します。または、作動中の場合、XMLSpy をアクティブ化します。
- コントロール内のデータのロード[1927]: XMLSpy にインストールされているサンプルドキュメントの1つを検索し開きます。
- 基本のイベントハンドリング[1928]: すべての開かれているドキュメントのビューをテキストビューに変えます。コードも開かれているドキュメント内での反復の方法を示しています。
- メニュー[1928]: アクティブなドキュメントを検証し、メッセージボックス内に結果を表示します。コードは出力パラメータの使用方法について説明しています。
- UI アップデートイベントハンドリング[1930]: XMLSpy イベントの処理方法について説明しています。
- XML ツリーの作成[1930]: XML ツリーの作成とモーダルアクティブ化について説明しています。

サンプルフォルダーへのパスの更新

与えられているサンプルを作動する前に、**XMLSpyContainer.java** ファイルを編集します。具体的には、XMLSpy サンプルファイルが使用中のオペレーティングシステム上で保存されている実際のフォルダーを参照する次のパスを確認します:

```
// Locate samples installed with the product.
final String strExamplesFolder = System.getenv( "USERPROFILE" ) + "\\My Documents\
\Altova\\XMLSpy2021\\XMLSpyExamples\\";
```

コマンドラインからサンプルを実行する

コマンドラインからサンプルを作動します:

1. すべての必要条件が満たされていることを確認してください(次を参照してください: 必要条件[1905]).
2. コマンドプロンプトウィンドウを開き、サンプルJava プロジェクト フォルダーに対して現在のディレクトリを変更し、次を入力します:

   ```
   buildAndRun.bat "<Path-to-the-Java-bin-folder>"
   ```

3. 「Enter」を押します。

XMLSpyContainer.java 内のJava ソースがコンパイルされ、実行されます。

Eclipse 内でサンプルをコンパイルし実行する

サンプルJava プロジェクトをEclipse にインポートする

1. すべての必要条件が満たされていることを確認してください(次を参照してください: 必要条件[1905]).
2. ファイル メニューから、「インポート」をクリックします。
3. 「ワークスペースで既存のプロジェクトを開く」を選択し、次の場所にあるEclipse プロジェクトファイルを参照します: <ApplicationFolder>¥Examples¥ActiveX¥Java¥。このフォルダーでは書き込みのアクセスがないため、インポートダイアログボックス上の「ワークプレースにプロジェクトをコピーする」チェックボックスを選択することが奨励されます。

サンプルアプリケーションを作動するには、Package Explorer 内のプロジェクトを右クリックし、コマンド「実行 | Java アプリケーション」を選択します。

コード内のコメントを使用するとJava API クラスのためのヘルプを使用することができ、とEclipse 内でJavadoc ビューを有効化するには、メニューコマンド「Window | ビューの表示 | JavaDoc」を選択してください。

(C) 2015-2021 Altova GmbH

29.4.5.3.2 ActiveX コントロールの作成

下のコードリストは、ActiveX コントロールの作成方法を示しています。コンストラクターは、Java ラッパーオブジェクトを作成します。カンバスにより派生したオブジェクトをパネル、または、フレームに追加すると、ラップされた ActiveX の生成がトリガーされます。

```
01  /**
02  * XMLSpy manager control - always needed
03  */
04    public static XMLSpyControl         xmlSpyControl = null;
05
06  /**
07  * XMLSpy document editing control
08  */
09    public static XMLSpyControlDocument     xmlSpyDocument = null;
10
11  /**
12  * Tool windows - XMLSpy place-holder controls
13  */
14    private static XMLSpyControlPlaceHolder   xmlSpyProjectToolWindow = null;
15    private static XMLSpyControlPlaceHolder   xmlSpyXpathToolWindow = null;
16    private static XMLSpyControlPlaceHolder   xmlSpyEHAttributeToolWindow = null;
17
18  // Create the XMLSpy ActiveX control; the parameter determines that we want
19  // to place document controls and place-holder controls individually.
20  // It gives us full control over the menu, as well.
21     xmlSpyControl = new XMLSpyControl(
       ICActiveXIntegrationLevel.ICActiveXIntegrationOnDocumentLevel.getValue() );
22     xmlSpyDocument = new XMLSpyControlDocument();
23     xmlSpyDocument.setPreferredSize( new Dimension ( 640, 480 ) );
24
25  // Create a project window and open the sample project in it
26     xmlSpyProjectToolWindow = new XMLSpyControlPlaceHolder(
27     XMLSpyControlPlaceholderWindow.XMLSpyControlProjectWindowToolWnd.getValue() );
28     xmlSpyProjectToolWindow.setPreferredSize( new Dimension( 200, 200 ) );
29     xmlSpyXpathToolWindow = new XMLSpyControlPlaceHolder(
       XMLSpyControlPlaceholderWindow.XMLSpyControlXPathDialogToolWnd.getValue() );
30     xmlSpyEHAttributeToolWindow = new XMLSpyControlPlaceHolder(
       XMLSpyControlPlaceholderWindow.XMLSpyControlEntryHelperTopToolWnd.getValue() );
31
32     frame.add( xmlSpyControl, BorderLayout.NORTH );
33     frame.add( xmlSpyDocument, BorderLayout.CENTER );
34     southPanel.add( xmlSpyProjectToolWindow );
35     southPanel.add( xmlSpyXpathToolWindow );
36     southPanel.add( xmlSpyEHAttributeToolWindow );
```

29.4.5.3.3 コントロール内のデータのロード

下のコードリストは、ActiveX コントロール内にデータをロードを表示する方法を表示しています。

```
1    // Locate samples installed with the product.
2       final String strExamplesFolder = System.getenv( "USERPROFILE" ) +
        "\\My Documents\\Altova\\XMLSpy2021\\Examples\\";
3       xmlSpyProjectToolWindow.openProject( strExamplesFolder  + "Examples.spp" );
```

29.4.5.3.4 基本のイベントハンドリング

下のコードリストは、基本のイベントの処理方法について表示しています。XMLSpyControl の `open` メソッド、または、メニューまたはプロジェクトツリーを使用してファイルを開こうとすると、`onOpenedOrFocused` イベントは、アタッチ済みのイベントハンドラーに送信されます。このイベントのための基本の処理により、XMLSpyDocumentControl の `open` メソッドを呼び出すことによりファイルを開きます。

```
01       // Open the PXF file when button is pressed
02       btnOpenPxf.addActionListener( new ActionListener() {
03         public void actionPerformed(ActionEvent e) {
04           try {
05             xmlSpyControl.open( strExamplesFolder + "OrgChart.pxf" );
06           } catch (AutomationException e1) {
07             e1.printStackTrace();
08           }
09         }
10       } );
11       public void onOpenedOrFocused( String i_strFileName, boolean i_bOpenWithThisControl,
boolean i_bFileAlreadyOpened ) throws AutomationException
12   {
13     // Handle the New/Open events coming from the Project tree or from the menus
14     if ( !i_bFileAlreadyOpened )
15     {
16       // This is basically an SDI interface, so open the file in the already existing document
control
17       try {
18         XMLSpyContainer.xmlSpyDocument.open( i_strFileName );
19         XMLSpyContainer.xmlSpyDocument.requestFocusInWindow();
20       } catch (Exception e) {
21         e.printStackTrace();
22       }
23     }
24   }
```

29.4.5.3.5 メニュー

下のコードリストは、メニューアイテムの作成方法を表示しています。 各 `XMLSpyCommand` オブジェクトは、コマンドのID へのセットされている `ActionCommand` を持つ、対応する `MenuItem` オブジェクト、を取得します。全てのメニューアイテムにより生成されているアクションは、(閉じられたメカニズムの再度解釈するなどの) 特定の処理を行う同じ関数により処理されます。または、`exec` メソッドを呼び出すことにより `XMLSpyControl` オブジェクトに実行を委任することができます。メニューの作成中にフィルされる `menuMap` オブジェクトは、後で使用されます (次のセクションを参照してください: UI アップデートイベントハンドリング 1930).

```
01  // Load the file menu when the button is pressed
02       btnMenu.addActionListener( new ActionListener() {
03         public void actionPerformed(ActionEvent e) {
04           try {
05             // Create the menubar that will be attached to the frame
06             MenuBar mb = new MenuBar();
07             // Load the main menu's first item - the File menu
08             XMLSpyCommand xmlSpyMenu = xmlSpyControl.getMainMenu().getSubCommands().getItem( 0 );
09             // Create Java menu items from the commands objects
10             Menu fileMenu = new Menu();
11             handlerobject.fillMenu( fileMenu, xmlSpyMenu.getSubCommands() );
12             fileMenu.setLabel( xmlSpyMenu.getLabel().replace( "&", "" ) );
13             mb.add( fileMenu );
14             frame.setMenuBar( mb );
```

```
15              frame.validate();
16            } catch (AutomationException e1) {
17              e1.printStackTrace();
18            }
19            // Disable the button when the action has been performed
20            ((AbstractButton) e.getSource()).setEnabled( false );
21          }
22        } ) ;
23 /** * Populates a menu with the commands and submenus contained in an XMLSpyCommands object  */
24        public void fillMenu(Menu newMenu, XMLSpyCommands xmlSpyMenu) throws AutomationException
25    {
26      // For each command/submenu in the xmlSpyMenu
27      for ( int i = 0 ; i < xmlSpyMenu.getCount() ; ++i  )
28      {
29        XMLSpyCommand xmlSpyCommand = xmlSpyMenu.getItem( i );
30        if ( xmlSpyCommand.getIsSeparator() )
31          newMenu.addSeparator();
32        else
33        {
34          XMLSpyCommands subCommands = xmlSpyCommand.getSubCommands();
35          // Is it a command (leaf), or a submenu?
36          if ( subcommands.isNull() || subCommands.getCount() == 0 )
37          {
38            // Command -> add it to the menu, set its ActionCommand to its ID and store it in the
menuMap
39            MenuItem mi = new MenuItem( xmlSpyCommand.getLabel().replace( "&", "" ) );
40            mi.setActionCommand( "" + xmlSpyCommand.getID() );
41            mi.addActionListener( this );
42            newMenu.add( mi );
43            menuMap.put( xmlSpyCommand.getID(), mi );
44          }
45          else
46          {
47            // Submenu -> create submenu and repeat recursively
48            Menu newSubMenu = new Menu();
49            fillMenu( newSubMenu, subCommands );
50            newSubMenu.setLabel( xmlSpyCommand.getLabel().replace( "&", "" ) );
51            newMenu.add( newSubMenu );
52          }
53        }
54      }
55    }
56
57     /**
58     * Action handler for the menu items
59     * Called when the user selects a menu item; the item's action command corresponds to the
command table for XMLSpy
60     */
61    public void actionPerformed( ActionEvent e )
62    {
63      try
64      {
65        int iCmd = Integer.parseInt( e.getActionCommand() );
66        // Handle explicitly the Close commands
67        switch ( iCmd )
68        {
69          case 57602:       // Close
70          case 34050:       // Close All
71            XMLSpyContainer.initXmlSpyDocument();
72            break;
73          default:
74            XMLSpyContainer.xmlSpyControl.exec( iCmd );
75            break;
76        }
77      }
```

```
78      catch ( Exception ex )
79      {
80        ex.printStackTrace();
81      }
82
83    }
```

29.4.5.3.6 UI アップデートイベントハンドリング

下のコードリストは、UI-Update イベントハンドラーの作成方法を表示しています。

```
01 /**
02    * Call-back from the XMLSpyControl.
03    * Called to enable/disable commands
04    */
05   @Override
06   public void onUpdateCmdUI() throws AutomationException
07   {
08     // A command should be enabled if the result of queryStatus contains the Supported (1) and
Enabled (2) flags
09     for ( java.util.Map.Entry<Integer, MenuItem> pair : menuMap.entrySet() )
10       pair.getValue().setEnabled( XMLSpyContainer.xmlSpyControl.queryStatus( pair.getKey() ) >
2 );
11   }
12    /**
13    * Call-back from the XMLSpyControl.
14    * Usually called while enabling/disabling commands due to UI updates
15    */
16   @Override
17   public boolean onIsActiveEditor( String i_strFilePath ) throws AutomationException
18   {
19     try {
20       return
XMLSpyContainer.xmlSpyDocument.getDocument().getFullName().equalsIgnoreCase( i_strFilePath );
21     } catch ( Exception e ) {
22       return false;
23     }
24   }
```

29.4.5.3.7 XML ツリーの作成

下のリストは、XML データオブジェクトをツリー内にノードとしてロードします。

```
01 // access required XMLSpy Java-COM classes
02 import com.altova.automation.XMLSpy.XMLData;
03
04 // access AWT and Swing components
05 import java.awt.*;
06 import javax.swing.*;
07 import javax.swing.tree.*;
08
09 /**
10  * A simple example of a tree control loading the structure from an XMLData object.
11  * The class receives an XMLData object, loads its nodes in a JTree, and prepares
12  * for modal activation.
13  *
```

```
14  * Feel free to modify and extend this sample.
15  *
16  * @author Altova GmbH
17  */
18 class XMLTreeDialog extends JDialog
19 {
20   /**
21    * The tree control
22    */
23   private JTree myTree;
24
25   /**
26    * Root node of the tree control
27    */
28   private DefaultMutableTreeNode top ;
29
30   /**
31    * Constructor that prepares the modal dialog containing the filled tree control
32    * @param xml   The data to be displayed in the tree
33    * @param parent  Parent frame
34    */
35   public XMLTreeDialog( XMLData xml, Frame parent )
36   {
37     // Construct the modal dialog
38     super( parent, "XML tree", true );
39     // Arrange controls in the dialog
40     top = new DefaultMutableTreeNode("root");
41     myTree = new JTree(top);
42     setContentPane( new JScrollPane( myTree ) );
43     // Build up the tree
44     fillTree( top, xml );
45     myTree.expandRow( 0 );
46   }
47
48   /**
49    * Loads the nodes of an XML element under a given tree node
50    * @param node  Target tree node
51    * @param elem  Source XML element
52    */
53   private void fillTree( DefaultMutableTreeNode node, XMLData elem)
54   {
55     try
56     {
57       // There are several ways to iterate through child elements: either using the
getFirstChild/getNextChild,
58       // or by incrementing an index up to countChildren and calling getChild [as shown below].
59       // If you only want to get children of one kind, you should use
countChildrenKind/getChildKind,
60       // or provide a kind to the getFirstChild before iterating with the getNextChild.
61       int nSize = elem.countChildren() ;
62       for ( int i = 0 ; i < nSize ; ++i)
63       {
64         // Create a new tree node for each child element, and continue recursively
65         XMLData newElem = elem.getChild(i) ;
66         DefaultMutableTreeNode newNode = new DefaultMutableTreeNode( newElem.getName() ) ;
67         node.add( newNode ) ;
68         fillTree( newNode, newElem ) ;
69       }
70     }
71     catch (Exception e)
72     {
73       e.printStackTrace();
74     }
75   }
```

```
76
77 }
```

29.4.6 コマンド レファレンス

このセクションは、XMLSpy 内で使用することのできる全てのメニューコマンドの名前と識別子をリストしています。各サブセクションは XMLSpy. コマンドテーブルの対応するトップレベルメニューからのコマンドが以下のとおりに整理されています:

- 「メニューアイテム」列は、コマンドのメニューテキストを XMLSpy で表示される順番に表示します。コマンドの機能を簡単に識別することができます。
- 「コマンド名」列は、XMLSpy インストールディレクトリの **ActiveX\Images** フォルダーから同じ名前を持つアイコンを取得するために使用される文字列を指定します。
- 「ID」列は、このコマンドを実行または クエリするメソッドの引数として与えられる列の数値識別子を表示します。

コマンドを実行するには、XMLSpyControl.Exec[1958] または XMLSpyControlDocument.Exec[1964] メソッドを利用します。コマンドのステータスをクエリするには、XMLSpyControl.QueryStatus[1959] または XMLSpyControlDocument.QueryStatus[1965] メソッドを使用します。

インストールされた XMLSpy のエディションにより、コマンドの一部がサポートされない場合があります。

29.4.6.1 「ファイル」メニュー

「ファイル」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
新規作成...	ID_FILE_NEW	57600
開く...	ID_FILE_OPEN	57601
再ロード	IDC_FILE_RELOAD	34065
エンコード	IDC_ENCODING	34061
閉じる	ID_FILE_CLOSE	57602
全て閉じる	IDC_CLOSE_ALL	34050
アクティブなファイル以外閉じる	IDC_CLOSE_OTHERS	34271
保存	ID_FILE_SAVE	57603
名前を付けて保存	ID_FILE_SAVE_AS	57604
全て保存する	ID_FILE_SAVE_ALL	34208
電子メールで送信する	ID_FILE_SEND_MAIL	57612
印刷	ID_FILE_PRINT	57607
印刷プレビュー	IDC_PRINT_PREVIEW	34104

メニューアイテム	コマンド名	ID
印刷設定	ID_FILE_PRINT_SETUP	57606
最近使ったファイル	ID_FILE_MRU_FILE1	57616
終了	ID_APP_EXIT	57665

29.4.6.2 「編集」メニュー

「編集」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
元に戻す	ID_EDIT_UNDO	57643
やり直し	ID_EDIT_REDO	57644
切り取り	ID_EDIT_CUT	57635
コピー	ID_EDIT_COPY	57634
貼り付け	ID_EDIT_PASTE	57637
削除	ID_EDIT_CLEAR	57632
XML テキストとしてコピー	IDC_COPY_AS_XML_TEXT	33443
構成されたテキストとしてコピー	IDC_COPY_AS_STRUCTURED_TEXT	33442
XPath のコピー	IDC_COPY_XPATH	33444
XPointer のコピー	IDC_COPY_XPOINTER	33445
ファイルパス...	IDC_EDIT_INSERT_PATH_STRING	34013
XInclude...	IDC_EDIT_INSERT_XINCLUDE_STRING	34017
エンコードされた外部ファイル	IDC_EDIT_INSERT_ENCODED_BINARY_STRING	34273
整形出力	IDC_PRETTY_PRINT	34101
空白スペースを取り去る	IDC_STRIP_WHITESPACES	34296
全て選択	ID_EDIT_SELECT_ALL	57642
検索	ID_EDIT_FIND	57636
次を検索	ID_EDIT_REPEAT	57640
置換	ID_EDIT_REPLACE	57641
ファイル名の検索...	IDC_FIND_IN_FILES	34000

メニューアイテム	コマンド名	ID
ブックマークの挿入/削除	IDC_TOGGLE_BOOKMARK	34162
全てのブックマークの削除	IDC_REMOVEALLBOOKMARKS	34132
次のブックマーク移動	IDC_GOTONEXTBOOKMARK	34070
前のブックマークへ移動	IDC_GOTOPREVBOOKMARK	34071
コメント イン/アウト	IDC_TOGGLE_XML_COMMENT	34029

29.4.6.3 「プロジェクト」メニュー

「プロジェクト」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
新規のプロジェクト	IDC_ICPROJECTGUI_NEW	37200
プロジェクトを開く	IDC_ICPROJECTGUI_OPEN	37201
プロジェクトを再ロードする	IDC_ICPROJECTGUI_RELOAD	37202
プロジェクトを閉じる	IDC_ICPROJECTGUI_CLOSE	37203
プロジェクトを保存する	IDC_ICPROJECTGUI_SAVE	37204
プロジェクトに名前を付けて保存する	IDC_ICPROJECTGUI_SAVE_AS	37207
ソース管理を有効化する	ID_SCC_ENABLE	38602
プロジェクトにファイルを追加する	IDC_ICPROJECTGUI_ADD_FILES_TO_PROJECT	37205
グローバルリソースをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_GLOBAL_RESOURCE_TO_PROJECT	37239
URL をプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_URL_TO_PROJECT	37206
アクティブなファイルをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_ACTIVE_FILE_TO_PROJECT	37208
アクティブならびに関係するファイルをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_ACTIVE_AND_RELATED_FILES_TO_PROJECT	37209
プロジェクト フォルダーをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_FOLDER_TO_PROJECT	37210
外部フォルダーをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_EXT_FOLDER_TO_PROJECT	37211

メニューアイテム	コマンド名	ID
外部 W eb フォルダーをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_EXT_URL_FOLDER_TO_PROJECT	37212
スクリプト設定...	IDC_PROJECT_SCRIPT_SETTINGS	34136
プロパティ...	IDC_ICPROJECTGUI_PROJECT_PROPERTIES	37223
最近使ったプロジェクト	IDC_ICPROJECTGUI_RECENT	37224

29.4.6.4 「XML」メニュー

「XML」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
属性	IDC_INSERT_ATTRIBUTE	33449
要素	IDC_INSERT_STRUCT	33459
テキスト	IDC_INSERT_TEXT	33460
CDATA	IDC_INSERT_CDATA	33450
コメント	IDC_INSERT_COMMENT	33451
XML	IDC_INSERT_XML	33461
処理命令	IDC_INSERT_PI	33458
XInclude...	IDC_INSERT_XINCLUDE	34019
DOCTYPE	IDC_INSERT_DEF_DOCTYPE	33453
ExternalID	IDC_INSERT_DEF_EXTERNAL_ID	33456
ELEMENT	IDC_INSERT_DEF_ELEMENT	33454
ATTLIST	IDC_INSERT_DEF_ATTLIST	33452
ENTITY	IDC_INSERT_DEF_ENTITY	33455
NOTATION	IDC_INSERT_DEF_NOTATION	33457
エンコードされた外部ファイル	IDC_INSERT_ENCODED_BINARY	34274
属性	IDC_APPEND_ATTRIBUTE	33415
要素	IDC_APPEND_STRUCT	33425
テキスト	IDC_APPEND_TEXT	33426

メニューアイテム	コマンド名	ID
CDATA	IDC_APPEND_CDATA	33416
コメント	IDC_APPEND_COMMENT	33417
XML	IDC_APPEND_XML	33427
処理命令	IDC_APPEND_PI	33424
XInclude...	IDC_APPEND_XINCLUDE	34026
DOCTYPE	IDC_APPEND_DEF_DOCTYPE	33419
ExternalID	IDC_APPEND_DEF_EXTERNAL_ID	33422
ELEMENT	IDC_APPEND_DEF_ELEMENT	33420
ATTLIST	IDC_APPEND_DEF_ATTLIST	33418
ENTITY	IDC_APPEND_DEF_ENTITY	33421
NOTATION	IDC_APPEND_DEF_NOTATION	33423
エンコードされた外部ファイル	IDC_APPEND_ENCODED_BINARY	34276
属性	IDC_ADD_CHILD_ATTRIBUTE	33402
要素	IDC_ADD_CHILD_STRUCT	33412
テキスト	IDC_ADD_CHILD_TEXT	33413
CDATA	IDC_ADD_CHILD_CDATA	33403
コメント	IDC_ADD_CHILD_COMMENT	33404
XML	IDC_ADD_CHILD_XML	33414
処理命令	IDC_ADD_CHILD_PI	33411
XInclude...	IDC_ADD_CHILD_XINCLUDE	34027
DOCTYPE	IDC_ADD_CHILD_DEF_DOCTYPE	33406
ExternalID	IDC_ADD_CHILD_DEF_EXTERNAL_ID	33409
ELEMENT	IDC_ADD_CHILD_DEF_ELEMENT	33407
ATTLIST	IDC_ADD_CHILD_DEF_ATTLIST	33405
ENTITY	IDC_ADD_CHILD_DEF_ENTITY	33408
NOTATION	IDC_ADD_CHILD_DEF_NOTATION	33410
エンコードされた外部ファイル	IDC_ADD_CHILD_ENCODED_BINARY	34277
属性	IDC_CONVERT_TO_ATTRIBUTE	33429

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
要素	IDC_CONVERT_TO_STRUCT	33439
テキスト	IDC_CONVERT_TO_TEXT	33440
CDATA	IDC_CONVERT_TO_CDATA	33430
コメント	IDC_CONVERT_TO_COMMENT	33431
XML	IDC_CONVERT_TO_XML	33441
処理命令	IDC_CONVERT_TO_PI	33438
DOCTYPE	IDC_CONVERT_TO_DEF_DOCTYPE	33433
ExternalID	IDC_CONVERT_TO_DEF_EXTERNAL_ID	33436
ELEMENT	IDC_CONVERT_TO_DEF_ELEMENT	33434
ATTLIST	IDC_CONVERT_TO_DEF_ATTLIST	33432
ENTITY	IDC_CONVERT_TO_DEF_ENTITY	33435
NOTATION	IDC_CONVERT_TO_DEF_NOTATION	33437
テーブルとして表示	IDC_GRID_VIEW_AS_TABLE	34075
行の挿入	IDC_TABLE_INSERT_ROW	34158
行の追加	IDC_TABLE_APPEND_ROW	34157
昇順の並べ替えと	IDC_TABLE_SORT_ASC	33464
降順の並べ替え	IDC_TABLE_SORT_DESC	33465
左に移動	IDC_MOVE_LEFT	34091
右に移動	IDC_MOVE_RIGHT	34092
要素の中へ	IDC_ENCLOSE_IN_ELEMENT	33446
XPath の検証...	IDC_EVALUATE_XPATH	34007
整形式のチェック	IDC_CHECK_WELL_FORM	34049
XML の検証	IDC_VALIDATE	32954
サーバー上で XML の検証(ハイパフォーマンス)	IDC_VALIDATE_RAPTOR	34309
入力ヘルパーの更新	IDC_UPDATE_ELEMENT_CHOICE	34173
名前スペースプレフィックス...	IDC_NAMESPACE	33462
XML 署名の作成...	IDC_XML_SIGNATURE_CREATE	34280
XML 署名の検証...	IDC_XML_SIGNATURE_VERIFY	34281

29.4.6.5 「DTD/スキーマ」メニュー

「DTD/スキーマ」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
DTD の割り当て...	IDC_ASSIGN_DTD	34032
スキーマの割り当て...	IDC_ASSIGN_SCHEMA	34033
他のDTD を取り込み...	IDC_INCLUDE_DTD	34084
DTD へ移動	IDC_GOTO_DTD	34072
スキーマへ移動	IDC_GOTO_SCHEMA	34074
定義へ移動	IDC_GOTO_DEFINITION	33447
DTD/スキーマの生成...	IDC_GENERATE_DTD_SCHEMA	34068
DTD のフラット化...	IDC_FLATTEN_DTD	34301
DTD をスキーマに変換...	IDC_CONVERT_DTD_TO_SCHEMA	34299
スキーマのフラット化...	IDC_FLATTEN_SCHEMA	34302
スキーマをDTD に変換...	IDC_CONVERT_SCHEMA_TO_DTD	34300
UML へ変換...	IDC_CONVERT_SCHEMA_TO_UML	34008
MapForce でDB、Excel、EDI からXML を生成...	IDC_DTD_OPENIN_MAPFORCE	34056
StyleVision でHTML/PDF/W ord 出力をデザイン...	IDC_DTD_OPENIN_STYLEVISION	34057
サンプルXML/JSON ファイルの生成...	IDC_GENERATE_XML_FROM_SCHEMA	34069
プログラムコードの生成...	IDC_GENERATE_CODE_FROM_SCHEMA	34067
メモリキャッシュの破棄	IDC_FLUSH_CACHED_FILES	34066

29.4.6.6 「スキーマ設計」メニュー

「スキーマ設計」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
スキーマ設定...	IDC_SCHEMA_NAMESPACES	33571
図の保存...	IDC_SCHEMA_SAVE_DIAGRAM	33581
ドキュメントの生成	IDC_SCHEMA_DOCUMENTATION	34146

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
表示設定...	IDC_SCHEMA_VIEW_CONFIG	33593
ズーム..	IDC_SCHEMA_ZOOM	34150
全てのグローバルを表示	IDC_SCHEMA_MODE_GLOBALS	34147
図を表示	IDC_SCHEMA_MODE_DIAGRAM	33570
Oracle スキーマ拡張子を有効にする	IDC_SCHEMA_ORACLE_EXTENSIONS	33577
Oracle スキーマ設定...	IDC_SCHEMA_ORACLE_SCHEMA_SETTINGS	33578
SQL Server スキーマ拡張子を有効にする	IDC_SCHEMA_SQLSERVER_EXTENSIONS	33588
名前付きスキーマリレーションシップ...	IDC_SCHEMA_SQLSERVER_GLOBAL_RELATIONSHIPS	33589
名前なしスキーマリレーションシップ...	IDC_SCHEMA_SQLSERVER_LOCAL_RELATIONSHIPS	33590
SchemaAgent Server に接続する..	IDC_SCHEMA_SCHEMAAGENT_SERVER_CONNECT	33582
SchemaAgent Server から切断	IDC_SCHEMA_SCHEMAAGENT_SERVER_DISCONNECT	33583
ファイルのみ	IDC_SCHEMAAGENT_SHOW_FILE_ONLY	33504
直接関連付けられているスキーマファイルも表示	IDC_SCHEMAAGENT_SHOW_WITH_DIRECTLY_REFERENCED_SCHEMAS	33608
直接参照しているスキーマファイルも表示	IDC_SCHEMAAGENT_SHOW_WITH_DIRECTLY_REFERENCING_SCHEMAS	33602
直接関連しているスキーマファイルも表示	IDC_SCHEMAAGENT_SHOW_WITH_DIRECTLY_RELATED_SCHEMAS	33613
SchemaAgent 検証..	IDC_SCHEMA_EXTVALID_MENU	33539
スキーマサブセットの作成...	IDC_SCHEMA_CREATE_SUBSET	33650
スキーマのフラット化...	IDC_SCHEMA_FLATTEN	33651

29.4.6.7 「XSL/XQuery」メニュー

「XSL/XQuery」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
XSL 変換	IDC_TRANSFORM_XSL	33006

メニューアイテム	コマンド名	ID
XSL スピードオプティマイザー	IDC_TRANSFORM_XSLPBO	34306
XSL-FO 変換	IDC_TRANSFORM_XSLFO	33007
XSL パラメーター/XQuery 変数...	IDC_TRANSFORM_XSL_PARAMS	33008
XQuery/Update 実行	IDC_TRANSFORM_XQUERY	34170
バックマッピングの有効化	IDC_ENABLE_BACKMAPPING	34364
XSLT/XQuery プロファイリングの有効化...	IDC_PROFILING_OPTIONS	34105
XSL の割り当て...	IDC_ASSIGN_XSL	33001
XSL-FO の割り当て...	IDC_ASSIGN_XSLFO	33002
サンプルXML ファイルの割り当て...	IDC_ASSIGN_SAMPLE_XML	33000
XSL へ移動	IDC_GOTO_XSL	33004
デバッガーの開始 / 移動	ID_PROCESS_XSL	34212
デバッガーの停止	ID_XSLT_DEBUGGER_STOP	33017
デバッガーの再起動	ID_XSLT_DEBUGGER_RESTART	33013
デバッグセッションの終了	ID_XSLT_DEBUGGER_END_SESSION	33011
ステップイン	ID_XSLT_DEBUGGER_STEP	33014
ステップアウト	ID_XSLT_DEBUGGER_STEP_OUT	33015
ステップオーバー	ID_XSLT_DEBUGGER_STEP_OVER	33016
現在の実行ノードを表示	ID_XSLT_DEBUGGER_GO_TO_CURRENT_EXECUTION_NODES	33012
ブレークポイントの挿入/削除	IDC_TOGGLE_BREAKPOINT	34246
トレースポイントの挿入/削除	IDC_TOGGLE_TRACEPOINT	34248
ブレークポイントの有効化/無効化	IDC_ENABLE_BREAKPOINT	34245
トレースポイントの有効化/無効化	IDC_ENABLE_TRACEPOINT	34247
ブレークポイント/トレースポイント...	ID_XSLTDEBUGGER_BREAKPOINTS	33009
コールスタック	ID_XSL_DEBUGWINDOWS_CALLSTACK	34238
XPath ウォッチ	ID_XSL_DEBUGWINDOWS_WATCH	34244
コンテキスト	ID_XSL_DEBUGWINDOWS_CONTEXT	34239
変数	ID_XSL_DEBUGWINDOWS_VARIABLE	34243

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
メッセージ	ID_XSL_DEBUGWINDOWS_MESSAGES	34240
テンプレート	ID_XSL_DEBUGWINDOWS_TEMPLATES	34241
情報	ID_XSLXQUERY_DEBUGWINDOWS_INFO	34237
トレース	ID_XSL_DEBUGWINDOWS_TRACES	34242
デバッグ設定...	ID_XSLTDEBUGGER_SETTINGS	33010

29.4.6.8 「Authentic」メニュー

「Authentic」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
新規ドキュメント...	IDC_AUTHENTIC_NEW_FILE	34036
データベースデータの編集...	IDC_AUTHENTIC_EDIT_DB	34035
StyleVision スタイルシートの割り当て...	IDC_ASSIGN_SPS	34034
StyleVision スタイルシートの編集	IDC_EDIT_SPS	34060
XML データの新たな行を選択し編集	IDC_CHANGE_WORKING_DB_XML_CELL	32861
XML 署名...	IDC_AUTHENTICGUI_XMLSIGNATURE	32862
XML エンティティを定義	IDC_DEFINE_ENTITIES	32805
マークアップを隠す	IDC_MARKUP_HIDE	32855
小さなマークアップの表示	IDC_MARKUP_SMALL	32858
大きなマークアップの表示	IDC_MARKUP_LARGE	32856
混合マークアップの表示	IDC_MARKUP_MIXED	32857
太字	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEBOLD	32813
イタリック	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEITALIC	32814
下線	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEUNDERLINE	32815
取り消し線	IDC_AUTHENTICGUI_RICHEDIT_TOGGLESTRIKETHROUGH	32816
前景色	IDC_AUTHENTICGUI_RICHEDIT_COLOR_FOREGROUND	32824

メニューアイテム	コマンド名	ID
背景色	IDC_AUTHENTICGUI_RICHEDIT_COLOR_BACKGROUND	32830
左寄せ	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_LEFT	32818
中央揃え	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_CENTER	32819
右寄せ	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_RIGHT	32820
行の追加	IDC_ROW_APPEND	32806
行の挿入	IDC_ROW_INSERT	32809
行の複製	IDC_ROW_DUPLICATE	32808
行を上に移動	IDC_ROW_MOVE_UP	32811
行を下に移動	IDC_ROW_MOVE_DOWN	32810
行の削除	IDC_ROW_DELETE	32807
HTML ドキュメントの生成	IDC_PXF_GENERATE_HTML	34283
RTF ドキュメントの生成	IDC_PXF_GENERATE_RTF	34284
PDF ドキュメントの生成	IDC_PXF_GENERATE_PDF	34285
W ord 2007+ ドキュメントの生成	IDC_PXF_GENERATE_DOCX	34286
信頼できる場所	IDC_TRUSTED_LOCATIONS	34288

29.4.6.9 「データベース」メニュー

「DB」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
データベースのクエリ	IDC_QUERYDATABASE	34012
XML スキーマの管理	IDC_DB_MANAGESCHEMAS	34014
XML スキーマの割り当て	IDC_DB_CHOOSEVALIDATIONSCHEMA	34016
XML スキーマの管理	IDC_DB_MANAGESCHEMAS	34014
XML スキーマの管理	IDC_DB_MANAGESCHEMAS	34014
Oracle XML ドキュメントの参照	ID_CONVERT_ORACLEXMLDB_BROWSE	34205

(C) 2015-2021 Altova GmbH

29.4.6.10 「変換」メニュー

「変換」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
テキストファイルをインポート...	IDC_IMPORT_TEXT	34082
データベースのデータをインポート...	IDC_IMPORT_DATABASE	34080
Microsoft W ord 文書をインポート...	IDC_IMPORT_WORD	34083
データベース構造からXML スキーマを作成	IDC_CREATE_DB_SCHEMA	34054
XML スキーマを基にデータベースをインポート	IDC_IMPORT_DB_SCHEMA	34081
XML スキーマを基にデータベース構造を作成	IDC_CREATE_DB_BASED_ON_SCHEMA	34053
テキストファイルにエクスポート...	IDC_EXPORT_TEXTFILE	34064
データベースにエクスポート	IDC_EXPORT_DB	34003
XML インスタンスをJSON にから変換する...	IDC_JSON_CONVERT_TOFROM_XML	34135
XML スキーマを JSON スキーマにから変換する..	IDC_JSON_CONVERT_TOFROM_XSD	34350

29.4.6.11 「表示」メニュー

「表示」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
テキストビュー	IDC_VIEW_TEXT	34180
拡張グリッドビュー	IDC_VIEW_GRID	34178
スキーマデザインビュー	IDC_VIEW_SCHEMA	34179
W SDL デザインビュー	IDC_VIEW_WSDL	34117
XBRL タクソノミビュー	IDC_VIEW_XBRL	34118
Authentic ビュー	IDC_VIEW_CONTENT	34177
ブラウザービュー	IDC_VIEW_BROWSER	34176
展開 +	IDC_SEL_EXPAND	34152
縮退 -	IDC_SEL_COLLAPSE	34151
全て展開	IDC_SEL_EXPAND_ALL	33463

メニューアイテム	コマンド名	ID
未選択を縮退	IDC_COLLAPSE_UNSELECTED	33428
最適な幅	IDC_OPTIMAL_WIDTHS	34099
ワードラップ	IDC_WORD_WRAP	34181
行/文字へ移動	IDC_GOTO_LINE	34073
ファイルへ移動	IDC_GOTO_FILE	33448
テキストビューの設定	IDC_TEXTVIEW_SETTINGS	34119

29.4.6.12 「ブラウザー」メニュー

「ブラウザー」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
戻る	IDC_STEP_BACK	32958
進む	IDC_STEP_FORWARD	32957
停止	IDC_BROWSER_STOP	34047
最新の状態に更新する	IDC_BROWSER_REFRESH	34046
最大	IDC_BROWSER_FONT_LARGEST	34041
大	IDC_BROWSER_FONT_LARGE	34040
中	IDC_BROWSER_FONT_MEDIUM	34042
小	IDC_BROWSER_FONT_SMALL	34043
最小	IDC_BROWSER_FONT_SMALLEST	34044

29.4.6.13 「WSDL」メニュー

「W SDL」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
メッセージの挿入	ID_WSDL_MESSAGES_ADDNEWMESSAGE	33715
メッセージの削除	ID_WSDL_MESSAGES_DELETESELECTED MESSAGE	33717
メッセージパートの追加（パラメーター）	ID_WSDL_MESSAGES_ADDMESSAGEPART	33714

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
メッセージパートの削除（パラメーター）	ID_WSDL_MESSAGES_DELETEMESSAGEPART	33716
要求/応答	IDC_WSDL_OPERATION_APPENDREQUESTRESPONSE	33734
送信請求/応答	IDC_WSDL_OPERATION_APPENDSOLICITRESPONSE	33737
一方向	IDC_WSDL_OPERATION_APPENDONEWAY	33735
通知	IDC_WSDL_OPERATION_APPENDNOTIFICATION	33736
空のオペレーション	ID_WSDL_OPERATIONS_APPENDAOPERATIONTOTHISPORTTYPE	33722
オペレーションの削除	ID_WSDL_OPERATIONS_DELETEOPERATION	33724
input 要素の追加	ID_WSDL_OPERATIONS_ADDINPUTFUNCTION	33719
output 要素の追加	ID_WSDL_OPERATIONS_ADDOUTPUTFUNCTION	33721
fault 要素の追加	ID_WSDL_OPERATIONS_ADDFAULTFUNCTION	33718
input/output/fault 要素の削除	ID_WSDL_OPERATIONS_DELETEINPUTOUTPUTFUNCTION	33723
input/output/fault 要素に新しいメッセージを追加	ID_WSDL_OPERATIONS_ADDNEWMESSAGETOTHISELEMENT	33720
ポートタイプの挿入	ID_WSDL_PORTTYPE_INSERTAPORTTYPE	33727
ポートタイプの削除	ID_WSDL_PORTTYPE_DELETETHISPORTTYPE	33726
バインディングの挿入	ID_WSDL_BINDING_NEWBINDING	33713
バインディングの削除	ID_WSDL_BINDING_DELETEBINDING	33711
soap:body	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPBODY	33706
soap:header	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPHEADER	33708
soap:headerfault	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPHEADERFAULT	33709

メニューアイテム	コマンド名	ID
soap:fault	ID_WSDL_BINDING_APPENDEXTENSIBILITY_SOAPFAULT	33707
mime:content	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMECONTENT	33702
mime:multipartrelated	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMEMULTIPARTRELATED	33704
mime:part	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMEPART	33705
mime:mimeXml	ID_WSDL_BINDING_APPENDEXTENSIBILITY_MIMEMIMEXML	33703
http:urlencoded	ID_WSDL_BINDING_APPENDEXTENSIBILITY_HTTPURLENCODED	33700
http:urlreplacement	ID_WSDL_BINDING_APPENDEXTENSIBILITY_HTTPURLREPLACEMENT	33701
拡張要素の削除	ID_WSDL_BINDING_DELETEEXTESIBILITY	33712
サービスの挿入	ID_WSDL_SERVICE_INSERTSERVICE	33731
サービスの削除	ID_WSDL_SERVICE_DELETETHISSERVICE	33729
ポートの挿入	ID_WSDL_SERVICE_INSERTNEWPORT	33730
ポートの削除	ID_WSDL_SERVICE_DELETETHISPORT	33728
新たなインターフェイスの追加	IDC_WSDL20_ADDINTERFACE	33794
インターフェイスの削除	IDC_WSDL20_DELETEINTERFACE	33795
fault 要素の追加	IDC_WSDL20_ADDINTERFACEFAULT	33796
fault 要素の削除	IDC_WSDL20_DELETEINTERFACEFAULT	33808
In-only	IDC_WSDL20_ADDINTERFACEOPERATION_INONLY	33797
Robust-in-only	IDC_WSDL20_ADDINTERFACEOPERATION_ROBUSTINONLY	33798
In-out	IDC_WSDL20_ADDINTERFACEOPERATION_INOUT	33801
In-opt-out	IDC_WSDL20_ADDINTERFACEOPERATION_INOPTOUT	33802
Out-in	IDC_WSDL20_ADDINTERFACEOPERATION_OUTIN	33803

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
Out-opt-in	IDC_WSDL20_ADDINTERFACEOPERATION_OUTOPTIN	33804
Out-only	IDC_WSDL20_ADDINTERFACEOPERATION_OUTONLY	33800
Robust-out-only	IDC_WSDL20_ADDINTERFACEOPERATION_ROBUSTOUTONLY	33799
空のオペレーション	IDC_WSDL20_ADDINTERFACEOPERATION_EMPTY	33805
オペレーションの削除	IDC_WSDL20_DELETEINTERFACEOPERATION	33809
新たなバインディングの追加	IDC_WSDL20_ADDBINDING	33820
バインディングの削除	IDC_WSDL20_DELETEBINDING	33821
新規のfaultバインディングの追加	IDC_WSDL20_ADDBINDINGFAULT	33822
faultバインディングの削除	IDC_WSDL20_DELETEBINDINGFAULT	33826
新たなオペレーションの追加	IDC_WSDL20_ADDBINDINGOPERATION	33823
オペレーションの削除	IDC_WSDL20_DELETEBINDINGOPERATION	33827
新たなサービスの追加	IDC_WSDL20_ADDSERVICE	33839
サービスの削除	IDC_WSDL20_DELETESERVICE	33840
新たなエンドポイントの追加	IDC_WSDL20_ADDENDPOINT	33841
エンドポイントの削除	IDC_WSDL20_DELETEENDPOINT	33842
新たなスキーマ	ID_WSDL_TYPES_NEWSCHEMA	33733
スキーマの埋め込み	ID_WSDL_TYPES_EMBEDSCHEMA	39456
スキーマの抽出	ID_WSDL_TYPES_EXTRACTSCHEMAS	39459
スキーマビューでのスキーマの編集	ID_WSDL_TYPES_EDITTHISSCHEMA	33732
図の保存...	IDC_WSDL_SAVE_DIAGRAM	39451
ドキュメントの生成	ID_WSDL_GENERATEDOCUMENTATION	39452
WSDLドキュメントの再解析	IDC_WSDL_REPARSE	33774
WSDL 2.0に変換する	IDC_WSDL_CONVERT_TO_WSDL20	39453
MapForceを使用してWSDLコードを生成する...	IDC_WSDL_GENERATE_CODE_MAPFORCE	34122

29.4.6.14 「SOAP」メニュー

「SOAP」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
新規 SOAP リクエストを作成...	ID_SOAP_GENERATESOAPMESSAGE	34224
サーバーにリクエストを送信...	ID_SOAP_SENDREQUESTTOSERVER	34225
SOAP リクエスト設定...	ID_SOAP_SOAPREQUESTSETTINGS	34227
SOAP デバッガーセッション	ID_SOAP_SOAPDEBUGGER	34226
実行	ID_SOAPDEBUGGER_BUTTONPLAY	34221
ステップ実行	ID_SOAPDEBUGGER_SINGLESTEP	34222
次のリクエストでブレーク	ID_SOAPDEBUGGER_BREAKONNEXTREQUEST	34219
次のレスポンスでブレーク	ID_SOAPDEBUGGER_BREAKONNEXTRESPONSE	34220
プロキシサーバーの停止	ID_SOAPDEBUGGER_STOPSERVER	34223
SOAP デバッガーオプション	ID_SOAPDEBUGGEROPTIONS	34218

29.4.6.15 「XBRL」メニュー

「XBRL」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
アークロール...	IDC_XMLSPYXBRLEDITOR_ARCROLES	34114
リンクロール...	IDC_XMLSPYXBRLEDITOR_LINKROLES	34115
名前スペースプレフィックス...	IDC_XMLSPYXBRLEDITOR_NAMESPACES	34116
ターゲット名前空間の設定...	IDC_XMLSPYXBRLEDITOR_SET_TARGETNAMESPACE	34039
パラメータ値...	IDC_ICXBRLEDITOR_PARAMETER_VALUES	38913
インポート/参照...	IDC_XMLSPYXBRLEDITOR_IMPORT_REFERENCE	34137
ID によりコンポーネントを検索...	IDC_ICXBRLEDITOR_FIND_COMPONENT_BY_ID	38893

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
ドキュメントの生成	IDC_XMLSPYXBRLEDITOR_GENERATEDOCUMENTATION	34125
ビューの設定...	IDC_XMLSPYXBRLEDITOR_VIEWSETTINGS	34113
MapForce でDB、Excel、CSV からXBRL を生成...	IDC_XBRL_GENERATE_WITH_MAPFORCE	34045
StyleVision でHTML/PDF/Word としてXBRL を生成...	IDC_XBRL_PRESENT_WITH_STYLEVISION	34121
フォーミュラを実行する..	IDC_XBRL_EXECUTE_FORMULA	34305
フォーミュラをサーバーで実行する(ハイパフォーマンス)...	IDC_XBRL_EXECUTE_FORMULA_RAPTOR	34352
テーブルを生成する	IDC_XBRL_GENERATE_TABLE	34304
サーバーでテーブルを生成する(ハイパフォーマンス)...	IDC_XBRL_GENERATE_TABLE_RAPTOR	34353
インラインXBRL の変換	IDC_IXBRL_TRANSFORM	34354

29.4.6.16 「ツール」メニュー

「ツール」メニューには以下のコマンドが含まれています:

メニューアイテム	コマンド名	ID
スペル..	IDC_SPELL_CHECK	34154
スペルオプション...	IDC_SPELL_OPTIONS	34155
スクリプトエディター...	ID_SCRIPTFORMEDITOR_EDIT_PROJECT	39666
無し	ID_SCRIPTFORMEDITOR_EXECUTE_MACRO_MENU_UPPDATE	39600
開かれているファイルとの比較..	ID_XMLDIFF_CHOOSE_FILES	34235
ディレクトリの比較	ID_XMLDIFF_CHOOSE_DIRECTORIES	34234
比較のオプション	ID_XMLDIFF_SETTINGS	34236
	IDC_TOOLS_ENTRY	34292
グローバルリソース	IDC_GLOBALRESOURCES	37401
	IDC_GLOBALRESOURCES_SUBMENUENTRY1	37408
Raptor Servers の管理..	IDC_VALIDATE_RAPTOR_MANAGER	34311
無し	IDC_VALIDATE_RAPTOR_NOCFG	34326

メニューアイテム	コマンド名	ID
カスタマイズ...	IDC_APP_TOOLS_CUSTOMIZE	32959
オプション...	IDC_SETTINGS	34133
	ID_SCRIPTING_MACROITEMS	34249

29.4.6.17 「ヘルプ」メニュー

「ウィンドウ」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
重ねて表示	ID_WINDOW_CASCADE	57650
上下に並べて表示	ID_WINDOW_TILE_HORZ	57651
左右に並べて表示	ID_WINDOW_TILE_VERT	57652
プロジェクトウィンドウ	IDC_PROJECT_WINDOW	34128
情報ウィンドウ	IDC_INFO_WINDOW	34085
入力ヘルパー	IDC_ENTRY_HELPERS	34062
出力ウィンドウ	IDC_OUTPUT_DIALOGBARS	34004
プロジェクトと入力ヘルパー	IDC_PROJECT_ENTRYHELPERS	34006
全てオン/オフ	IDC_ALL_BARS	34031

29.4.6.18 「ヘルプ」メニュー

「ヘルプ」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
目次	IDC_HELP_CONTENTS	32966
インデックス	IDC_HELP_INDEX	32967
検索	IDC_HELP_SEARCH	32969
キーボードマップ	IDC_HELP_KEYMAPDLG	32968
ライセンスの登録	IDC_ACTIVATION	32970
注文フォーム	IDC_OPEN_ORDER_PAGE	32971
登録	IDC_REGISTRATION	32972

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
最新情報のチェック	IDC_CHECK_FOR_UPDATES	32973
XMLSpy Product Comparison...	IDC_PRODUCT_COMPARISON	32955
サポートセンター	IDC_OPEN_SUPPORT_PAGE	32961
Web のFAQ	IDC_SHOW_FAQ	32962
コンポーネントや無料ツールのダウンロード	IDC_OPEN_COMPONENTS_PAGE	32963
インターネット上のXMLSpy	IDC_OPEN_XML_SPY_HOME	32964
XMLSpy トレーニング	ID_HELP_XMLSPYTRAINING	32965
XMLSpy について	ID_APP_ABOUT	57664

29.4.7 オブジェクトレファレンス

オブジェクト:
XMLSpyCommand 1951
XMLSpyCommands 1953
XMLSpyControl 1954
XMLSpyControlDocument 1962
XMLSpyControlPlaceHolder 1968

標準のXMLSpy 機能へのアクセスを与えるには、**XMLSpy 自動化インターフェイス** のオブジェクトにもアクセスすることができます。詳細に関しては、次を参照してください: XMLSpyControl.Application 1955, XMLSpyControlDocument.Document 1963 と XMLSpyControlPlaceHolder.Project 1969 。

29.4.7.1 XMLSpyCommand

プロパティ:
ID 1952
Label 1952
Name 1953
IsSeparator 1952
ToolTip 1953
StatusText 1953
Accelerator 1952
SubCommands 1953

詳細:

コマンドオブジェクトは次のオプションであることができます: 実行可能なコマンド, コマンド コンテナー(例えば、メニュー、サブメニュー、または、ツールバー)、または、メニューセパレーター。現在の`Command` オブジェクト内に保管される情報の種類を決定するには、`ID`、`IsSeparator`、と`SubCommands` プロパティをクエリしてください。

コマンドオブジェクト	使用可能な場合...
実行可能なコマンド	• ID はゼロよりも大きい場合 • IsSeparator がfalse の場合 • SubCommands が空白の場合
コマンド コンテナー	• ID はゼロの場合 • IsSeparator がtrue の場合 • SubCommands がCommand オブジェクトを含む場合
セパレーター	• ID はゼロの場合 • IsSeparator がtrue の場合

29.4.7.1.1 Accelerator

プロパティ: Accelerator を string として

詳細:
コマンドのために定義されたアクセラレータキーを返します。コマンドにアクセラレーターキーが割り当てられていない場合、このプロパティは、空の文字列を返します。 アクセラレータキーの文字列の表示は次のフォーマットを取ります:

```
[ALT+][CTRL+][SHIFT+]key
```

Windows プラットフォームSDK 関数 GetKeyNameText を使用して key が変換された箇所。

29.4.7.1.2 ID

プロパティ: ID を long として

詳細:
このプロパティは、コマンドの一意の識別子を取得します。コマンドのID は(Exec[1958] を使用して) コマンドを実行するために、または、(QueryStatus[1959] を使用して)ステータスをクエリするために必須です。コマンドが他のコマンド(例えば、トップレベルのメニュー)のためのコンテナー、または、セパレーターの場合、ID は0 です。

29.4.7.1.3 IsSeparator

プロパティ: IsSeparator を boolean として

詳細:
コマンドオブジェクトがメニューセパレーターの場合、の場合、プロパティは、 true を返します。それ以外の場合は、 false を返します。次も参照してください: コマンド[1951].

29.4.7.1.4 Label

プロパティ: Label を string として

詳細:

このプロパティは、XMLSpy のグラフィカルなユーザーインターフェイス内に表示されるとおりのコマンドのテキストを取得します。コマンドがセパレーターの場合、「ラベル」は、空の文字列です。このプロパティは、関連した GUI テキストを持たない、一部のツールバーコマンドのために空の文字列を返します。

29.4.7.1.5 Name

プロパティ: `Name` を string として

詳細:
このプロパティは一意のコマンド名を取得します。使用できる箇所で、この値はコマンドのアイコンファイルを取得するために使用できます。使用することのできるアイコンファイルは、使用中の XMLSpy インストールのフォルダー **<ApplicationFolder>\Examples\ActiveX\Images** 内にあります。

29.4.7.1.6 StatusText

プロパティ: `Label` を string として

詳細:
コマンドが選択されると、ステータステキストは、XMLSpy のステータスバー内で表示されるテキストです。他のコマンドのセパレーターまたは、コンテナオブジェクトではないコマンドオブジェクトに対してのみ適用することができます。それ以外の場合、プロパティは空の文字列です。

29.4.7.1.7 SubCommands

プロパティ: `SubCommands` を `Commands` 1953 として

詳細:
`SubCommands` プロパティは、現在のコマンドのサブコマンドである `Command` 1951 オブジェクトのコレクションを取得します。プロパティは、他のコマンド(メニュー、サブメニュー、または、ツールバー)のためのコンテナーであるコマンドにのみ適用されます。このようなコンテナーコマンドは、0 に設定された `ID` を持ち、および、`false` に設定された `IsSeparator` プロパティを持ちます。

29.4.7.1.8 ToolTip

プロパティ: `ToolTip` を string として

詳細:
このプロパティは、各コマンドのためのヒントを表示するテキストを取得します。コマンドがツールのヒントのテキストを持たない場合、プロパティは、空の文字列を返します。

29.4.7.2 XMLSpyCommands

プロパティ:
`Count` 1954
`Item` 1954

詳細:
XMLSpyControl のコマンドレベルと ID へのアクセスを取得するための `Command` 1951 オブジェクトのコレクションです。これらのコマンドは、`Exec` 1958 メソッドを使用して実行することができます。また、ステータスを `QueryStatus` 1959 を使用してクエリすることができます。

29.4.7.2.1 Count

プロパティ: Count を long として

詳細:
このコレクションのレベル上の Command[1951] オブジェクトの数。

29.4.7.2.2 Item

プロパティ: Item (n を long として) を Command[1951] として

詳細:
このコレクション内のインデックス n を持つコマンドを取得します。インデックスは 1 ベースです。

29.4.7.3 XMLSpyControl

プロパティ:
IntegrationLevel[1956]
Appearance[1955]
Application[1955]
BorderStyle[1955]
CommandsList[1956]
EnableUserPrompts[1956]
MainMenu[1957]
Toolbars[1957]

メソッド:
Open[1958]
Exec[1958]
QueryStatus[1959]

イベント:
OnUpdateCmdUI[1961]
OnOpenedOrFocused[1961]
OnCloseEditingWindow[1959]
OnFileChangedAlert[1960]
OnContextChanged[1960]
OnDocumentOpened[1960]
OnValidationWindowUpdated[1961]

このオブジェクトは完全な ActiveX コントロールで、XMLSpy ライブラリがアプリケーションレベルモードで使用されている場合、表示されます。

```
CLSID: a258bba2-3835-4c16-8590-72b44f52c471
ProgID: Altova.XMLSpyControl
```

29.4.7.3.1 プロパティ

以下のプロパティが定義されています:

IntegrationLevel 1956
EnableUserPrompts 1956
Appearance 1955
BorderStyle 1955

コマンドに関連したプロパティ:
CommandsList 1956
MainMenu 1957
Toolbars 1957

XMLSpyAPI へのアクセス:
Application 1955

29.4.7.3.1.1 Appearance

プロパティ: Appearance を short として

ディスパッチ ID : -520

詳細:
0 に対して等しくない値は、コントロールの周りでクライアントエッジを表示します。デフォルトの値は 0 です。

29.4.7.3.1.2 Application

プロパティ: Application を Application として

ディスパッチ ID : 1

詳細:
Application プロパティは、完全な XMLSpy 自動サーバーAPI の Application オブジェクトへのアクセスを与えます。プロパティは読み取り専用です

29.4.7.3.1.3 BorderStyle

プロパティ: BorderStyle を short として

ディスパッチ ID : -504

詳細:

1 の値は、薄いボーダーを持つコントロールを表示します。デフォルトの値は 0 です。

29.4.7.3.1.4 CommandsList

プロパティ: CommandList を Commands[1953] として（読み取り専用）

ディスパッチ ID : 1004

詳細:
このプロパティは、XMLSpyControl を持つ定義された全てのコマンドのフラットリストを返します。メニュー構造に従い、コマンドを整理するには、MainMenu[1957] を使用します。ツールバーコマンドを取得するには、Toolbars[1957] を使用します。

```
public void GetAllXmlSpyCommands()
{
    // Get all commands from the XMLSpy ActiveX control assigned to the current form
    XMLSpyControlLib.XMLSpyCommands commands = this.axXMLSpyControl1.CommandList;
    // Iterate through all commands
    for (int i = 0; i < commands.Count; i++)
    {
      // Get each command by index and output it to the console
      XMLSpyControlLib.XMLSpyCommand cmd = axXMLSpyControl1.CommandList[i];
      Console.WriteLine("{0} {1} {2}", cmd.ID, cmd.Name, cmd.Label.Replace("&", ""));
    }
}
```

C# サンプル

29.4.7.3.1.5 EnableUserPrompts

プロパティ: EnableUserPrompts を boolean として

ディスパッチ ID : 1006

詳細:
このプロパティを *false* に設定すると、コントロール内のユーザープロンプトを無効化します。デフォルトの値は *true* です。

29.4.7.3.1.6 IntegrationLevel

プロパティ: IntegrationLevel を ICActiveXIntegrationLevel[1971] として

ディスパッチ ID : 1000

詳細:
The IntegrationLevel プロパティ determines コントロールのオペレーションモード。次も参照してください: アプリケーションレベルの統合[1903] とドキュメントレベルの統合[1910] 。

メモ XMLSpyControl オブジェクトの作成後すぐに、このプロパティを設定することは重要です。

29.4.7.3.1.7 MainMenu

プロパティ: MainMenu を Command[1951] として(読み取り専用)

ディスパッチ ID : 1003

詳細:
このプロパティは、 XMLSpyControl メインメニュー内で使用することのできる構造の情報とコマンドを Command オブジェクトとして提供します。オブジェクトは、 XMLSpy の使用することのできる全てのサブメニューを含んでいます(例「ファイル」、「編集」、「表示」など)。サブメニューオブジェクトにアクセスするには、MainMenu プロパティの SubCommands プロパティを使用します。各サブメニューは Command オブジェクトでもあります。各サブメニューのために、対応する子コマンドセパレーターを取得するために、 SubCommands プロパティを使用して更に反復することができます(このテクニックが使用される可能性があります。例えば、アプリケーションメニューをプログラム的に作成するなど)。メニューコマンドの一部は、 SubCommands プロパティを持つ他のメニューコマンドのためのコンテナー(「親」)としての役割を果たします。全てのメニューコマンドの構造をプログラム的に取得するには、.再帰的な関数を作成する必要があります。

```
public void GetXmlSpyMenus()
{
    // Get the main menu from the XMLSpy ActiveX control assigned to the current form
    XMLSpyControlLib.XMLSpyCommand mainMenu = this.axXMLSpyControl1.MainMenu;

    // Loop through entries of the main menu (e.g. File, Edit, etc.)
    for (int i = 0; i < mainMenu.SubCommands.Count; i++)
    {
      XMLSpyControlLib.XMLSpyCommand menu = mainMenu.SubCommands[i];
      Console.WriteLine("{0} menu has {1} children items (including separators)",
menu.Label.Replace("&", ""), menu.SubCommands.Count);
    }
}
```

C# サンプル

29.4.7.3.1.8 Toolbars

プロパティ: Toolbars を Commands[1953] として (読み取り専用)

ディスパッチ ID : 1005

詳細:
このプロパティは、XMLSpyControl ツールバーの構造の情報 を Command オブジェクトとして与えます。 Command オブジェクトには、XMLSpy の使用することのできる全てのツールバーが含まれています。ツールバーにアクセスするには、 Toolbars プロパティの SubCommands プロパティを使用します。各ツールバーは、 Command オブジェクトでもあります。各ツールバーのために、コマンドを使用するために SubCommands プロパティを更に反復します(例えば、アプリケーションのツールバープログラム的に作成するためなどにこのテクニックは使用される場合があります)。

```
public void GetXmlSpyToolbars()
{
    // Get the application toolbars from the StyleVision ActiveX control assigned to
the current form
```

```
    XMLSpyControlLib.XMLSpyCommands toolbars = this.axXMLSpyControl1.Toolbars;

    // Iterate through all toolbars
    for (int i = 0; i < toolbars.Count; i++)
    {
      XMLSpyControlLib.XMLSpyCommand toolbar = toolbars[i];
      Console.WriteLine();
      Console.WriteLine("The toolbar \"{0}\" has the following commands:",
toolbar.Label);

      // Iterate through all commands of this toolbar
      for (int j = 0; j < toolbar.SubCommands.Count; j++)
      {
         XMLSpyControlLib.XMLSpyCommand cmd = toolbar.SubCommands[j];
         // Output only command objects that are not separators
         if (!cmd.IsSeparator)
         {
            Console.WriteLine("{0}, {1}, {2}", cmd.ID, cmd.Name, cmd.Label.Replace("&",
 ""));
         }
      }
    }
}
```

C# サンプル

29.4.7.3.2 メソッド

以下のメソッドが定義されます:

Open[1958]
Exec[1958]
QueryStatus[1959]

29.4.7.3.2.1 Exec

Method: Exec (nCmdID を long として) を boolean として

ディスパッチ ID : 6

詳細:
このメソッドは、ID nCmdID を持つXMLSpy コマンドを呼び出し、コマンドが実行可能な場合、メソッドは、true を返します。使用可能な全てのコマンドのリストを取得するには、CommandsList[1956] を使用してください。.コマンドの状態を取得するには、QueryStatus[1959] を使用してください。

29.4.7.3.2.2 Open

Method: Open (strFilePath を string として) を boolean として

ディスパッチ ID : 5

詳細:

メソッドの結果は、引数 strFilePath 内でパスされる拡張子により異なります。ファイル拡張子が .sps の場合、新規ドキュメントが開かれます。ファイル拡張子が .svp の場合、対応するプロジェクトが開かれます。異なるファイル拡張子がメソッドにパスされると、コントロールはファイルを新規のコンポーネントとしてアクティブなドキュメントにロードします。

ドキュメントレベルの統合モード内でコントロール使用する際、このメソッドを使用して、ドキュメントまたはプロジェクトをロードしないでください。代わりに、XMLSpyControlDocument.Open[1965] とXMLSpyControlPlaceHolder.OpenProject[1970] を使用してください。

29.4.7.3.2.3 QueryStatus

Method: QueryStatus (nCmdID を long として) を long として

ディスパッチ ID : 7

詳細:
QueryStatus は、nCmdID により指定されたコマンドのチェックされた/チェックが解除されたステータスを返します。このステータスはビットマスクとして返されます。

ビット	値	名前	意味
0	1	サポートされている	コマンドがサポートされている場合設定します。
1	2	有効化されている	コマンドが有効化されている場合設定します(実行可能)。
2	4	チェックされている	コマンドがチェックされている場合設定します。

これは、QueryStatus が 0 コマンドを返すと、ID は、有効な XMLSpy コマンドとして認識されないことを意味します。QueryStatus が 1 または 5 の値を返すとコマンドは無効化されます。

29.4.7.3.3 Events

XMLSpyControl ActiveX コントロールは次の接続ポイントイベントを提供します:

OnUpdateCmdUI[1961]
OnOpenedOrFocused[1961]
OnCloseEditingWindow[1959]
OnFileChangedAlert[1960]
OnContextChanged[1960]

OnDocumentOpened[1960]
OnValidationWindowUpdated[1961]

29.4.7.3.3.1 OnCloseEditingWindow

イベント : OnCloseEditingWindow (i_strFilePath を String として) を boolean として

ディスパッチ ID : 1002

詳細:
XMLSpy が既に開かれているドキュメントを閉じる場合、このイベントはトリガーされます。このイベントへの応答として、クライアントは *i_strFilePath* に関連するエディターウィンドウを閉じます。このイベントが *true* を返すと、クライアントがドキュメントを閉じたことを指します。

特定の処理が必要とされておらず、XMLSpyControl がエディターを閉じようと試み、関連したドキュメントコントロールを破棄する場合、クライアント は、*false* を返すことができます。

29.4.7.3.3.2 OnContextChanged

イベント: OnContextChanged (i_strContextName を String として、i_bActive を bool として) を bool として

ディスパッチ ID: *1004*

詳細:
このイベントは XMLSpy が次のオペレーションコンテキストを有効化、または、無効化するとトリガーされます:

- XSLT プロファイル – "XSLTProfiling" がコンテキストメニューとして渡されます。
- XSLT / XQuery デバッグ – "DebuggingXSLT" がコンテキストメニューとして渡されます。
- SOAP デバッグ – "DebuggingSOAP" がコンテキストメニューとして渡されます。(Enterprise エディションのみ)

29.4.7.3.3.3 OnDocumentOpened

イベント: OnDocumentOpened (objDocument を Document として)

ディスパッチ ID: *1*

詳細:
このイベントはドキュメントが開かれる都度トリガーされます。引数 objDocument は、XMLSpy 自動化インターフェイス、からの Document オブジェクトであり、ドキュメントの詳細をクエリするために使用することができ、または、追加オペレーションを行います。ドキュメントレベルで統合される場合、イベント XMLSpyControlDocument.OnDocumentOpened[1967] を代わりに使用することが奨励されます。

29.4.7.3.3.4 OnFileChangedAlert

イベント: OnFileChangedAlert (i_strFilePath を String として) を bool として

ディスパッチ ID: *1001*

詳細:
XMLSpyControl を使用してファイルがロードされると、isこのイベントはトリガーされます。他のアプリケーションによりハードディスク上で変更されます。イベントを処理すると、クライアントは、true を返します。または、XMLSpy が通常の方法で処理すると、false を返します。例えば、ユーザーに再ロードを促すなど。

29.4.7.3.3.5 OnLicenseProblem

イベント: OnLicenseProblem (i_strLicenseProblemText を String として)

ディスパッチ ID: *1005*

詳細:
XMLSpyControl がこのコントロールのための有効なライセンスを使用できないことを検知すると、このイベントはトリガーされます。制限されているユーザーファイセンスの場合、コントロールが初期化されると発生する可能性があります。インテグレーターはこのイベントをしようして、この

コントロールの機能へのアクセスを無効化することができます。このイベントから戻ると、コントロールは機能へのアクセスをブロックします（例、コントロール内の空のウィンドウを表示し、リクエストに対してエラーを返すなど）。

29.4.7.3.3.6 OnOpenedOrFocused

イベント： `OnOpenedOrFocused (i_strFilePath` を String として、`i_bOpenWithThisControl` を bool として)

ディスパッチ ID： *1000*

詳細:
アプリケーションレベルで統合する場合、このイベントはクライアントにドキュメントが開かれたこと、またはXMLSpy によりアクティブに設定されたことを通知します。

ドキュメントレベルで統合する場合、このイベントは、クライアントにドキュメントウィンドウを ファイル `i_strFilePath` で開くように命令します。ファイルが既に開かれている場合、対応するドキュメントウィンドウをアクティブなウィンドウにしてください。

`i_bOpenWithThisControl` が `true` の場合、内部アクセスが必要とされるため、ドキュメントは、XMLSpyControl により開かれる必要があります。それ以外の場合、ファイルは、異なるエディターで開くことができます。

29.4.7.3.3.7 OnToolWindowUpdated

イベント： `OnToolWindowUpdated(pToolWnd` を long として)

ディスパッチ ID： *1006*

詳細:
ツールウィンドウが更新されると、このイベントはトリガーされます。

29.4.7.3.3.8 OnUpdateCmdUI

イベント： `OnUpdateCmdUI ()`

ディスパッチ ID： *1003*

詳細:
頻繁に呼び出され、`XMLSpyControl.QueryStatus` 1959 を使用してXMLSpy コマンドの状況をチェックする良い機会をインテグレーターに与えます。このコールバック内で long オペレーションを実行しないでください。

29.4.7.3.3.9 OnValidationWindowUpdated

イベント： `OnValidationWindowUpdated ()`

ディスパッチ ID： *3*

詳細:
新しい情報と共に検証出力ウィンドウが更新されると、このイベントはトリガーされます。

29.4.7.4 XMLSpyControlDocument

プロパティ:
Appearance 1963
BorderStyle 1963
Document 1963
IsModified 1963
Path 1963
ReadOnly 1964

メソッド:
Exec 1964
New 1965
Open 1965
QueryStatus 1965
Reload 1965
Save 1966
SaveAs 1966

イベント:
OnDocumentOpened 1967
OnDocumentClosed 1967
OnModifiedFlagChanged 1968
OnContextChanged 1967
OnFileChangedAlert 1968
OnActivate 1966

XMLSpyControl がドキュメントレベルモード内で統合されている場合、各ドキュメントは、型 `XMLSpyControlDocument` のオブジェクト内で表示されています。`XMLSpyControlDocument` には、1度に1つのドキュメントのみが含まれていますが、場合によっては異なるファイルを表示するために使用される場合があります。

このオブジェクトは完全な ActiveX コントロールです。

```
CLSID: 52A552E6-2AB8-4e3e-B545-BE998233DDA0
ProgID: Altova.XMLSpyControlDocument
```

29.4.7.4.1 プロパティ

以下のプロパティが定義されています:

ReadOnly 1964
IsModified 1963
Path 1963
Appearance 1963
BorderStyle 1963

XMLSpyAPI へのアクセス:
Document 1963

(C) 2015-2021 Altova GmbH

29.4.7.4.1.1 Appearance

プロパティ: `Appearance` を short として

ディスパッチ ID : -520

詳細:
`0` に対して等しくない値は、ドキュメント コントロールの周りにクライアント エッジを表示します。デフォルトの値は `0` です。

29.4.7.4.1.2 BorderStyle

プロパティ: `BorderStyle` を short として

ディスパッチ ID : -504

詳細:
`1` の値は、薄いボーダー持つコントロールを表示します。デフォルトの値は `0` です。

29.4.7.4.1.3 Document

プロパティ: `Document` をドキュメントとして

ディスパッチ ID : 1

詳細:
`Document` プロパティは、XMLSpy 自動サーバー API の `Document` オブジェクト へのアクセスを与えます。このインターフェイスは、コントロール内にロードされるドキュメントと使用することのできる追加機能を与えます。プロパティは読み取り専用です

29.4.7.4.1.4 IsModified

プロパティ: `IsModified` を boolean として(読み取り専用)

ディスパッチ ID : 1006

詳細:
ドキュメント コンテンツが最後に開かれてから変更された場合、`IsModified` は *true* を返します。再ロードし、オペレーションを保存します。それ以外の場合は、`false` を返します。

29.4.7.4.1.5 Path

プロパティ: `Path` を string として

ディスパッチ ID: 1005

詳細:
コントロール内にロードされたドキュメントのフルパス名に設定、または、取得します。

29.4.7.4.1.6 ReadOnly

プロパティ: `ReadOnly` を boolean として

ディスパッチ ID: 1007

詳細:
このプロパティを使用すると、ドキュメントの読み取り専用モードをオンとオフに切り替えることができます。`ReadOnly` が `true` の場合、変更はできないことに注意してください。

29.4.7.4.2 メソッド

以下のメソッドが定義されています:

ドキュメントハンドリング:
New[1965]
Open[1965]
Reload[1965]
Save[1966]
SaveAs[1966]

コマンドハンドリング::
Exec[1964]
QueryStatus[1965]

29.4.7.4.2.1 Exec

メソッド: `Exec` (`nCmdID` を long として) を boolean として

ディスパッチ ID: 8

詳細:
このメソッドは、ID `nCmdID` を持つXMLSpy コマンドを呼び出し、コマンドが実行可能な場合、メソッドは、`true` を返します。現在アクティブなドキュメントがアプリケーションで使用できる場合のみ、このメソッドを呼び出すことができます。

メニュー構造に従いコマンドを整理するには、XMLSpyControl の `MainMenu`[1957] プロパティを使用します。ツールバーコマンドを取得するには、XMLSpyControl の `Toolbars`[1957] プロパティを使用します。

29.4.7.4.2.2 New

メソッド: New () を boolean として

ディスパッチ ID : 1000

詳細:
このメソッドはコントロール内で新規ドキュメントを初期化します。

29.4.7.4.2.3 Open

メソッド: Open (strFileName を string として) を boolean として

ディスパッチ ID : 1001

詳細:
Open は、ファイル strFileName を新規のドキュメントとして、コントロールにロードします。

29.4.7.4.2.4 QueryStatus

メソッド: QueryStatus (nCmdID を long として) を long として
ディスパッチ ID : 9

詳細:
QueryStatus は、nCmdID により指定されたコマンドのチェックされた/チェックが解除されたステータスを返します。このステータスはビットマスクとして返されます。

ビット	値	名前	意味
0	1	サポートされている	コマンドがサポートされている場合設定します。
1	2	有効化されている	コマンドが有効化されている場合設定します (実行可能)。
2	4	チェックされている	コマンドがチェックされている場合設定します。

これは、QueryStatus が 0 コマンドを返すと、ID は、有効な XMLSpy コマンドとして認識されないことを意味します。
QueryStatus が 1 または 5 の値を返すとコマンドは無効化されます。アプリケーション内で現在アクティブなドキュメントが存在する場合、クライアントは、ドキュメントコントロールの QueryStatus メソッドを呼び出します。

29.4.7.4.2.5 Reload

メソッド: Reload () を boolean として

ディスパッチ ID : 1002

詳細:
Reload ファイルシステムからドキュメントコンテンツを更新する。

29.4.7.4.2.6 Save

メソッド: Save () を boolean として

ディスパッチ ID: *1003*

詳細:
Save は、現在のドキュメントをロケーション Path[1963] に保存する。

29.4.7.4.2.7 SaveAs

メソッド: SaveAs (strFileName を string として) を boolean として

ディスパッチ ID: *1004*

詳細:
SaveAs は Path[1963] を *strFileName* に設定し、ドキュメントをこの場所に保存します。

29.4.7.4.3 Events

XMLSpyControlDocument ActiveX コントロールは次の接続ポイントイベントを提供します:

OnDocumentOpened[1967]
OnDocumentClosed[1967]
OnModifiedFlagChanged[1968]
OnContextChanged[1967]
OnFileChangedAlert[1968]
OnActivate[1966]
OnSetEditorTitle[1968]

29.4.7.4.3.1 OnActivate

イベント: OnActivate ()

ディスパッチ ID: *1005*

詳細:
ドキュメントコントロールが有効化されると、フォーカスのある、ユーザー入力として使用することのできるこのイベントがトリガーされます。

(C) 2015-2021 Altova GmbH

29.4.7.4.3.2 OnContextChanged

イベント: `OnContextChanged` (`i_strContextName` を String として、`i_bActive` を bool として) を bool として

ディスパッチ ID: *1004*

詳細:
異なる XMLSpy ビュー内でドキュメントが表示されると、このイベントはトリガーされます。次の値がパスされます:

- グリッドビュー - `"View_0"` がコンテキストメニューとしてパスされます。
- テキストビュー - `"View_1"` がコンテキストメニューとしてパスされます。
- ブラウザービュー - `"View_2"` がコンテキストメニューとしてパスされます。
- スキーマビュー - `"View_3"` がコンテキストメニューとしてパスされます。
- Authentic ビュー - `"View_4"` がコンテキストメニューとしてパスされます。
- W SDL ビュー - `"View_5"` がコンテキストメニューとしてパスされます。

29.4.7.4.3.3 OnDocumentClosed

イベント: `OnDocumentClosed` (`objDocument` を Document として)

ディスパッチ ID: *1001*

詳細:
このイベントはドキュメントがこのコントロールにロードされると閉じられます。引数 `objDocument` は、XMLSpy 自動化インターフェイスからの `Document` オブジェクトです、で慎重に使用してください。

29.4.7.4.3.4 OnDocumentOpened

イベント: `OnDocumentOpened` (`objDocument` を Document として)

ディスパッチ ID: *1000*

詳細:
このイベントは、このコントロール内でドキュメントが開かれるとトリガーされます。引数 `objDocument` は、XMLSpy 自動化インターフェイスからの `Document` オブジェクトです。ドキュメントの詳細をクエリするために使用することができ、または、追加オペレーションを行います。

29.4.7.4.3.5 OnDocumentSaveAs

イベント: `OnContextDocumentSaveAs` (`i_strFileName` を String として)

ディスパッチ ID: *1007*

詳細:
このドキュメントが新しい名前で内部で保存される場合このイベントがトリガーされます。

29.4.7.4.3.6 OnFileChangedAlert

イベント: OnFileChangedAlert () を bool として

ディスパッチ ID: *1003*

詳細:
このドキュメントにロードされると、このイベントはトリガーされます。コントロールはハードディスク上で他のアプリケーションに変更されます。イベントを処理すると、クライアントは、true を返します。または、XMLSpy が通常の方法で処理すると、false を返します。例えば、ユーザーに再ロードを促すなど。

29.4.7.4.3.7 OnModifiedFlagChanged

イベント: OnModifiedFlagChanged (i_bIsModified を boolean として)

ディスパッチ ID: *1002*

詳細:
このイベントはドキュメントの状態を、変更された、および、変更されていない状態に切り替えると、トリガーされます。ドキュメントコンテンツが元のコンテンツと異なる場合、パラメーター *i_bIsModifed* is *true* それ以外の場合は、false を返します。

29.4.7.4.3.8 OnSetEditorTitle

イベント: OnSetEditorTitle ()

ディスパッチ ID: *1006*

詳細:
このイベントは、含まれているドキュメントが内部で名前を与えられると挙げられます。

29.4.7.5 XMLSpyControlPlaceHolder

すべてのプロジェクトプレースホルダーウィンドウに使用することのできるプロパティ:
PlaceholderWindowID 1969

プロジェクトプレースホルダーウィンドウのためのプロパティ:
Project 1969

プロジェクトプレースホルダーウィンドウのためのメソッド:
OpenProject 1970
CloseProject 1970

XMLSpyControlPlaceHolder コントロールは、概要、ライブラリまたはプロジェクトウィンドウなどの追加の XMLSpy ウィンドウを表示するために使用されます。他の ActiveX コントロールと同様に使用することができ、クライアントアプリケーション内に配置することができます。

CLSID: 135DEEF4-6DF0-47c2-8F8C-F145F5F3F672

ProgID: Altova.XMLSpyControlPlaceHolder

29.4.7.5.1 プロパティ

以下のプロパティが定義されています:

PlaceholderWindowID[1969]

XMLSpyAPI へのアクセス:
Project[1969]

29.4.7.5.1.1 Label

プロパティ: Label を String (読み取り専用)として

ディスパッチ ID: *1001*

詳細:
このプロパティはプレースホルダのタイトルへのアクセスを与えます。プロパティは読み取り専用です

29.4.7.5.1.2 PlaceholderWindowID

プロパティ: PlaceholderWindowID を XMLSpyControlPlaceholderWindow[1971] として

ディスパッチ ID: *1*

詳細:
このプロパティを使用すると、コントロールのクライアントエリア内で表示される XMLSpy ウィンドウが通知されます。PlaceholderWindowID は、XMLSpyControlPlaceholderWindow[1971] 列挙の有効な値に設定することができます。コントロールはすぐに状態を変更し、すぐに新規の XMLSpy ウィンドウを表示します。

29.4.7.5.1.3 Project

プロパティ: Project Project として(読み取り専用)

ディスパッチ ID: *2*

詳細:
Project プロパティは、XMLSpy 自動サーバー API の Project オブジェクト へのアクセスを与えます。このインターフェイスは、コントロールへロードされるプロジェクトに使用される追加機能を与えます。プレースホルダウィンドウが XMLSpyXProjectWindow (=3) の値を持つ PlaceholderWindowID[1969] の場合のみ、プロパティは有効なプロジェクトインターフェイスを返します。プロパティは読み取り専用です

29.4.7.5.2 メソッド

以下のメソッドが定義されています:

OpenProject[1970]
CloseProject[1970]

29.4.7.5.2.1 OpenProject

メソッド: OpenProject (strFileName を string として) を boolean として

ディスパッチ ID : 3

詳細:
OpenProject は strFileName を新規のプロジェクトとしてコントロールにロードします。プレースホルダウィンドウが XMLSpyXProjectWindow (=3) と異なる PlaceholderWindowID[1969] を持つ場合、メソッドは失敗します。

29.4.7.5.2.2 CloseProject

メソッド: CloseProject ()

ディスパッチ ID : 4

詳細:
CloseProject はプロジェクトにロードされたコントロールを閉じます。プレースホルダウィンドウが XMLSpyXProjectWindow (=3) と異なる PlaceholderWindowID[1969] を持つ場合、メソッドは失敗します。

29.4.7.5.3 イベント

XMLSpyControlPlaceholder ActiveX コントロールは次の接続ポイントイベントを提供します:

OnModifiedFlagChanged[1970]

29.4.7.5.3.1 OnModifiedFlagChanged

イベント: OnModifiedFlagChanged (i_bIsModified を boolean として)

ディスパッチ ID : 1

詳細:
このイベントは XMLSpyXProjectWindow (=3) の PlaceholderWindowID[1969] を持つプレースホルダコントロールのためのみにトリガーされます。プロジェクトのコンテンツが変更された、または、変更されていない状態の間で変更があった場合、イベントは実行されます。プロジェクトコンテンツが元のコンテンツと異なる場合、パラメーター *i_bIsModifed* は *true* を返し、それ以外の場合は、false を返します。

29.4.7.5.3.2 OnSetLabel

イベント: `OnSetLabel(i_strNewLabel` を string として)

ディスパッチ ID: *1000*

詳細:
プレースホルダウィンドウのタイトルが変更されると、挙げられます。

29.4.7.6 列挙

以下の列挙が定義されています:

ICActiveXIntegrationLevel[1971]
XMLSpyControlPlaceholderWindow[1971]

29.4.7.6.1 ICActiveXIntegrationLevel

XMLSpyControl の IntegrationLevel[1956] プロパティのための可能な値。

```
ICActiveXIntegrationOnApplicationLevel  = 0
ICActiveXIntegrationOnDocumentLevel     = 1
```

29.4.7.6.2 XMLSpyControlPlaceholderWindow

この列挙にはサポートされる追加 XMLSpy ウィンドウのリストが含まれています。

```
XMLSpyControlNoToolWnd                          = -1
XMLSpyControlEntryHelperTopToolWnd              = 0
XMLSpyControlEntryHelperMiddleToolWnd           = 1
XMLSpyControlEntryHelperBottomToolWnd           = 2
XMLSpyControlValidatorOutputToolWnd             = 3
XMLSpyControlProjectWindowToolWnd               = 4
XMLSpyControlXSLTDebuggerContextToolWnd         = 5
XMLSpyControlXSLTDebuggerCallstackToolWnd       = 6
XMLSpyControlXSLTDebuggerVariableToolWnd        = 7
XMLSpyControlXSLTDebuggerWatchToolWnd           = 8
XMLSpyControlXSLTDebuggerTemplateToolWnd        = 9
XMLSpyControlXSLTDebuggerInfoToolWnd            = 10
XMLSpyControlXSLTDebuggerMessageToolWnd         = 11
XMLSpyControlXSLTDebuggerTraceToolWnd           = 12
XMLSpyControlSOAPDebuggerToolWnd                = 13
XMLSpyControlXPathProfilerListToolWnd           = 14
XMLSpyControlXPathProfilerTreeToolWnd           = 15
XMLSpyControlXPathDialogToolWnd                 = 16
XMLSpyControlDBQueryManagerToolWnd              = 17
```

```
XMLSpyControlInfoToolWnd                    = 18
XMLSpyControlXSLOutlineToolWnd              = 19
XMLSpyControlSchemaFindToolWnd              = 20
XMLSpyControlXBRLFindToolWnd                = 21
XMLSpyControlChartsToolWnd                  = 22
```


(C) 2015-2021 Altova GmbH

30 付録

以下の付録には、XMLSpy に関する技術的な情報や、ライセンスに関する重要な情報が収められています。各付録には以下のようにサブセクションが収められています:

エンジン情報 1974

- XSLT とXQuery エンジン情報 1974
- XSLT とXQuery 拡張関数 1980

データ型、DB とXML スキーマ間の変換

- DB からXML スキーマへの変換 2081
- XML スキーマからDB への変換 2087

技術情報 2094

- OS ならびにメモリの必要条件 2094
- Altova XML パーサー 2094
- Altova XSLT ならびにXQuery エンジン 2094
- Unicode のサポート 2094
- インターネットへの接続 2095

ライセンス情報 2096

- 電子的なソフトウェアの配布 2096
- ソフトウェアのアクティベーションとライセンスメータリング 2096
- エンドユーザー使用許諾契約書 2097

30.1 XSLT および XQuery エンジンに関する情報

XMLSpy のXSLT およびXQuery エンジンは、W 3C 仕様に従っています、ですから、XMLSpy の以前のバージョン内のAltova エンジンより厳密です。この結果、以前のエンジンで無視されていた小さなエラーが、XMLSpy によりエラーとして挙げられます。

例えば

- パス演算子の結果がノードと非-ノードを両方含む場合、型エラー(`err:XPTY0018`) です。
- パス式 `E1/E2` 内の`E1` がノードのシーケンスを評価しない場合、型エラー(`err:XPTY0019`) です。

この種類のエラーが発生した場合、XSLT/XQuery ドキュメントまたはインスタンスドキュメントを必要に応じて修正してください。

このセクションは、エンジンの実装固有の機能を仕様別に整理して説明します。

- XSLT 1.0 1974
- XSLT 2.0 1974
- XSLT 3.0 1976
- XQuery 1.0 1976
- XQuery 3.1 1979

30.1.1 XSLT 1.0

XMLSpy のXSLT 1.0 エンジンは、W orld W ide W eb Consortium(ワールド・ワイド・ウェブ・コンソーシアム) (W 3C) の1999 年 11 月 16 日版のXSLT 1.0 勧告 および1999 年 11 月 16 日版のXPath 1.0 勧告 に準拠します。実装に関しての以下の情報に注意してください。

実装についての注意点

`xsl:output` の`method` 属性がHTMLに設定された場合、または、がHTML 出力 デフォルトで選択されている場合、XML またはXSLT ファイル内の特殊文字はHTML ドキュメントにHTML 文字参照として出力内に挿入されます。例えば、文字 U+00A0 (ブレーク無しのスペースのための16 進数レファレンス) がHTML コード内に文字の参照(` ` or ` `) 、または、エンティティ参照 ` ` として挿入されます

30.1.2 XSLT 2.0

このセクション:

- エンジン 適合性 1975
- 下位互換性 1975
- 名前空間 1975
- スキーマ認識 1975
- 実装固有の振る舞い 1976

適合性

XMLSpy のXSLT 2.0 エンジンは、W orld W ide W eb Consortium（ワールド・ワイド・ウェブ・コンソーシアム）（W 3C）の2007 年 1 月 23 日版のXSLT 2.0 勧告 および2010 年 12 月 14 日版のXPath 2.0 勧告に準拠します。

下位互換性

XSLT 2.0 エンジン は下位互換性を有します。XSLT 2.0 エンジンの 下位互換性が有効になるのは、 XSLT 1.0 スタイルシートを処理するためにXSLT 2.0 エンジン が使用される際です。XSLT 1.0 エンジン と下位互換性を持つXSLT 2.0 エンジンにより作成される出力に違いがあるかもしれないことに注意してください。

名前空間

XSLT 2.0 スタイルシートは、 XSLT 2.0. プレフィックス内で使用することのできる型コンストラクタ および関数を使用するため、以下の名前空間を宣言する必要があります。 下のリストは通常使用されるリスト です。希望する場合は、代替プレフィックスを使用することもできます。

名前空間	プレフィックス	名前空間 URI
XML スキーマ型	xs:	http://www.w3.org/2001/XMLSchema
XPath 2.0 関数	fn:	http://www.w3.org/2005/xpath-functions

通常これらの名前空間は、以下のリスト で表示されるようにxsl：スタイルシート または、 xsl:transform 要素で宣言されます:

```
<xsl：スタイルシート version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    ...
</xsl：スタイルシート>
```

次の点に注意してください:

- XSLT 2.0 エンジンは、（上のテーブルでリストされている） XPath 2.0 およびXQuery 1.0 関数 名前空間 をデフォルト の関数名前空間として使用します。XPath 2.0 およびXSLT 2.0 関数をプレフィックス無しでスタイルシート内で使用することができます。XPath 2.0 関数 名前空間 をスタイルシート内でプレフィックスと共に宣言すると、 割り当てられた宣言内でプレフィックスを追加して使用することができます。
- XML スキーマ 名前空間から型コンストラクタと型を使用する場合、名前空間 宣言内で使用された、プレフィックスを使用して型コンストラクタを呼び出さなければなりません（例えば、 xs:date）。
- XPath 2.0 関数の一部は、XML スキーマ データ型と同じ名前を保有します。例えば、 XPath 関数 fn:string および fn:boolean のためには、同じロケーション名: xs:string および xs:boolean を持つ、XML スキーマ データ型 が存在します。ですから、 XPath 式 string('Hello')を使用する場合、式は、 xs:string('Hello') ではなくて fn:string('Hello') として検証します。

スキーマ認識

XSLT 2.0 エンジンは、スキーマを認識します。ですから、ユーザー定義 スキーマ型 および xsl:validate 命令を使用することができます。

実装固有の振る舞い

以下は、XSLT 2.0 エンジンが、特定のXSLT 2.0 関数の振る舞いの実装-特定のアスペクトをどのように扱うかの説明です。

xsl:result-document
追加してサポートされるエンコードは以下の通りです(Altova-固有): `x-base16tobinary` および `x-base64tobinary`.

function-available
インスコープ関数の使用をテストする関数(XSLT、XPath、および拡張関数)。

unparsed-text
`href` 属性は、以下を受け入れます(i) ベース-uri フォルダー内のファイルの相対パス、および(ii) 相対パスを持つまたは持たない `file://` プロトコール。追加してサポートされるエンコードは以下の通りです(Altova-固有): `x-binarytobase16` および `x-binarytobase64`.

unparsed-text-available
`href` 属性は、以下を受け入れます(i) ベース-uri フォルダー内のファイルの絶対パス、および(ii) 絶対パスを持つまたは持たない `file://` プロトコール。追加してサポートされるエンコードは以下の通りです(Altova-固有): `x-binarytobase16` および `x-binarytobase64`.

メモ RaptorXML の先行製品である AltovaXML で実装されていた以下のエンコード値は使用しないでください: `base16tobinary`, `base64tobinary`, `binarytobase16` and `binarytobase64`.

30.1.3 XSLT 3.0

XMLSpy のXSLT 3.0 エンジンは、W orld W ide W eb Consortium(ワールド・ワイド・ウェブ・コンソーシアム)(W 3C) の2017 年 6 月 8 日版 XSLT 3.0 勧告候補 および2017 年 3 月 21 日版 XPath 3.1 勧告候補 に準拠します。

XSLT 3.0 エンジンは、XSLT 2.0 エンジンと同様の実装固有の機能を搭載しています。更に、以下のXSLT 3.0 機能へのサポートを含みます。更に次の一連の新規のXSLT 3.0 機能をサポートします: XPath/XQuery 3.1 関数とオペレーターとXPath 3.1 仕様。

メモ 任意のストリーミングの機能は現在サポートされていません。`streamable` 属性の値に関係なく、ドキュメント全体がメモリにロードされ、使用することのできるメモリが十分な場合は処理されます。64-ビットアプリでは、これは問題になりません。thisメモリが問題の場合は、システムに解決策を追加する必要があります。

30.1.4 XQuery 1.0

このセクション:

- エンジン適合性 1977
- スキーマ認識 1977
- エンコード 1977
- 名前空間 1975
- XML ソースと検証 1978
- 静的および動的な型のチェック 1978
- ライブラリモジュール 1978
- 外部関数 1978

- 照合順序 1978
- 数値データの精度 1978
- XQuery 命令サポート 1979

適合性

XMLSpy のXQuery 1.0 エンジンは、W orld W ide W eb Consortium（ワールド・ワイド・ウェブ・コンソーシアム）（W 3C）の2010 年 12 月 14 日版のXQuery 1.0 勧告に準拠します。XQuery 標準は、多数の機能の実装についての裁量を提供します。下には、XQuery 1.0 エンジンがどのようにこれらの機能を実装するかについて説明するリストが下に挙げられています。

スキーマ認識

XQuery 1.0 エンジン はスキーマを認識します。

エンコード

UTF-8 およびUTF-16 文字のエンコードは、サポートされています。

名前空間

以下の名前空間 URI と関連するバインドは定義済みです。

名前空間	プレフィックス	名前空間 URI
XML スキーマ型	xs:	http://www.w3.org/2001/XMLSchema
スキーマインスタンス	xsi:	http://www.w3.org/2001/XMLSchema-instance
内蔵の関数	fn:	http://www.w3.org/2005/xpath-functions
Local 関数	local:	http://www.w3.org/2005/xquery-local-functions

次の点に注意してください：

- XQuery 1.0 エンジンは、上にリストされたプレフィックスを名前空間に対応するバウンドとして認識します。
- Since the 上にリストされた内蔵の関数 名前空間は、XQuery 内のデフォルトの関数です。内蔵の関数 が呼び出される際、名前空間、fn: プレフィックス を使用する必要はありません。（例えば、 `string("Hello")` が `fn:string` 関数を呼び出す場合。）しかし、プレフィックス `fn:` は、クエリプロローグ内で名前空間を宣言することなく内蔵の関数を呼び出す時に使用することができます。（サンプル: `fn:string("Hello")`）.
- クエリプロローグ内で `default function` 名前空間 式を宣言することにより、デフォルトの関数 名前空間をすることにより変更することができます。
- XML スキーマ 名前空間空の型を使用する場合、プレフィックス `xs:` は、名前空間を明確に宣言することなく、また、これらのプレフィックスをクエリプロローグ内でバインドすることなく使用することができます。（サンプル: `xs:date` および `xs:yearMonthDuration`。）XML スキーマ 名前空間のために、他のプレフィックスを使用する場合は、クエリプロローグ内で明確に宣言されている必要があります。（サンプル: `declare 名前空間 alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.）
- `untypedAtomic`, `dayTimeDuration`, および `yearMonthDuration` データ型 が23 January 2007 のCR 共に、XPath データ型 名前空間からXML スキーマ 名前空間へ移動されていることに注意してください。ですから以下となります: `xs:yearMonthDuration`。

関数のための名前空間、型コンストラクタ、ノード テスト、が間違って割り当てられている場合、エラーが発生します。しかし、一部の関数はスキーマデータ型と同じ名前を持つことに注意してください。 例: `fn:string` および `fn:boolean`。(`xs:string` および `xs:boolean` は宣言されています) 名前空間 プレフィックス、関数または 型コンストラクタが使用されるか決定します。

XML ソースドキュメントと検証

XQuery 1.0 エンジンを使用して、実行される XQuery ドキュメント内の XML ドキュメントは、整形式である必要があります。しかし、XML スキーマに従い有効である必要はありません。XML ファイルが外部スキーマと関連付けられ、また有効な場合、ポストスキーマ検証情報が XML データのために生成され、クエリ検証のために使用されます。

静的および動的な型のチェック

静的分析フェーズは、外部レファレンスの存在（例、モジュールのため）、呼び出された関数と変数が定義済みであるか、など構文などのクエリのアスペクトをチェックします。静的分析フェーズでエラーが検知されると、実行は停止されます。

クエリが実際に作動中にランタイム中に動的な型チェックは実行されます。型がオペレーションの必要条件と整合性を持たない場合、エラーが報告されます。例えば、式 `xs:string("1") + 1` は、エラーを返します。型 `xs:string` のオペランドを足し算のオペレーションが実行できないためです。

ライブラリモジュール

ライブラリモジュールは、再利用のため関数と変数を保管します。XQuery 1.0 エンジンは単一の外部 XQuery ファイルに保管されているモジュールをサポートします。このようなモジュールファイルはプロローグにターゲット名前空間に関連するモジュール宣言を含む必要があります。 以下はモジュールサンプルです:

```
module namespace libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns:webaddress() { "http://www.altova.com" };
```

すべての関数および変数は、モジュールに関連した名前空間に属するモジュール内で宣言されています。モジュールはクエリプロローグ内の `import module` ステートメントを使用して XQuery ファイルにインポートする際に使用されます。`import module` ステートメントは、ライブラリモジュールファイル内で直接宣言された、関数と変数のみをインポートします。例:

```
import module namespace modlib = "urn:module-library" at "modulefilename.xq";
if         ($modlib:company = "Altova")
then        modlib:webaddress()
else        error("No match found.")
```

外部関数

外部関数 は、サポートされていません。例えば、`external` キーワードを仕様する式など。以下参照:

```
declare function hoo($param as xs:integer) as xs:string external;
```

照合順序

デフォルトの照合順序は、Unicode コードポイントをベースにした文字列を比較する Unicode-コードポイント照合順序です。その他にサポートされる照合順序は、ICU 照合順序 はここ[1980] にリストされるとおりです。特定の照合順序を使用する場合、サポートされる照合順序のリスト[1980] 内に与えられているとおり URI を提供します。`fn:max` と `fn:min` 関数を含む文字列の比較は、指定された照合順序により行われます。 照合順序オプションが指定されていない場合、デフォルトの Unicode-コードポイント照合順序 が使用されます。

数値データの精度

- `xs:integer` データ型には任意の精度があり、表記できる桁数に制限はありません。
- `xs:decimal` データ型には小数点の後に 20 桁の制限があります。
- `xs:float` と `xs:double` データ型には 15 桁の精度の制限があります。

(C) 2015-2021 Altova GmbH

XQuery 命令サポート

`Pragma` 命令 は、サポートされていません。発生した場合、無視されフォールバックの式が検証されます。

30.1.5 XQuery 3.1

XMLSpy のXQuery 3.1 エンジンは、W orld W ide W eb Consortium（ワールド・ワイド・ウェブ・コンソーシアム）（W 3C）の2017年 3 月 21 日版のXQuery 3.1 候補勧告に準拠し、またXPath およびXQuery 関数 3.1. へのサポートを含みます。XQuery 3.1 仕様は、3.0 仕様のスーパーセットです。XQuery 3.1 エンジンは、ですかXQuery 3.0 機能をサポートします。

実装固有の特性はXQuery 1.0[1976]同様です。

30.2 XSLT と XPath/XQuery 関数

このセクションでは、XPath および/または XQuery 式で使用することのできる、Altova 拡張関数と他の拡張関数をリストします。Altova 拡張関数は Altova の XSLT および XQuery エンジンで使用することができ、W3C 標準で定義された関数ライブラリで使用することのできる機能に追加して機能を提供します。

一般的な情報

以下の一般的な情報に注意してください:

- W3C 仕様により定義されているコア関数ライブラリの関数は、関数の呼び出しにプレフィックスは必要ありません。これは、XSLT および XQuery エンジンが、XPath/XQuery 関数仕様で指定されている `http://www.w3.org/2005/xpath-functions` プレフィックス無しの関数をデフォルト関数の名前空間に属するものとして読み込むためです。
- 関数において、各アイテムが引数となるようなシーケンスが期待されており、2つ以上のアイテムがシーケンスにより呼び出された場合、エラーが返されます。
- 全ての比較は Unicode コードポイントコレクションを使用することで行われます。
- QName の結果は `[prefix:]localname` という形式でシリアル化されます。

xs:decimal の精度

精度とは、数値内にある桁数のことで、仕様では少なくとも18桁が求められます。`xs:decimal` 型に結果が収められる除算の場合、端数処理を行うことなく精度は小数点以下の19桁になります。

黙示的なタイムゾーン

2つの `date`、`time`、または `dateTime` 値を比較する場合、比較する値のタイムゾーンを明らかにする必要があります。値の中にタイムゾーンが明示的に与えられていない場合、黙示的なタイムゾーンが使用されます。黙示的なタイムゾーンはシステムクロックから取得され `implicit-timezone()` 関数によりその値をチェックすることができます。

照合順序

デフォルトの照合順序は、Unicode コードポイントをベースに文字列を比較する Unicode コードポイント照合順序です。エンジンは Unicode 照合アルゴリズムを使用しています。他のサポートされる照合順序は下にリストされる ICU 照合順序 です。使用するには、サポートされる照合順序のリストの URI を提供してください(*下のテーブル*)。`max` と `min` 関数を含む、文字列の比較は、指定された照合順序に沿って行われます。照合順序オプションが指定されていない場合、デフォルトの Unicode コードポイント照合順序が使用されます。

言語	URI
`da`: デンマーク語	`da_DK`
`de`: ドイツ語	`de_AT, de_BE, de_CH, de_DE, de_LI, de_LU`
`en`: 英語	`en_AS, en_AU, en_BB, en_BE, en_BM, en_BW, en_BZ, en_CA, en_GB, en_GU, en_HK, en_IE, en_IN, en_JM, en_MH, en_MP, en_MT, en_MU, en_NA, en_NZ, en_PH, en_PK, en_SG, en_TT, en_UM, en_US, en_VI, en_ZA, en_ZW`
`es`: スペイン語	`es_419, es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_ES, es_GQ, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_US, es_UY, es_VE`
`fr`: フランス語	`fr_BE, fr_BF, fr_BI, fr_BJ, fr_BL, fr_CA, fr_CD, fr_CF, fr_CG, fr_CH, fr_CI, fr_CM, fr_DJ, fr_FR, fr_GA, fr_GN, fr_GP, fr_GQ, fr_KM, fr_LU, fr_MC, fr_MF, fr_MG, fr_ML, fr_MQ, fr_NE, fr_RE, fr_RW, fr_SN, fr_TD, fr_TG`

(C) 2015-2021 Altova GmbH

it: イタリア語	it_CH, it_IT
ja: 日本語	ja_JP
nb: ノルウェー語 (ブークモール)	nb_NO
nl: オランダ語	nl_AW, nl_BE, nl_NL
nn: ノルウェー語 (ニーノシュク)	nn_NO
pt: ポルトガル語	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: ロシア語	ru_MD, ru_RU, ru_UA
sv: スウェーデン語	sv_FI, sv_SE

名前空間軸
名前空間軸はXPath 2.0 にて廃止されましたが、名前空間軸の使用はサポートされています。XPath 2.0 メカニズムにより名前空間情報へアクセスするには、 in-scope-prefixes()、namespace-uri()、namespace-uri-for-prefix(関数を使用してください。

30.2.1 Altova 拡張関数

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova 拡張関数名前空間、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

W 3C のXPath/XQuery 関数仕様で定義された関数は、以下で使用することができます: (i) XSLT 子アンテキスト内のXPath 式と、(ii) XQuery 文書内のXQuery 式。このドキュメントでは、前者(XSLT 内のXPath) のコンテキストで使用することのできる関数を、**XP** シンボルと共に表示し、と称します。後者(XQuery) で使用することのできる関数はの**XQ** シンボルと共に表示され、XQuery 関数と共に作業することができます。W 3C のXSLT 仕様は、XPath/XQuery 関数の仕様ではなく、XSLT 文書内のXPath 式でも使用することのできる関数を定義します。これらの関数は、**XSLT** シンボルと共に表示され、XSLT 関数と称されます。関数を使用することのできるXPath/XQuery およびXSLT のバージョンは、関数の詳細に記載されています(*下のシンボルを参照してください*)。
XPath/XQuery およびXSLT 関数ライブラリからの関数は、プレフィックス無しでリストされています。Altova 拡張関数などの、他のライブラリからの関数はプレフィックスと共にリストされています。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3.1**

XSLT 関数[1982]

XSLT 関数はXSLT 2.0 の`current-group()` や`key()` 関数と同様に、XSLT コンテキストにて使用することができます。(例えば、XQuery コンテキストなどの) 非-XSLT コンテキストでは使用することができません。XBRL に対するXSLT 関数は、XBRL をサポートするエディションのAltova 製品でのみ使用することができます。

XPath/XQuery 関数

XPath/XQuery 関数は、 XSLT コンテキスト、 XQuery 関数のXPath 式で使用することができます:

- 日付/時刻 1985
- 位置情報 1999
- イメージに関連した 2010
- 数値 2018
- シーケンス 2038
- 文字列 2045
- その他 2051

チャート関数 (Enterprise および Server Editions のみ) 2053

チャート関数のためのAltova 拡張子は、**Enterprise** ならびに**サーバー エディション**のAltova 製品でしかサポートされていません。XML データからチャートを生成することができます。

30.2.1.1 XSLT 関数

XSLT 拡張関数 はXSLT コンテキスト内のXPath 式にて使用することができます。(例えば、XQuery コンテキストなどの) 非-XSLT コンテキストでは使用することができません。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、**`http://www.altova.com/xslt-extensions`** に収められており、**`altova:`** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3.1

標準関数

▼ distinct-nodes [altova:]

`altova:distinct-nodes(`*`node()*`***`)`** を **`node()*`** とする XSLT1 XSLT2 XSLT3
入力として1つ以上のノードを必要とし、同じセットから重複した値を持つノードを除いたノードを返します。XPath/XQuery 関数 `fn:deep-equal` を使用して比較を行うことができます。

▣ *サンプル*

- **`altova:altova:distinct-nodes`**`(country)` は重複した値を持つものを除く、全ての子 `country` ノード返しま

す。

evaluate [altova:]

altova:evaluate(XPathExpression *as xs:string*[**, ValueOf$p1, ... ValueOf$pN**]) **XSLT1** **XSLT2** **XSLT3**

XPath 式を必要とし、必須引数として文字列をパスします。評価された式の出力を返します。例えば **altova:evaluate('//Name[1]')** は、ドキュメント内の最初の Name 要素のコンテンツを返します。式 //Name[1] は、一重引用符を使用することにより、文字列としてパスされます。

altova:evaluate 関数は、オプションとして追加の引数を持つことができます。これらの引数は、p1, p2, p3... pN の名前を持つスコープ内の変数の値です。使用に関して以下の点に注意してください：(i) 変数は、X が整数である箇所のフォーム pX の名前と共に定義される必要があります。(ii) altova:evaluate 関数の引数は、(*上の署名参照*) 2 番目の引数からは、数値順の変数のシーケンスに対応した引数のシーケンス変数の値を与えます: p1 to pN: 第 2 引数は変数 p1 の値で、第 3 引数は、変数 p2 の値です。(iii) 変数の値は型 item* である必要があります。

サンプル

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 'hi')" />
```

outputs "hi 20 10"

上のリストに関して、以下の点に注意してください：

- altova:evaluate 式の第 2 引数は、変数 $p1 に割り当てられた値で、第三の引数は変数 $p2 に割り当てられた値です。
- 関数の第 4 番目の引数は、引用符による囲いで表示された文字列の値です。
- xs:variable 要素の select 属性は、XPath 式を提供します。この式は xs:string の型である必要があり、一重引用符で囲まれています。

変数の使用方法を更に説明するサンプル

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, //Name[1])" />
  ```
  最初の Name *要素の出力値*

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, '//Name[1]')" />
  ```
  Outputs "//Name[1]"

altova:evaluate() 拡張関数は、XSLT スタイルシート内の XPath 式が動的に評価される必要のあるシチュエーションで役に立ちます。例えば、ユーザーが並べ替えの必要条件をリクエストする場合、このシチュエーションは属性 UserReq/@sortkey に保管されます。スタイルシートでは、以下の式が使用できます: <xsl:sort select="**altova:evaluate(../UserReq/@sortkey)**" order="ascending"/>。altova:evaluate() 関数は、コンテキストノードの親の UserReq 子要素の sortkey 属性を読み込みます。sortkey 属性の値が Price の場合、Price は、altova:evaluate() 関数により返され、select 属性 <xsl:sort select="**Price**" order="ascending"/> の値になります。この sort 命令が、Order という要素のコンテキスト内で発生する場合、Order 要素は Price の子の値に従い並べ替えられます。また、@sortkey の値が、Date の場合、Order 要素は、Date の子の値に従い並べ替えられます。ですから、Order の並べ替えの条件は、ランタイムでの sortkey 属性から選択されます。これは、以下の式などでは達成することはできません: <xsl:sort select="**../UserReq/@sortkey**" order="ascending"/>。上の場合、並べ替え条件は sortkey 属性自身であり、Price または Date（または、現在の sortkey のコンテンツ）ではありません。

メモ 静的なコンテキストは、変数以外以外で、呼び出し環境の名前空間、型、機能、を含みます。ベースURI とデフォルトの名前空間は継承されます。

追加サンプル

- 静的な変数: `<xsl:value-of select="$i3, $i2, $i1" />`
  3 つの変数の値を出力します。

- 動的な変数を持つ動的 XPath 式:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />
  ```
  "30 20 10" を出力します。

- 動的な変数を持たない XPath 式:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath)" />
  ```
  出力エラー: $p3 に対して定義されている変数はありません。

encode-for-rtf [altova:]

altova:encode-for-rtf(input *as xs:string*, **preserveallwhitespace** *as xs:boolean*, **preservenewlines** *as xs:boolean*) を **xs:string** とする **XSLT2** **XSLT3**
RTF のためのコードに入力文字列を変換します。空白と新しい行は、それぞれの引数により指定される boolean の値に基づき保管されます。

[トップ 1982]

XBRL 関数

Altova XBRL 関数は XBRL をサポートする Altova 製品のエディションのみで使用することができます。

xbrl-footnotes [altova:]

altova:xbrl-footnotes(*node()***)** を **node()*** とする **XSLT2** **XSLT3**
ノードを入力引数として必要とし、入力ノードに参照される XBRL フットノート ノードを返します。

xbrl-labels [altova:]

altova:xbrl-labels(*xs:QName*, *xs:string***)** を **node()*** とする **XSLT2** **XSLT3**
以下の2 つの入力引数を必要とします: ノード名とノードを含むタクソノミファイルロケーション。関数は、入力ノードと関連した XBRL ラベルノードを返します。

[トップ 1982]

30.2.1.2 XPath/XQuery 関数: 日付と時刻

Altova の日付/時刻拡張関数は XPath と XQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の **XPath 3.0** および **XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数は XPath/XQuery 式で使用することができ、XPath、XQuery、および XSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は **Altova 拡張関数名前空間**、`http://www.altova.com/xslt-extensions` に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内の XPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内の XPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内の XQuery 式で使用):	XQ1 XQ3.1

▼ 機能によりグループ化

- xs:dateTime に期間を追加して、xs:dateTime を返す 1986
- xs:date に期間を追加して、xs:date を返す 1988
- xs:time に期間を追加して、return xs:time を返す 1989
- フォーマットと期間の取得 1988
- 現在の日付/時刻を生成する関数からタイムゾーンを削除する 1990
- 期間から日付、時刻、分数、および 秒数を返す 1991
- 日付から整数を週の曜日として返す 1993
- 日付から週数を整数として返す 1993
- 各型の構文コンポーネントから日付、時刻、期間 の型を構築する 1995
- 文字列入力から 日付、日付時刻 または 時刻 を構築する 1996
- 年齢に関連した関数 1998

▼ アルファベット順にグループ化

altova:add-days-to-date 1988
altova:add-days-to-dateTime 1986
altova:add-hours-to-dateTime 1986
altova:add-hours-to-time 1989
altova:add-minutes-to-dateTime 1986
altova:add-minutes-to-time 1989
altova:add-months-to-date 1988
altova:add-months-to-dateTime 1986
altova:add-seconds-to-dateTime 1986
altova:add-seconds-to-time 1989
altova:add-years-to-date 1988
altova:add-years-to-dateTime 1986
altova:age 1998
altova:age-details 1998
altova:build-date 1995
altova:build-duration 1995
altova:build-time 1995

altova:current-dateTime-no-TZ 1990
altova:current-date-no-TZ 1990
altova:current-time-no-TZ 1990
altova:date-no-TZ 1990
altova:dateTime-no-TZ 1990
altova:days-in-month 1991
altova:hours-from-dateTimeDuration-accumulated 1991
altova:minutes-from-dateTimeDuration-accumulated 1991
altova:seconds-from-dateTimeDuration-accumulated 1991
altova:format-duration 1988
altova:parse-date 1996
altova:parse-dateTime 1996
altova:parse-duration 1988
altova:parse-time 1996
altova:time-no-TZ 1990
altova:weekday-from-date 1993
altova:weekday-from-dateTime 1993
altova:weeknumber-from-date 1994
altova:weeknumber-from-dateTime 1994

[トップ 1985]

xs:dateTime に期間を追加する XP3.1 XQ3.1

これらの関数は**xs:dateTime** に期間を追加し、**xs:dateTime** を返します。xs:dateTime 型はCCYY-MM-DDThh:mm:ss.sss のフォーマットです。これはxs:date とxs:time フォーマットの連結で、T により区切られています。タイムゾーンサフィックス+01:00 (for example)は任意です。

add-years-to-dateTime [altova:]

altova:add-years-to-dateTime(DateTime *as xs:dateTime*, **Years** を *xs:integer*)
asxs:dateTime とする XP3.1 XQ3.1
日付までの期間を年数で表示します。第 2 の引数は第 1 の引数として与えられたxs:date に追加される年数です。結果はxs:date 型です。

サンプル

- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) は2024-01-15T14:00:00 を返します。
- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -4) は2010-01-15T14:00:00 を返します。

add-months-to-dateTime [altova:]

altova:add-months-to-dateTime(DateTime *as xs:dateTime*, **Months** を *xs:integer*)
asxs:dateTime とする XP3.1 XQ3.1
xs:dateTime に月数での期間を追加します(*下のサンプル参照*)。第 2 の引数は、第 1 の引数として与えられたxs:dateTime に追加される月数です。結果はxs:dateTime 型です。

サンプル

- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) 2014-11-15T14:00:00 を返します。
- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -2) 2013-11-

15T14:00:00 を返します。

add-days-to-dateTime [altova:]

altova:add-days-to-dateTime(DateTime *as xs:dateTime*, **Days** *as xs:integer*) を **xs:dateTime** とする **XP3.1** **XQ3.1**

xs:dateTime に日数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられた xs:dateTime に追加される日数です。結果は xs:dateTime 型です。

サンプル

- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) は 2014-01-25T14:00:00 を返します。
- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8) は 2014-01-25T14:00:00 を返します。

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(DateTime *as xs:dateTime*, **Hours** *as xs:integer*) を **xs:dateTime** とする **XP3.1** **XQ3.1**

xs:dateTime に時間数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられた xs:dateTime に追加される時間数です。結果は xs:dateTime 型です。

サンプル

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) は 2014-01-15T23:00:00 を返します。
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) は 2014-01-15T05:00:00 を返します。

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(DateTime *as xs:dateTime*, **Minutes** *as xs:integer*) を **xs:dateTime** とする **XP3.1** **XQ3.1**

xs:dateTime に分数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられた xs:dateTime に追加される分数です。結果は xs:dateTime 型です。

サンプル

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) 2014-01-15T14:55:00 を返します。
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) 2014-01-15T14:05:00 を返します。

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(DateTime *as xs:dateTime*, **Seconds** を *xs:integer*) **as xs:dateTime** とする **XP3.1** **XQ3.1**

xs:dateTime に秒数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられた xs:dateTime に追加される秒数です。結果は xs:dateTime 型です。

サンプル

- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20) 2014-01-

15T14:00:30 を返します。
- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5) 2014-01-15T14:00:05 を返します。

[トップ 1985]

xs:date に期間を追加する XP3.1 XQ3.1

これらの関数は**xs:date** に期間を追加し、**xs:date** を返します。xs:date 型はCCYY-MM-DD フォーマットです。

add-years-to-date [altova:]

altova:add-years-to-date(Date *as xs:date*, **Years** *as xs:integer*) を**xs:date** とする XP3.1 XQ3.1
日付までの期間を年数で表示します。第 2 の引数は第 1 の引数として与えられたxs:date に追加される年数です。結果は xs:date 型です。

サンプル

- **altova:add-years-to-date**(xs:date("2014-01-15"), 10) は2024-01-15 を返します。
- **altova:add-years-to-date**(xs:date("2014-01-15"), -4) は2010-01-15 を返します。

add-months-to-date [altova:]

altova:add-months-to-date(Date *as xs:date*, **Months** *as xs:integer*) を**xs:date** とする XP3.1 XQ3.1
日付までの期間を月数で表示します。第 2 の引数は第 1 の引数として与えられたxs:date に追加される月数です。結果は xs:date 型です。

サンプル

- **altova:add-months-to-date**(xs:date("2014-01-15"), 10) 2014-11-15 を返します。
- **altova:add-months-to-date**(xs:date("2014-01-15"), -2) 2013-11-15 を返します。

add-days-to-date [altova:]

altova:add-days-to-date(Date *as xs:date*, **Days** *as xs:integer*) を**xs:date** とする XP3.1 XQ3.1
日付までの期間を日数で表示します。第 2 の引数は第 1 の引数として与えられたxs:date に追加される日数です。結果は xs:date 型です。

サンプル

- **altova:add-days-to-date**(xs:date("2014-01-15"), 10) は 2014-01-25 を返します。
- **altova:add-days-to-date**(xs:date("2014-01-15"), -8) は2014-01-07 を返します。

[トップ 1985]

フォーマットと期間の取得 XP3.1 XQ3.1

これらの関数は入力 **xs:duration** または**xs:string** を解析し、それぞれan **xs:string** または**xs:duration** を返します。

▼ format-duration [altova:]

altova:format-duration(Duration *as xs:duration*, **Picture** *as xs:string*) **asxs:string** とする XP3.1 XQ3.1
第 1 の引数として提出された期間を、第 2 の引数として提出された文字列によりをフォーマットします。出力は、文字列によりフォーマットされたテキスト文字列です。

サンプル

- **altova:format-duration**(xs:duration("P2DT2H53M11.7S"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") は"Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7" を返します。
- **altova:format-duration**(xs:duration("P3M2DT2H53M11.7S"), "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") は"Months:03 Days:02 Hours:02 Minutes:53" を返します。

▼ parse-duration [altova:]

altova:parse-duration(InputString *as xs:string*, **Picture** *as xs:string*) を**xs:duration** とする XP3.1 XQ3.1
パターン化された文字列を最初の引数として、文字を第 2 の引数とします。入力文字列は文字をベースに解析され、xs:duration が返されます。

サンプル

- **altova:parse-duration**("Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") は "P2DT2H53M11.7S" を返します。
- **altova:parse-duration**("Months:03 Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") は "P3M2DT2H53M" を返します。

[トップ 1985]

xs:time に期間を追加する XP3.1 XQ3.1

これらの関数は**xs:time** に期間を追加し、 **xs:time** を返します。 **xs:time** 型はhh:mm:ss.sss 構文フォームです。文字 Z は協定世界時（UTC）を表します。他のタイムゾーンは、UTC との差異を +hh:mm または -hh:mm のフォーマットで表示しています。タイムゾーンの値が無い場合は、UTC ではない未知のタイムゾーンとして見なされます。

▼ add-hours-to-time [altova:]

altova:add-hours-to-time(Time *as xs:time*, **Hours** *as xs:integer*) を**xs:time** とする XP3.1 XQ3.1
日付までの期間を時間数で表示します。第 2 の引数は第 1 の引数として与えられた xs:time に追加される時間数です。結果は xs:time 型です。

サンプル

- **altova:add-hours-to-time**(xs:time("11:00:00"), 10) は21:00:00 を返します。
- **altova:add-hours-to-time**(xs:time("11:00:00"), -7) は04:00:00 を返します。

▼ add-minutes-to-time [altova:]

altova:add-minutes-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) を**xs:time** とする

XP3.1 XQ3.1
日付までの期間を分数で表示します。第 2 の引数は第 1 の引数として与えられた xs:date に追加される分数です。結果は xs:date 型です。

サンプル

- **altova:add-minutes-to-time**(xs:time("14:10:00"), 45) 14:55:00 を返します。
- **altova:add-minutes-to-time**(xs:time("14:10:00"), -5) 14:05:00 を返します。

add-seconds-to-time [altova:]

altova:add-seconds-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) を **xs:time** とする
XP3.1 XQ3.1
時間までの期間を秒数で表示します。第 2 の引数は、第 1 の引数として与えられた xs:time に追加される秒数です。結果は xs:time 型です。第 2 のコンポーネントは 0 から 59.999 の範囲であることができます。

サンプル

- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20) は 14:00:20 を返します。
- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20.895) は 14:00:20.895 を返します。

[トップ 1985]

日付/時刻データ型からタイムゾーンの部分を削除する XP3.1 XQ3.1

これらの関数は、現在の **xs:dateTime**、**xs:date**、または **xs:time** 値からそれぞれタイムゾーンを削除します。xs:dateTime と xs:dateTimeStamp の差異は、後者のタイムゾーンが必要な場合です。(前者の場合は任意です。) xs:dateTimeStamp 値のフォーマットは CCYY-MM-DDThh:mm:ss.sss±hh:mm または CCYY-MM-DDThh:mm:ss.sssZ です。、日付と時刻が xs:dateTimeStamp としてシステムクロックから読み込まれる場合、 current-dateTime-no-TZ() 関数がタイムゾーンを削除するために使用されます。

current-date-no-TZ [altova:]

altova:current-date-no-TZ() を **xs:date** とする XP3.1 XQ3.1
この関数は引数を必要としません。current-date() (システムクロックによる現在の時刻) のタイムゾーンの部分を削除し、xs:date の値を返します。

サンプル

現在の date が **2014-01-15+01:00** の場合:

- **altova:current-date-no-TZ**() は 2014-01-15 を返します。

current-dateTime-no-TZ [altova:]

altova:current-dateTime-no-TZ() を **xs:dateTime** とする XP3.1 XQ3.1
この関数は引数を必要としません。current-dateTime() (システムクロックによる現在の時刻) のタイムゾーンの部分を削除し、xs:dateTime の値を返します。

サンプル

現在の dateTime が **2014-01-15T14:00:00+01:00** の場合:

- **altova:current-dateTime-no-TZ**() は 2014-01-15T14:00:00 を返します。

current-time-no-TZ [altova:]

altova:current-time-no-TZ() を **xs:time** とする **XP3.1** **XQ3.1**
この関数は引数を必要としません。current-time() (システムクロックによる現在の時刻) のタイムゾーンの部分を削除し、xs:time の値を返します。

サンプル

現在の time が **14:00:00+01:00** の場合:

- **altova:current-time-no-TZ()** は 14:00:00 を返します。

date-no-TZ [altova:]

altova:date-no-TZ(InputDate *as xs:date*) を **xs:date** とする **XP3.1** **XQ3.1**
この関数は xs:date 引数を必要とし、タイムゾーンの部分を削除し、xs:date の値を返します。日付が変更されていない点に注意してください。

サンプル

- **altova:date-no-TZ**(xs:date("2014-01-15+01:00")) は 2014-01-15 を返します。

dateTime-no-TZ [altova:]

altova:dateTime-no-TZ(InputDateTime *as xs:dateTime*) を **xs:dateTime** とする **XP3.1** **XQ3.1**
この関数は xs:dateTime 引数を必要とし、タイムゾーンの部分を削除し、xs:dateTime の値を返します。日付が変更されていない点に注意してください。

サンプル

- **altova:dateTime-no-TZ**(xs:date("2014-01-15T14:00:00+01:00")) は 2014-01-15T14:00:00 を返します。

time-no-TZ [altova:]

altova:time-no-TZ(InputTime *as xs:time*) を **xs:time** とする **XP3.1** **XQ3.1**
この関数は xs:time 引数を必要とし、タイムゾーンの部分を削除し、xs:time 値を返します。時刻は変更されていないことに注意してください。

サンプル

- **altova:time-no-TZ**(xs:time("14:00:00+01:00")) は 14:00:00 を返します。

[トップ 1985]

期間から日数、時間数、分数、および、秒数を返す **XP3.1** **XQ3.1**

これらの関数は、期間から、月内の日数、時間数、分数、秒数をそれぞれ返します。

days-in-month [altova:]

altova:days-in-month(Year *as xs:integer*, **Month** *as xs:integer*) **as xs:integer** **XP3.1** **XQ3.1**
は 指定された月内の日数 を返します。Year と Month 引数を使用して月を指定することができます。

サンプル

- **altova:days-in-month**(2018, 10) は31 を返します。
- **altova:days-in-month**(2018, 2) は28 を返します。
- **altova:days-in-month**(2020, 2) は29 を返します。

hours-from-dayTimeDuration-accumulated

altova:hours-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*) **asxs:integer** **XP3.1** **XQ3.1**

DayAndTime 引数 (which is of type xs:duration) により提供される期間内の時間の総数を返します。Day とTime コンポーネント内の時間は、整数である結果に追加されます。60 分として新規の一時間は考えられます。ネガティブな時間の値で期間の結果をナビゲートすることができます。

サンプル

- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5D") は5 日間の時間の総計である120 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H") は5 日間の時間の総計プラス2 時間である122 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H60M") は5 日間の時間の総計プラス2 時間プラス60 分である123 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H119M") は5 日間の時間の総計プラス2 時間とプラス119 分である123 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H120M") は5 日間の時間の総計プラス2 時間プラス120 分である124 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("-P5DT2H") は-122 を返します。

minutes-from-dayTimeDuration-accumulated

altova:minutes-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*)
asxs:integer **XP3.1** **XQ3.1**

DayAndTime 引数 (which is of type xs:duration) により提出される期間内の分数の総数を返します Day とTime コンポーネントは、追加され、整数である結果に追加されます。ネガティブなマイナスの値で期間の結果をナビゲートします。

サンプル

- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT60M") は を返します。60
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H") は一時間内の分数である 60 を返します。
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H40M") は100 を返します。
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("P1D") は一日の秒数の総数である 1440 を返します。
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("-P1DT60M") は を返します。-1500

seconds-from-dayTimeDuration-accumulated

altova:seconds-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*)
asxs:integer **XP3.1** **XQ3.1**

(xs:duration の型である)DayAndTime 引数 により提出された期間内の秒数の総数を返します。Day とTime コンポーネ

ント内の秒数は、整数である結果に追加されます。ネガティブな秒の値で期間をナビゲートします。

サンプル

- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1M") は一分内の秒数である 60 を返します。
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H") は一時間内の秒数の総数である 3600 を返します。
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H2M") は 3720 を返します。
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("P1D") は一日の秒数の総数である 86400 を返します。
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("-P1DT1M") は -86460 を返します。

xs:dateTime または xs:dateReturn から週数を返す XP3.1 XQ3.1

これらの関数は、xs:dateTime または xs:date から曜日を(整数として)を返します。曜日は(米国式フォーマットを使用して) 1 から 7 と番号づけられています。 日曜=1 と番号付けられます。欧州のフォーマットでは月曜 (=1) として番号付けられます。日曜=1 である米国フォーマットは整数 0 がフォーマットを表示するために使用できる箇所で設定することができます。

weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *as xs:dateTime*) を **xs:integer** とする XP3.1 XQ3.1

時刻付きの日付を単一の引数として、この日付の曜日を正数として返します。曜日は 日曜=1 から開始して番号を付けます。(月曜=1 とする) ヨーロッパのフォーマットが必要な場合、この関数の他の署名を使用します (*下の次の署名を参照*)。

サンプル

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00")) は月曜を表示する 2 を返します。

altova:weekday-from-dateTime(DateTime *as xs:dateTime*, **Format** *as xs:integer*) **asxs:integer** とする XP3.1 XQ3.1

時間月の日付を最初の引数として、この日付の曜日を正数として返します。曜日は 月曜=1 から開始して番号を付けます。第 2 の引数(整数)が 0 の場合、日曜=1 から開始して、曜日は 1 から 7 と番号を付けます。第 2 の引数が 0 以外の整数の場合、月曜=1 です。第 2 の引数がない場合、関数はこの関数の他の署名を持つと読み込まれます (*前の次の署名を参照*)。

サンプル

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 1) は月曜を表示する 1 を返します。
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 4) は月曜を表示する 1 を返します。
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 0) は月曜を表示する 2 を返します。

weekday-from-date [altova:]

altova:weekday-from-date(Date *as xs:date*) を **xs:integer** とする XP3.1 XQ3.1

日付を単一の引数として、この日付の曜日を正数として返します。曜日は 日曜=1 から開始して番号を付けます。(月曜=1 とする) ヨーロッパのフォーマットが必要な場合、この関数の他の署名を使用します (*下の次の署名を参照*)。

サンプル

- **altova:weekday-from-date**(xs:date("2014-02-03+01:00")) は月曜を表示する 2 を返します。

altova:weekday-from-date(Date *as xs:date*, **Format** *as xs:integer*) を **xs:integer** とする
XP3.1 **XQ3.1**
日付を最初の引数とし、この日付の曜日を正数として返します。曜日は 月曜=1 から開始して番号を付けます。第 2 (フォーマット) 引数が 0, の場合、日曜=1 から開始し、曜日は 1 から 7 で番号付けられます。第 2 引数が整数で 0 以外の場合、月曜=1 です。第 2 の引数がない場合、関数はこの関数の他の署名を持つと読み込まれます (*前の署名を参照*)。

サンプル

- **altova:weekday-from-date**(xs:date("2014-02-03"), 1) は月曜を示す 1 を返します。
- **altova:weekday-from-date**(xs:date("2014-02-03"), 4) は月曜を示す 1 を返します。
- **altova:weekday-from-date**(xs:date("2014-02-03"), 0) は月曜を示す 2 を返します。

[トップ 1985]

xs:dateTime または xs:date から週数を返す **XP2** **XQ1** **XP3.1** **XQ3.1**

これらの関数は週数(整数として)を xs:dateTime から xs:date. から返します。週の番号付けは、米国、欧州、イスラムのカレンダーフォーマットで使用することができます。週の始まりが異なるため、週数の番号付けは、カレンダーのフォーマットにより異なります。(米国フォーマットでは、日曜、欧州フォーマットでは月曜、イスラムフォーマットでは土曜が週の開始日です)。

weeknumber-from-date [altova:]

altova:weeknumber-from-date(Date *as xs:date*, **Calendar** *as xs:integer*) を **xs:integer** とする
XP2 **XQ1** **XP3.1** **XQ3.1**
正数として提出された **Date** 引数の週数を返します。第 2 の引数(カレンダー)は続くカレンダーシステムを指定します。
サポートされる カレンダー の値は次のとおりです:

- 0 = US 米国のカレンダー (週の始まりは日曜日)
- 1 = ISO 標準、 欧州のカレンダー (週の始まりは月曜日)
- 2 = イスラムのカレンダー (週の始まりは土曜日)

デフォルトは 0 です。

サンプル

- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 0) は 13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 1) は 13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 2) は 13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23")) は 13 を返します。

上のサンプル(2014-03-23) の date の曜日は日曜日です。米国およびイスラムのカレンダーは欧州カレンダーのこの日付よりも一週間先です。

weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(DateTime *as xs:dateTime*, **Calendar** *as xs:integer*) を **xs:integer** とする **XP2** **XQ1** **XP3.1** **XQ3.1**
正数として提出された **DateTime** 引数の週数を返します。第 2 の引数(カレンダー)は続くカレンダーシステムを指定します。

サポートされる カレンダー の値は次のとおりです:

- 0 = US 米国のカレンダー（週の始まりは日曜日）
- 1 = ISO 標準、 欧州のカレンダー（週の始まりは月曜日）
- 2 = イスラムのカレンダー（週の始まりは土曜日）

Default is 0.

サンプル

- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 0)` は `13` を返します。
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 1)` は `13` を返します。
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 2)` は `13` を返します。
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00") )` は `13` を返します。

上のサンプル（`2014-03-23T00:00:00`）の dateTime の曜日は日曜日です。米国およびイスラムのカレンダーは欧州カレンダーのこの日付より丁度一週間先です。

[トップ 1985]

各型の構文コンポーネントから日付、時刻、期間の型を構築する XP3.1 XQ3.1

関数は、`xs:date`, `xs:time` または `xs:duration` の構文コンポーネントを入力引数とし、引数を結合させて対応するデータ型を構築します。

build-date [altova:]

`altova:build-date(Year` *`as xs:integer`*`, Month` *`as xs:integer`*`, Date` *`as xs:integer`*`)` を **`xs:date`** とする XP3.1 XQ3.1

第 1 、第 2 、第3 の引数は、それぞれ、年、月、日を表します。`xs:date` 型の値を構築するため結合されます。整数の値は、特定の日付の一部の適正な範囲内である必要があります。例えば第 2 の引数（月の部分）は `12` 以上であってはなりません。

サンプル

- `altova:build-date(2014, 2, 03)` は `2014-02-03` を返します。

build-time [altova:]

`altova:build-time(Hours` *`as xs:integer`*`, Minutes` *`as xs:integer`*`, Seconds` *`as xs:integer`*`)`** を **`xs:time`** とする XP3.1 XQ3.1

第 1 、第 2 、第3 引数はそれぞれ時間数（`0` から `23`）、分数（`0` から `59`）、および秒数（`0` から `59`）の値です。これらの値は、`xs:time` 型の値を作成するために結合されます。整数の値は、それぞれの部分の正しい範囲内である必要があります。例えば、秒（分）引数は、`59` 以上であってはなりません。値にタイムゾーンを追加するには、この関数の他の署名を使用してください（*次の署名を参照*）。

サンプル

- `altova:build-time(23, 4, 57)` は `23:04:57` を返します。

`altova:build-time(Hours` *`as xs:integer`*`, Minutes` *`as xs:integer`*`, Seconds` *`as xs:integer`*`,`

TimeZone *as xs:string*) を**xs:time** とする XP3.1 XQ3.1
第 1 、第 2 、第 3 引数はそれぞれ時間数（0 から 23）、分数（0 から 59）、および秒数（0 から 59）の値です。第四の引数は、値の一部としてタイムゾーンを与えます。4 つの引数が結合され、xs:time 型の値を構築します。例えば、第 2 の引数（分数）は 59 以上であってはなりません。

サンプル

- **altova:build-time**(23, 4, 57, '+1') は23:04:57+01:00 を返します。

build-duration [altova:]

altova:build-duration(Years *as xs:integer*, **Months** *as xs:integer*) を **xs:yearMonthDuration** とする XP3.1 XQ3.1
xs:yearMonthDuration 型の値を構築するためには2 つの引数が必要です。最初の引数は期間の年数 値の部分を与え、第 2 の引数は、月数 値の部分を与えます。第 2 の引数（月数）が同じまたはより大きくなると、整数は 12 により分割されます。商は、年数 の部分を表す 最初の引数に加算され、（除算の）残りはの部分は 月数の部分を表すために使用されます。期間の xs:dayTimeDuration 型を作成するには、次の署名を参照してください。

サンプル

- **altova:build-duration**(2, 10) はP2Y10M を返します。
- **altova:build-duration**(14, 27) はP16Y3M を返します。
- **altova:build-duration**(2, 24) はP4Y を返します。

altova:build-duration(Days *as xs:integer*, **Hours** *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*) を**xs:dayTimeDuration** とする XP3.1 XQ3.1
xs:dayTimeDuration 型の値を構築するためには4 つの引数が必要です。最初の引数は期間の日数 値の部分で、第 2 、第 3 、第 4 引数は、それぞれ期間の時間数、分数、秒数 値のです。これらの3 つの時間 引数は、次の高い単位の値に加算され、結果は期間の全体の値を計算するために使用されます。例えば、72 秒は、1M+12S（1 分と 12 秒）に変換され、この値が期間全体の値を計算するために使用されます。xs:yearMonthDuration 型の期間を構築するためには、次の署名を参照してください。

サンプル

- **altova:build-duration**(2, 10, 3, 56) はP2DT10H3M56S を返します。
- **altova:build-duration**(1, 0, 100, 0) はP1DT1H40M を返します。
- **altova:build-duration**(1, 0, 0, 3600) はP1DT1H を返します。

[トップ 1985]

文字列入力から 日付、日付時刻 または 時刻 を構築する XP2 XQ1 XP3.1 XQ3.1

関数は、文字列を引数として、xs:date、xs:dateTime または xs:time データ型を構築します。文字列は、データ型のコンポーネントを提出されたパターン引数をベースにして分析されます。

parse-date [altova:]

altova:parse-date(Date *as xs:string*, **DatePattern** *as xs:string*) を**xs:date** とする XP2 XQ1 XP3.1 XQ3.1
Date 入力文字列を**xs:date** の値として返します。第二の引数である**DatePattern** は、入力文字列のパターン（コンポーネントのシーケンス）を指定します。**DatePattern** は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネントセパレータと共に説明されています。下のサンプルを参照してください。

D 日付
M 月

(C) 2015-2021 Altova GmbH

Y	年

DatePattern のパターンは、Date のパターンに一致する必要があります。出力は xs:date 型であるため、出力は常に YYYY-MM-DD 構文フォーマットになります。

サンプル

- altova:parse-date(xs:string("06-03-2014"), "[D]-[M]-[Y]") は 2014-03-06 を返します。
- altova:parse-date(xs:string("06-03-2014"), "[M]-[D]-[Y]") は 2014-03-06 を返します。
- altova:parse-date("06/03/2014", "[M]/[D]/[Y]") は 2014-03-06 を返します。
- altova:parse-date("06 03 2014", "[M] [D] [Y]") は 2014-03-06 を返します。
- altova:parse-date("6 3 2014", "[M] [D] [Y]") は 2014-03-06 を返します。

parse-dateTime [altova:]

altova:parse-dateTime(DateTime *as xs:string*, **DateTimePattern** *as xs:string*) を xs:dateTime とする XP2 XQ1 XP3.1 XQ3.1

DateTime 入力文字列を xs:dateTime の値として返します。第二の引数である DateTimePattern は、入力文字列のパターン（コンポーネントのシーケンス）を指定します。DateTimePattern は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネント セパレータと共に説明されています。下のサンプルを参照してください。

D	日
M	月
Y	年
H	時間
m	分
s	秒

DateTimePattern のパターンは、DateTime のパターンに一致する必要があります。出力は xs:dateTime 型であるため、出力は常に YYYY-MM-DDTHH:mm:ss 構文フォーマットになります。

サンプル

- altova:parse-dateTime(xs:string("06-03-2014 13:56:24"), "[D]-[M]-[Y] [H]:[m]:[s]") は 2014-03-06T13:56:24 を返します。
- altova:parse-dateTime("time=13:56:24; date=06-03-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]") は 2014-03-06T13:56:24 を返します。

parse-time [altova:]

altova:parse-time(Time *as xs:string*, **TimePattern** *as xs:string*) を xs:time とする XP2 XQ1 XP3.1 XQ3.1

Time 入力文字列を xs:time の値として返します。第二の引数である TimePattern は、入力文字列のパターン（コンポーネントのシーケンス）を指定します。TimePattern は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネント セパレータと共に説明されています。下のサンプルを参照してください。

H	時間
m	分
s	秒

TimePattern のパターンは、Time のパターンに一致する必要があります。出力は xs:time 型であるため、出力は常に YYYY-HH:mm:ss 構文フォーマットになります。

サンプル

- **altova:parse-time**(xs:string("13:56:24"), "[H]:[m]:[s]") は13:56:24 を返します。
- **altova:parse-time**("13-56-24", "[H]-[m]") は13:56:00 を返します。
- **altova:parse-time**("time=13h56m24s", "time=[H]h[m]m[s]s") は13:56:24 を返します。
- **altova:parse-time**("time=24s56m13h", "time=[s]s[m]m[H]h") は13:56:24 を返します。

[トップ 1985]

年齢に関連した関数 XP3.1 XQ3.1

関数は計算された年齢(i) 入力引数の日付と現在の日付の期間 (ii) 2 つの入力引数の日付の期間 を返します。**altova:age** 関数は、年齢を年数で返します、**altova:age-details** は年齢を年数、月数、日数からなる3 つの整数のシーケンスで返します。

age [altova:]

altova:age(StartDate *as xs:date*) を **xs:integer** とする XP3.1 XQ3.1
引数として提出された開始日から(システムクロックから取得された)現在の日付 までの日数を数えて、あるオブジェクトの年齢の年数である正数を返します。入力引数が1 年より先大きい場合、または一年の場合、返される値は負の数です。

サンプル

If the current date is **2014-01-15**:

- **altova:age**(xs:date("2013-01-15")) は1 を返します。
- **altova:age**(xs:date("2013-01-16")) は0 を返します。
- **altova:age**(xs:date("2015-01-15")) は-1 を返します。
- **altova:age**(xs:date("2015-01-14")) は0 を返します。

altova:age(StartDate *as xs:date*, **EndDate** *as xs:date*) を **xs:integer** とする XP3.1 XQ3.1
最初の引数として提出された開始日から第 2 の引数の終了日 までの日数を数えて、あるオブジェクトの年齢の年数である正数を返します。最初の引数が1 年または第 2 の引数より先後の日付の場合、返される値は負の数です。

サンプル

If the current date is **2014-01-15**:

- **altova:age**(xs:date("2000-01-15"), xs:date("2010-01-15")) は10 を返します。
- **altova:age**(xs:date("2000-01-15"), current-date()) は、現在の日付が2014-01-15 の場合 14 を返します。
- **altova:age**(xs:date("2014-01-15"), xs:date("2010-01-15")) は-4 を返します。

age-details [altova:]

altova:age-details(InputDate *as xs:date*) を(**xs:integer**)* とする XP3.1 XQ3.1
引数と(システムクロックから取得された)現在の日付として提出された日付の間の年数、月数、日数の3 つの整数を返します。返された years+months+days の合計は、2 つの日付(入力の日付と現在の日付)の時間差です。 入力の日付は現在の日付より先早いまたはお遅い値をもつことができますが、入力の日付が早いか遅いかは返される値で示されていません。戻される値は常に正の数です。

サンプル

現在の日付が**2014-01-15** の場合:

- **altova:age-details**(xs:date("2014-01-16")) は (0 0 1)を返します。
- **altova:age-details**(xs:date("2014-01-14")) は (0 0 1) を返します。
- **altova:age-details**(xs:date("2013-01-16")) は (1 0 1) を返します。
- **altova:age-details**(current-date()) は (0 0 0)を返します。

altova:age-details(Date-1 *as xs:date*, **Date-2** *as xs:date*) を(**xs:integer**)* とする **XP3.1** **XQ3.1**
二つの引数間の年数、月数、日数の3 つの整数を返します。 返されたyears+months+days の合計は、2 つの入力の日付の時間差です。最初の引数として提出される二つの日付はどちらが速くても、または遅くてもかまいません。返される値は入力の日付が現在の日付より早いかまたは遅いかを示しません。戻される値は常に正の数です。

サンプル

- **altova:age-details**(xs:date("2014-01-16"), xs:date("2014-01-15")) は (0 0 1) を返します。
- **altova:age-details**(xs:date("2014-01-15"), xs:date("2014-01-16")) は (0 0 1)を返します。

[トップ 1985]

30.2.1.3 XPath/XQuery 関数: 位置情報

以下の位置情報 XPath/XQuery 拡張関数は、XMLSpy の現在のバージョンによりサポートされています。また、次で使用することができます: (i) XSLT コンテキスト内のXPath 式、または(ii) XQuery ドキュメント内のXQuery 式。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova 拡張関数名前空間、**http://www.altova.com/xslt-extensions** に収められており、**altova:**プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3.1**

format-geolocation [altova:]

altova:format-geolocation(Latitude *as xs:decimal*, **Longitude** *as xs:decimal*, **GeolocationOutputStringFormat** *as xs:integer*) を**xs:string** とする **XP3.1** **XQ3.1**
緯度と経度を最初の2 つの引数とし、位置情報を文字列として出力します。第 3 の引数 **GeolocationOutputStringFormat** は、位置情報出力文字列のフォーマットです。出力文字列のフォーマットを識別するために、正数の値 1 から 4 を使用します。(*下の「位置情報出力文字列フォーマット」を参照してください*)。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西)です。

メモ image-exif-data 2010 関数とExif メタデータの@Geolocation 2010 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- `altova:format-geolocation(33.33, -22.22, 4)` は `xs:string "33.33 -22.22"` を返します。
- `altova:format-geolocation(33.33, -22.22, 2)` は `xs:string "33.33N 22.22W"` を返します。
- `altova:format-geolocation(-33.33, 22.22, 2)` は `xs:string "33.33S 22.22E"` を返します。
- `altova:format-geolocation(33.33, -22.22, 1)` は `xs:string "33°19'48.00"S 22° 13'12.00"E"` を返します。

位置情報出力文字列フォーマット:

与えられた緯度と経度は下にリストされる出力フォーマットによりフォーマットされます。希望するフォーマットは整数 ID (1 から 4) により識別されます。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西) です。

1
度、分、10 進の秒、方角のサフィックス付き (N/S, E/W) **D°M'S.SS"N/S D°M'S.SS"E/W** *サンプル:* **33°55'11.11"N 22°44'66.66"W**

2
10 進の度、方角のサフィックス付き (N/S, E/W) **D.DDN/S D.DDE/W** *サンプル:* **33.33N 22.22W**

3
度、分、10 進の秒、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。 **+/-D°M'S.SS" +/-D°M'S.SS"** *サンプル:* **33°55'11.11" -22°44'66.66"**

4
10 進の度、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。 **+/-D.DD +/-D.DD** *サンプル:* **33.33 -22.22**

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つの Exif タグの連結です: 単位の追加された (*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151°

(C) 2015-2021 Altova GmbH

				13'11.73"E

parse-geolocation [altova:]

altova:parse-geolocation(GeolocationInputString *as xs:string*) を **xs:decimal+** とする **XP3.1 XQ3.1**

GeolocationInputString 引数を解析して、位置情報の緯度と経度（この通りの順番）を2 つの xs:decimal アイテムのシーケンスとして返します。位置情報入力文字列が提供されることのできるフォーマットは以下のリストの通りです。

メモ image-exif-data 2010 関数と Exif メタデータの @Geolocation 2010 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:parse-geolocation**("33.33 -22.22") は2 つの xs:decimals (33.33, 22.22) のシーケンスを返します。
- **altova:parse-geolocation**("48°51'29.6""N 24°17'40.2""") は2 つの xs:decimals (48.8582222222222, 24.2945) のシーケンスを返します。
- **altova:parse-geolocation**('48°51''29.6"N 24°17''40.2"') は2 つの xs:decimals (48.8582222222222, 24.2945) のシーケンスを返します。
- **altova:parse-geolocation**(**image-exif-data**(//MyImages/Image20141130.01)**/@Geolocation**) は2 つの xs:decimals のシーケンスを返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度（この通りの順番）を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1 つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き (N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS"
  +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM'
  +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (**N/S E/W**) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つの Exif タグの連結です: 単位の追加された (*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-km [altova:]

altova:geolocation-distance-km(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) を **xs:decimal** とする **XP3.1** **XQ3.1**
2 つの位置情報の間の距離をキロメートルで計算します。位置情報入力文字列を提供することのできるフォーマットは下にリストされています。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西) です。

メモ image-exif-data 2010 関数と Exif メタデータの @Geolocation 2010 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-distance-km**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""")
  は xs:decimal 4183.08132372392 を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が 1 つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い (") でハイライトされており、エスケープした単位インジケーターは青い ("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き (N/S, E/W)

D°M'S.SS"N/S D°M'S.SS"W/E
サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

- *Altova Exif 属性: 位置情報*

  Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つの Exif タグの連結です: 単位の追加された (*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-mi [altova:]

altova:geolocation-distance-mi(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) を **xs:decimal** とする **XP3.1** **XQ3.1**
2 つの位置情報の間の距離をマイルで計算します。位置情報入力文字列を提供することのできるフォーマットは下にリストされています。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西) です。

メモ image-exif-data 2010 関数と Exif メタデータの @Geolocation 2010 属性を位置情報入力文字列を提供する際に使用することができます。

- *サンプル*

- **altova:geolocation-distance-mi**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""") は xs:decimal 2599.40652340653 を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度（この通りの順番）を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き (N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:

```
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45
```

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 Geolocation を標準 Exif メタデータタグから生成します。Geolocation は、4 つの Exif タグの連結です: 単位の追加された (*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

▼ geolocations-bounding-rectangle [altova:]

altova:geolocations-bounding-rectangle(Geolocations を *xs:sequence* として GeolocationOutputStringFormat を *xs:integer* として) を **xs:string** として **XP3.1** **XQ3.1**
最初の引数として文字列のシーケンスを取ります。シーケンスの各文字列が位置情報です。関数はそれぞれが、最初の引数に送信されたすべての位置情報を含むために最適にサイズ調整された長方形の左上と右下の位置情報の座標である2つの文字列のシーケンスを返します。位置情報入力文字列が表示される書式は以下にリストされています(*「位置情報入力文字列書式」を参照してください*)。緯度の値は +90 から -90 への(N から S への)の範囲です。経度の値は +180 から -180 への(E から W への)の範囲です。

関数の2番目の引数は出力シーケンス内の2つの位置情報文字列の書式を指定します。引数はそれぞれの値が異なる位置情報文字列書式を識別する 1 から 4 の整数の値を取ります(*下記の「位置情報出力文字列書式」を参照してください*)。

メモ image-exif-data 2010 関数とExif メタデータの属性は入力文字列を提供するために使用することができます。

▣ *サンプル*

- **altova:geolocations-bounding-rectangle**(("48.2143531 16.3707266", "51.50939 -0.11832"), 1) はシーケンス ("51°30'33.804"N 0°7'5.952"W", "48°12'51.67116"N 16°22'14.61576"E") を返します。
- **altova:geolocations-bounding-rectangle**(("48.2143531 16.3707266", "51.50939 -0.11832", "42.5584577 -70.8893334"), 4) はシーケンス ("51.50939 -70.8893334", "42.5584577 16.3707266") を返します

▣ *位置情報入力文字列フォーマット:*

位置情報入力文字列は空白で区別された緯度と経度(この通りの順番)を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き(N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き(+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き(+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'

サンプル: +33°55.55' -22°44.44'

- 10 進の度、方角のサフィックス付き (N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: 33.33N 22.22W

- 10 進の度、プレフィックスサイン付き (+/-); (N/S E/W) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: 33.33 -22.22

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

位置情報出力文字列フォーマット:

与えられた緯度と経度は下にリストされる出力フォーマットによりフォーマットされます。希望するフォーマットは整数 ID (1 から 4) により識別されます。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西) です。

1
度、分、10 進の秒、方角のサフィックス付き (N/S, E/W) D°M'S.SS"N/S D°M'S.SS"E/W *サンプル:* 33°55'11.11"N 22°44'66.66"W

2
10 進の度、方角のサフィックス付き (N/S, E/W) D.DDN/S D.DDE/W *サンプル:* 33.33N 22.22W

3
度、分、10 進の秒、プレフィックスサイン付き (+/-); (N/E) のためのプラスサインは任意です。 +/-D°M'S.SS" +/-D°M'S.SS" *サンプル:* 33°55'11.11" -22°44'66.66"

4
10 進の度、プレフィックスサイン付き (+/-); (N/E) のためのプラスサインは任意です。 +/-D.DD +/-D.DD *サンプル:* 33.33 -22.22

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 Geolocationを標準 Exif メタデータ タグから生成します。Geolocation は、4 つのExif タグの連結です: 単位の追加された (*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

▼ geolocation-within-polygon [altova:]

altova:geolocation-within-polygon(Geolocation *as xs:string*, **((PolygonPoint** *as xs:string*)+)) を **xs:boolean** とする **XP3.1 XQ3.1**

PolygonPoint 引数により規明されている **Geolocation**（最初の引数）が多角形のエリア内に存在するかを決定します。もし、PolygonPoint 引数が最初と最後のポイントが同じ場合に作成される）閉じられたフィギュアを作成しない場合、フィギュアを閉じるために、最初のポイントが明示的に最後のポイントとして追加されます。全ての引数（Geolocation および PolygonPoint+）は、（*下にリストされるフォーマットの*）位置情報入力文字列により提供されます。Geolocation 引数が多角形エリア内にある場合、関数は true(); を返します。その他の場合は false() を返します。緯度の値の範囲は +90 から -90（北 から 南）です。経度の値の範囲は +180 から -180（東 から 西）です。

メモ image-exif-data 2010 関数と Exif メタデータの @Geolocation 2010 属性を位置情報入力文字列を提供する際に使用することができます。

▣ *サンプル*

- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48 24", "58 -32")**)** は true() を返します。
- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48 24")**)** は true() を返します。
- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48° 51'29.6""N 24°17'40.2""")**)** は true() を返します。

▣ *位置情報入力文字列フォーマット:*

位置情報入力文字列は空白で区別された緯度と経度（この通りの順番）を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90（N から S）。経度の値の範囲は +180 から -180（E から W）。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い（"）でハイライトされており、エスケープした単位インジケーターは青い（""）でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き（N/S, E/W）
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き（+/-）; （**N/E**）のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き（N/S, E/W）
  D°M.MM'N/S D°M.MM'W/E

サンプル: `33°55.55'N 22°44.44'W`

- 度、分、10 進の分、プレフィックスサイン付き(+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: `+33°55.55' -22°44.44'`

- 10 進の度、方角のサフィックス付き(N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: `33.33N 22.22W`

- 10 進の度、プレフィックスサイン付き(+/-); (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: `33.33 -22.22`

フォーマットの組み合わせのサンプル:
```
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45
```

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つのExif タグの連結です: 単位の追加された(*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-within-rectangle [altova:]

altova:geolocation-within-rectangle(Geolocation *as xs:string*, **RectCorner-1** *as xs:string*, **RectCorner-2** *as xs:string*) を**xs:boolean** とする **XP3.1** **XQ3.1**
長方形の対格の角を指定する第 2 及び第 3 引数(**RectCorner-1**、**RectCorner-2**)により説明されている **Geolocation**(最初の引数) が長方形のエリア内に存在するかを決定します。全ての引数(Geolocation、**RectCorner-1** および**RectCorner-2**) は、(*下にリストされるフォーマットの*) 位置情報入力文字列により提供されます。Geolocation 引数が長方形エリア内にある場合、関数は`true()`; を返します。その他の場合は`false()`を返します。緯度の値の範囲は`+90` から`-90`(北 から 南) です。経度の値の範囲は`+180` から`-180`(東 から 西)です。

メモ image-exif-data 2010 関数とExif メタデータの @Geolocation 2010 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "-48 24") は`true()` を返します。
- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "48 24") は`false()` を返します。
- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "48°51'29.6""S 24°

17'40.2""") は true() を返します。

- *位置情報入力文字列フォーマット:*

  位置情報入力文字列は空白で区別された緯度と経度（この通りの順番）を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

  メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルとしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い (") でハイライトされており、エスケープした単位インジケーターは青い ("") でハイライトされています。

  - 度、分、10 進の秒、方角のサフィックス付き (N/S, E/W)
    D°M'S.SS"N/S D°M'S.SS"W/E
    サンプル: **33°55'11.11"N 22°44'55.25"W**

  - 度、分、10 進の秒、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
    サンプル: **33°55'11.11" -22°44'55.25"**

  - 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
    D°M.MM'N/S D°M.MM'W/E
    サンプル: **33°55.55'N 22°44.44'W**

  - 度、分、10 進の分、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
    サンプル: **+33°55.55' -22°44.44'**

  - 10 進の度、方角のサフィックス付き (N/S, E/W)
    D.DDN/S D.DDW/E
    サンプル: **33.33N 22.22W**

  - 10 進の度、プレフィックスサイン付き (+/-); (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
    サンプル: **33.33 -22.22**

  フォーマットの組み合わせのサンプル:
  33.33N -22°44'55.25"
  33.33 22°44'55.25"W
  33.33 22.45

- *Altova Exif 属性: 位置情報*

  Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータタグから生成します。**Geolocation** は、4 つの Exif タグの連結です: 単位の追加された (*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

[トップ 1999]

30.2.1.4 XPath/XQuery 関数: イメージに関連

以下のイメージに関連したXPath/XQuery 拡張関数は、XMLSpy の現在のバージョンによりサポートされています。また、次で使用することができます: (i) XSLT コンテキスト内のXPath 式、または(ii) XQuery ドキュメント内のXQuery 式。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova 拡張関数名前空間、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3.1**

suggested-image-file-extension [altova:]

altova:suggested-image-file-extension(Base64String *as string*) を **string?** とする **XP3.1** **XQ3.1**
イメージファイルのBase64 エンコード を引数として、イメージのファイル拡張子を、イメージのBase64エンコード内の記録として返します。返された値は、エンコード内で使用することのできるイメージ型情報を基にしたヒントです。この情報が使用できない場合、空の文字列が返されます。この関数は、Base64 イメージをファイルとして保存し、適切なファイル拡張子を動的取得する際に役に立ちます。

サンプル

- **altova:suggested-image-file-extension**(/MyImages/MobilePhone/Image20141130.01) は 'jpg' を返します。
- **altova:suggested-image-file-extension**($XML1/Staff/Person/@photo) は '' を返します。

上のサンプルでは、関数の引数として与えられたノードはBase64 エンコードイメージを含むと仮定します。最初のサンプルは jpg をファイルの型および拡張子として取得します。二番目のサンプルでは、与えられたBase64 エンコードは使用できる拡張子の情報を提供しません。

mt-transform-image [altova:]

altova:mt-transform-image(Base64Image *as Base64BinaryString*, **Size** *as item()+*, **Rotation** *as xs:integer*, **Quality** *as xs:integer*) を **Base64BinaryString** とする **XP3.1** **XQ3.1**
Base64-エンコード イメージを最初の引数として、変換されたBase64-エンコード イメージを返します。第 2、第 3、第 4 引数は変換された以下のイメージ パラメーターです: サイズ、回転、およびクオリティ。

- サイズ引数には3 つのサイズ変更のオプションがあります。

(X,Y)	絶対ピクセルの値。アスペクト率は保持されません。高さと幅は自動的にイメージの長いおよび短い辺に逢わされるため、高さと幅の指示は関係ありません。2 つの整数アイテムのシーケンスとして値が入力されます。かっこが必要です。
X	X をピクセルの新しい長い辺として、イメージの縦横比のサイズ変更が行われます。アスペクト率は保持されます。値は整数で引用符なしで入力されます。
'X%'	元のディメンションの与えられたパーセンテージにイメージのサイズを変更します。値は文字列として引用符を付けて入力される必要があります。

- 回転は以下の値を持つことができます: 90、180、270、-90、-180、-270。これらの値は回転の度数です。正の値はイメージを時計回りに回転させます。負の値はイメージを反時計回りに回転させます。 Altova Exif 属性 **OrientationDegree** を使用して、イメージの現在の回転の度数(0, 90, 180, 270) をイメージの Exif **Orientation** タグから取得することができます。ですが、**OrientationDegree** 属性はデータの **Orientation** タグから取得されるため、Exif データ内に **Orientation** タグが存在する場合のみ使用することができます。(*下の OrientationDegree 説明を参照してください*)。
- クオリティは、0 から 100 の値で、JPEG 圧縮の IJG クオリティスケールの値を参照していますが、クオリティのパーセンテージのインジケーターではありません。サイズとクオリティのどちらかを優先すると、もう一方の優先度が下がります。フルカラーソースのイメージでは、75 が通常最高値と見なされています。もし、75 が満足のいく結果をもたらさない場合は、値を上げてください。

メモ Exif データが元のイメージに存在する場合、変換時に削除され、変換されたイメージには Exif データは存在しません。

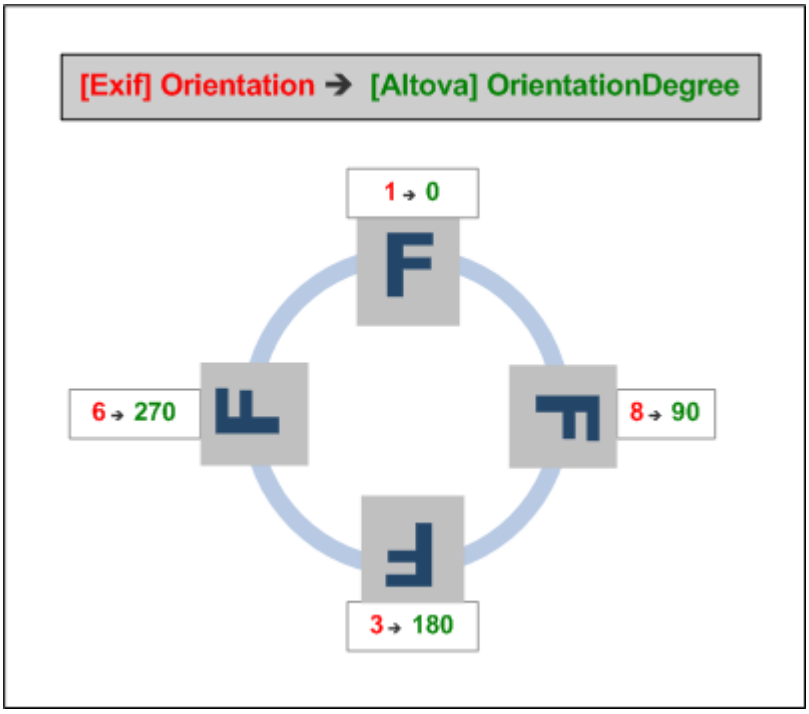
サンプル

- **mt-transform-image**(Images/Image[@id='43'], '50%', 90, 75)
  関数は、43 の @id 値を持つイメージ子孫ノード内で Base64-エンコード 文字列として保管されているイメージを入力とします。関数は変換されたイメージを返します。変換されたイメージは 50% まで、サイズ変更され、時計回りに 90 度回転され、75 のクオリティレベルを与えられます。
- **mt-transform-image**(Images/Image[@id='43'], 400, 90, 75)
  関数は前のサンプルと同じ結果を出しますが、長い辺は 400 ピクセルの特定の値に設定されています。元のイメージのアスペクト率は保持されます。
- **mt-transform-image**(Images/Image[@id='43'], (400, 280), image-exif-data($XML1/$XML1/Images/ReferenceImage)/@OrientationDegree, 75)
  このサンプルは、前のサンプルと同じイメージを選択し、同じクオリティ値 (75) を設定します。イメージのサイズは 400x280 ピクセルに設定されています。回転値は、ノード内の Base64-エンコード イメージの **@OrientationDegree** 属性から得られます。

Altova Exif 属性: OrientationDegree

Altova XPath/XQuery エンジンはカスタム属性 **OrientationDegree** を Exif メタデータタグ **Orientation** から生成します。

OrientationDegree は標準 Exif タグ Orientation を正数の値 (1、8、3 または 6) からそれぞれ対応する度数の値 (0, 90, 180, 270) へ下の図に示されているように変換します。2、4、5、7 の Orientation 値の変換はないことに注意してください。(これらの向きはイメージ 1 を垂直方向中央軸で反転して、2 の値を持つイメージを取得します。そして、このイメージを 90-度ごとに時計回りにジャンプさせそれぞれ 7、4、および 5 の値を取得します。)。



標準 Exif メタタグ

- ImageWidth
- ImageLength
- BitsPerSample
- Compression
- PhotometricInterpretation
- Orientation
- SamplesPerPixel
- PlanarConfiguration
- YCbCrSubSampling
- YCbCrPositioning
- XResolution
- YResolution
- ResolutionUnit
- StripOffsets
- RowsPerStrip
- StripByteCounts
- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients
- ReferenceBlackWhite
- DateTime
- ImageDescription

(C) 2015-2021 Altova GmbH

- Make
- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion
- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio

- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

image-exif-data [altova:]

altova:image-exif-data(Base64BinaryString *as string*) を **element?** とする **XP3.1** **XQ3.1**

Base64 エンコード JPEG イメージを引数として、イメージの Exif メタデータを含む **Exif** という名の要素を返します。Exif メタデータは `Exif` 要素の属性の値ペアとして作成されます。属性名は、Base64 エンコード内で検出された Exif データタグです。Exif 仕様タグのリストは以下の通りです。ベンダー特有のタグが Exif データ内に存在する場合、タグとその値も属性値のペアとして返されます。標準 Exif メタデータタグに追加して（*下のリスト参照*）Altova 特有の属性値のペアもまた生成されます。これらの Altova Exif 属性は以下のとおりです。

サンプル

- 属性にアクセスするには、以下の関数を使用します:
  ```
  image-exif-data(//MyImages/Image20141130.01)/@GPSLatitude
  image-exif-data(//MyImages/Image20141130.01)/@Geolocation
  ```
- 全ての属性にアクセスするには、以下の関数を使用します:
  ```
  image-exif-data(//MyImages/Image20141130.01)/@*
  ```
- 全ての属性の名前にアクセスするには、以下の式を使用します:
  ```
  for $i in image-exif-data(//MyImages/Image20141130.01)/@* return name($i)
  ```
  関数により返される属性の名前を検出するために役に立ちます。

Altova Exif 属性: 位置情報

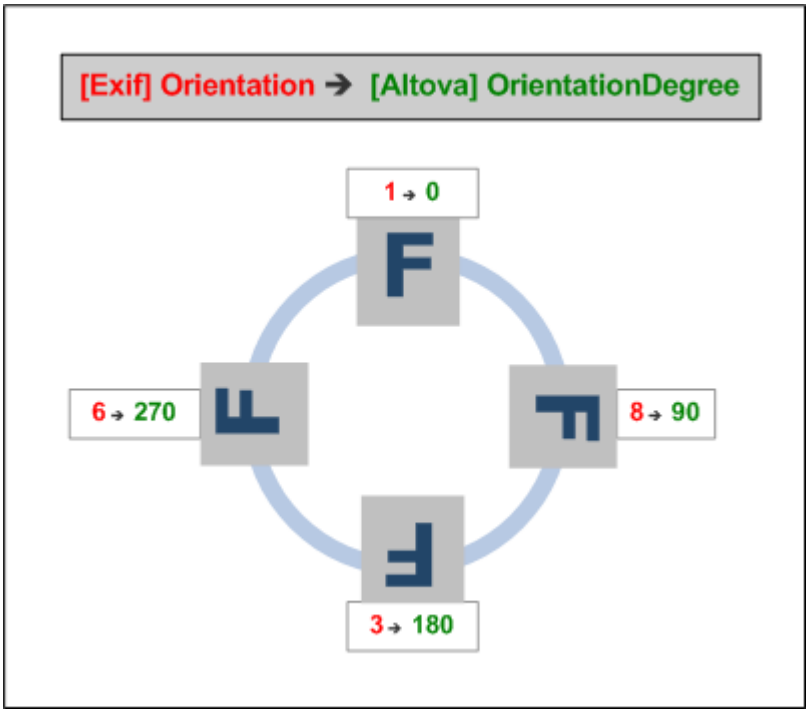
Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つのExif タグの連結です: 単位の追加された(*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

Altova Exif 属性: OrientationDegree

Altova XPath/XQuery エンジンはカスタム属性 **OrientationDegree** をExif メタデータタグ **Orientation** から生成します。

OrientationDegree は標準 Exif タグ Orientation を 正数の値 (1、8、3 または 6) からそれぞれ対応する度数の値 (0, 90, 180, 270) へ下の図に示されているように変換します。2、4、5、7 の Orientation 値の変換はないことに注意してください。(これらの向きはイメージ 1 を垂直方向中央軸で反転して、2 の値を持つイメージを取得します。そして、このイメージを 90-度ごとに時計回りにジャンプさせそれぞれ 7、4、および 5 の値を取得します。)。



標準 Exif メタデータ タグのリスト

- ImageWidth
- ImageLength
- BitsPerSample
- Compression
- PhotometricInterpretation
- Orientation
- SamplesPerPixel
- PlanarConfiguration
- YCbCrSubSampling
- YCbCrPositioning
- XResolution
- YResolution
- ResolutionUnit
- StripOffsets
- RowsPerStrip
- StripByteCounts
- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients
- ReferenceBlackWhite
- DateTime
- ImageDescription

(C) 2015-2021 Altova GmbH

- Make
- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion
- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio

- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

[トップ2010]

30.2.1.5 XPath/XQuery 関数: 数値

Altova の数値拡張関数は XPath と XQuery 内で使用することができ、データを更に処理するための追加機能を提供します。このセクションの関数は Altova の **XPath 3.0** と **XQuery 3.0** エンジンと使用することができます。XPath/XQuery コンテキストで使用することができます。

(C) 2015-2021 Altova GmbH

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova 拡張関数名前空間、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3.1

自動付番関数

generate-auto-number [altova:]

altova:generate-auto-number(ID *as xs:string*, **StartsWith** *as xs:double*, **Increment** *as xs:double*, **ResetOnChange** *as xs:string*) を xs:integer とする XP1 XP2 XQ1 XP3.1 XQ3.1
関数が呼び出される度に番号を生成します。関数が初めて呼び出される際に生成される最初の番号はStartsWith 引数により指定されます。その後の関数の呼び出しは新しい番号を生成します。この番号は前に生成された番号を、Increment 引数で指定された値ごとにインクリメントします。実質的には、altova:generate-auto-number 関数は、ID 引数により指定されたカウンターを作成し、このカウンターが関数が呼び出されるごとにインクリメントされます。ResetOnChange 引数の値が、前の関数呼び出しから変更されると、生成される番号の値がStartsWith 値にリセットされます。自動付番はaltova:reset-auto-number 2051 関数を使用してリセットすることができます。

サンプル

- **altova:generate-auto-number**("ChapterNumber", 1, 1, "SomeString") は、関数が呼び出される度に番号を 1 つ返します。1 から始まり、関数が呼び出される度に1 ずつインクリメントします。その後の呼び出しの4 番目の"SomeString"引数がである限り、インクリメントは継続されます。4 番目の引数の値が変更されると、(ChapterNumber と呼ばれる) カウンターは1 にリセットされます。ChapterNumber の値もaltova:reset-auto-number 2051 関数の呼び出しによりのaltova:reset-auto-number 2051 ("ChapterNumber") ようにリセットされます。

reset-auto-number [altova:]

altova:reset-auto-number(ID *as xs:string*) XP1 XP2 XQ1 XP3.1 XQ3.1
この関数は、ID 引数内で名づけられた自動付番カウンターの番号をリセットします。引数内のカウンター名を作成した関数の引数により指定された数値にリセットされます。altova:generate-auto-number 2051

サンプル

- **altova:reset-auto-number**("ChapterNumber") は、altova:generate-auto-number 2051 関数により作成されたChapterNumber という名の自動付番カウンターをリセットします。
- ChapterNumber を作成したaltova:generate-auto-number 2051 関数のStartsWith 引数の値に数値をリセットします。

[[トップ] 2018]

数値関数

hex-string-to-integer [altova:]

altova:hex-string-to-integer(HexString *as xs:string*) を**xs:integer** とする **XP3.1** **XQ3.1**
10 進のシステム(10進)内の正数の16 進の同値の文字列引数を必要とし、10 進の整数を返します。

サンプル

- **altova:hex-string-to-integer**('1') は1 を返します。
- **altova:hex-string-to-integer**('9') は9 を返します。
- **altova:hex-string-to-integer**('A') は10 を返します。
- **altova:hex-string-to-integer**('B') は11 を返します。
- **altova:hex-string-to-integer**('F') は15 を返します。
- **altova:hex-string-to-integer**('G') はエラーを返します。
- **altova:hex-string-to-integer**('10') は16 を返します。
- **altova:hex-string-to-integer**('01') は1 を返します。
- **altova:hex-string-to-integer**('20') は32 を返します。
- **altova:hex-string-to-integer**('21') は33 を返します。
- **altova:hex-string-to-integer**('5A') は90 を返します。
- **altova:hex-string-to-integer**('USA') はエラーを返します。

integer-to-hex-string [altova:]

altova:integer-to-hex-string(Integer *as xs:integer*) を**xs:string** とする **XP3.1** **XQ3.1**
正数を引数として必要とし、文字列として自身のベース16 の同値を返します。

サンプル

- **altova:integer-to-hex-string**(1) は'1' を返します。
- **altova:integer-to-hex-string**(9) は'9' を返します。
- **altova:integer-to-hex-string**(10) は'A' を返します。
- **altova:integer-to-hex-string**(11) は'B' を返します。
- **altova:integer-to-hex-string**(15) は'F' を返します。
- **altova:integer-to-hex-string**(16) は'10' を返します。
- **altova:integer-to-hex-string**(32) は'20' を返します。
- **altova:integer-to-hex-string**(33) は'21' を返します。
- **altova:integer-to-hex-string**(90) は'5A' を返します。

[[トップ] 2018]

数値フォーマット関数

[[トップ] 2018]

30.2.1.6 XPath/XQuery 関数: スキーマ

下にリストされる Altova 拡張関数はスキーマの情報を返します。以下は関数の詳細 (i) サンプル(ii) スキーマコンポーネントのリストとパースペクティブのプロパティです。 Altova の**XPath 3.0** と**XQuery 3.0** エンジンと共に使用することができ XPath/XQuery コンテキスト内で見つけることができます。

スキーマドキュメントからのスキーマ情報
関数 **altova:schema** には以下の2つの引数が存在します: 1つのゼロ引数と他の2つの引数。ゼロ引数関数はスキーマ全体を返します。この開始点からスキーマ内をナビゲートし必要とするスキーマコンポーネントをロケートします。2-引数関数は自身のQName により識別される特定のコンポーネントの型を返します。両方の場合、戻り値は関数です。返されたコンポーネント内をナビゲートするには、特定のコンポーネントのプロパティを選択する必要があります。プロパティが非動的アイテム(すなわちコンポーネントの場合) このコンポーネントのプロパティをさらに選択してナビゲートすることが可能です。選択されたプロパティが動的なアイテムの場合、アイテムの値が返され、ナビゲートを更に行うことはできません。

メモ Xquery 式ではスキーマは明示的にインポートされる必要があります。XPath 式では、スキーマは処理環境にインポートされる必要があります。例えば、XSLT には **xslt:import** 命令を使用してインポートします。

XML ノードからのスキーマ情報
関数 **altova:type** はXMLドキュメントのノードを送信しノードの型情報をPSVI から返します。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova 拡張関数名前空間、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3.1

▼ Schema (引数無し)

altova:schema() as (function(xs:string) as item()*)? XP3.1 XQ3.1
schema コンポーネント全体を返します。**schema** コンポーネントを **schema** コンポーネントのプロパティの一つを選択して更にナビゲートすることができます。

- このプロパティがコンポーネントの場合、このコンポーネントのプロパティの1つを選択して更に深く他のステップをナビゲートすることができます。このステップは更にスキーマをナビゲートするために繰り返すことができます。
- コンポーネントが動的な値の場合、動的な値が返され更にナビゲートすることはできません。

schema コンポーネントのプロパティ:

```
"type definitions"
"attribute declarations"
"element declarations"
"attribute group definitions"
"model group definitions"
"notation declarations"
"identity-constraint definitions"
```

(**schema** 以外の) 他のすべてのコンポーネント型のプロパティが下にリストされています。

メモ Xquery 式ではスキーマは明示的にインポートされる必要があります。XPath 式では、スキーマは処理環境にインポートされる必要があります。例えば、XSLT には **xslt:import** 命令を使用してインポートします。

▣ *サンプル*

- `import schema "" at "C:\Test\ExpReport.xsd"; for $typedef in altova:schema() ("type definitions") return $typedef ("name")` はスキーマ内のすべての単純型または複合型の名前を返します
- `import schema "" at "C:\Test\ExpReport.xsd"; altova:schema() ("type definitions")[1]("name")` はスキーマ内の単純型または複合型の最初の名前を返します

コンポーネントとそのプロパティ

Assertion

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Assertion"
test	XPath プロパティレコード	

Attribute Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Declaration"
name	文字列	属性のローカル名
target namespace	文字列	属性の名前空間 URI
type definition	Simple Type または Complex Type	
scope	プロパティを持つ関数 ("class":"Scope", "variety": "global" or "local", "parent": the containing Complex Type or Attribute Group)	
value constraint	存在する場合、プロパティを持つ関数 ("class": "Value Constraint"、"variety": "fixed" または "default", "value": atomic value, "lexical form": string。"value" プロパティは namespace-sensitive 型のために使用することはできないことにご注意してください。	
inheritable	ブール値	

Attribute Group Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Group Definition"
name	文字列	属性グループのローカル名
target namespace	文字列	属性グループの名前空間 URI
attribute uses	(Attribute Use) のシーケンス	
attribute wildcard	任意の属性のワイルドカード	

Attribute Use

(C) 2015-2021 Altova GmbH

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Use"
required	ブール値	属性が必要な場合はtrue 、任意の場合はfalse
value constraint	Attribute Declaration を参照	
inheritable	ブール値	

- Attribute Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

- Complex Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
base type definition	Complex Type Definition	
final	文字列のシーケンス ("restriction"|"extension")	
context	空のシーケンス(not implemented)	
derivation method	文字列 (″restriction″|″extension″)	
abstract	ブール値	
attribute uses	Attribute Use のシーケンス	
attribute wildcard	任意の属性のワイルドカード	
content type	プロパティを持つ関数: (″class″:″Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited	文字列のシーケンス	

substitutions	("restriction"|"extension")	
assertions	アサーションのシーケンス	

Element Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
type definition	Simple Type または Complex Type	
type table	プロパティを持つ関数 ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Simple Type or Complex Type)	
scope	プロパティを持つ関数 ("class":"Scope", "variety": ("global"|"local"), "parent": optional Complex Type)	
value constraint	Attribute Declaration を参照	
nillable	ブール値	
identity-constraint definitions	Identity Constraint のシーケンス	
substitution group affiliations	Element Declaration のシーケンス	
substitution group exclusions	文字列のシーケンス ("restriction"|"extension")	
disallowed substitutions	文字列のシーケンス ("restriction"|"extension"|"substitution")	
abstract	ブール値	

Element Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 ("strict"|"lax"|"skip")	

Facet

プロパティ名	プロパティ型	プロパティ値

(C) 2015-2021 Altova GmbH

kind	文字列	ファセットの名前、例えば、"minLength" または "enumeration"
value	ファセットによる	ファセットの値
fixed	ブール値	
typed-value	ファセットの列挙のみ、Array(xs:anyAtomicType	列挙値を含む配列。それぞれがアトミック値のシーケンスである場合があります。(メモ 列挙ファセットに関しては、実際の型にかかわらず "value" 値プロパティは文字列のシーケンスです)

Identity Constraint

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Identity-Constraint Definition"
name	文字列	制約のローカル名
target namespace	文字列	制約の名前空間 URI
identity-constraint category	文字列 ("key"|"unique"|"keyRef")	
selector	XPath プロパティレコード	
fields	XPath プロパティレコード のシーケンス	
referenced key	(keyRef のためのみ): Identity Constraint	対応するキー制約

Model Group

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group"
compositor	文字列 ("sequence"|"choice"|"all")	
particles	Particle のシーケンス	

Model Group Definition

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group Definition"
name	文字列	モデルグループのローカル名
target namespace	文字列	モデルグループの名前空間 URI
model group	Model Group	

Notation

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Notation Declaration"
name	文字列	表記のローカル名

target namespace	文字列	表記の名前空間 URI
system identifier	anyURI	
public identifier	文字列	

- Particle

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Particle"
min occurs	整数	
max occurs	整数または文字列("unbounded")	
term	Element Declaration、Element Wildcard、または ModelGroup	

- Simple Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Simple Type Definition"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
final	文字列のシーケンス ("restriction"|"extension"|"list"|"union")	
context	含まれるコンポーネント	
base type definition	Simple Type	
facets	Sequence of Facet	
fundamental facets	(実装されていない)空のシーケンス	
variety	文字列 ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(リスト型のみのための) Simple Type	
member type definitions	(ユニオン型のための) Simple Type のシーケンス	

- Type Alternative

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Type Alternative"
test	XPath プロパティレコード	
type definition	Simple Type または Complex Type	

- XPath Property Record

プロパティ名	プロパティ型	プロパティ値

(C) 2015-2021 Altova GmbH

namespace bindings	プロパティを持つ関数のシーケンス("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	XPath 式の静的ベースURI
expression	文字列	文字列としてのXPath 式

▼ Schema (2つの引数)

altova:schema(ComponentKind *as xs:string*, **Name** *as xs:QName*) **as (function(xs:string) as item()*)?** **XP3.1** **XQ3.1**

2番目の引数内で与えられている名前と同じ名前を持つ最初の引数内で指定されているコンポーネントの型を返します。コンポーネントのプロパティの一つを選択して更にナビゲートすることができます。

- このプロパティがコンポーネントの場合、このコンポーネントのプロパティの1つを選択して更に深く他のステップをナビゲートすることができます。このステップは更にスキーマをナビゲートするために繰り返すことができます。
- コンポーネントが動的な値の場合、動的な値が返され更にナビゲートすることはできません。

メモ Xquery 式ではスキーマは明示的にインポートされる必要があります。XPath 式では、スキーマは処理環境にインポートされる必要があります。例えば、XSLT には**xslt:import** 命令を使用してインポートします。

サンプル

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  altova:schema("element declaration", xs:QName("OrgChart"))("type definition")
  ("content type")("particles")[3]!.("term")("kind")
  ```
  は3番目の`particles` コンポーネントの `kind` プロパティを返します。 この`particles` コンポーネントは`OrgChart` の `Qname` を持つ要素宣言の子孫です。

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  let $typedef := altova:schema("type definition", xs:QName("emailType"))
  for $facet in $typedef ("facets")
  return [$facet ("kind"), $facet("value")]
  ```
  は各 **emailType** コンポーネントの各 **facet** にそのファセットの型と値を含む列挙を返します。

コンポーネントとそのプロパティ

Assertion

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Assertion"
test	XPath プロパティレコード	

Attribute Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Declaration"
name	文字列	属性のローカル名

target namespace	文字列	属性の名前空間 URI
type definition	Simple Type または Complex Type	
scope	プロパティを持つ関数 ("class":"Scope", "variety": "global" or "local", "parent": the containing Complex Type or Attribute Group)	
value constraint	存在する場合、プロパティを持つ関数 ("class": "Value Constraint"、"variety": "fixed" または "default", "value": atomic value, "lexical form": string。"value" プロパティは namespace-sensitive 型のために使用することはできないことにご注意してください。	
inheritable	ブール値	

Attribute Group Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Group Definition"
name	文字列	属性グループのローカル名
target namespace	文字列	属性グループの名前空間 URI
attribute uses	(Attribute Use) のシーケンス	
attribute wildcard	任意の属性のワイルドカード	

Attribute Use

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Use"
required	ブール値	属性が必要な場合は true 、任意の場合は false
value constraint	Attribute Declaration を参照	
inheritable	ブール値	

Attribute Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 ("strict"|"lax"|"skip")	

(C) 2015-2021 Altova GmbH

- Complex Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
base type definition	Complex Type Definition	
final	文字列のシーケンス ("restriction"|"extension")	
context	空のシーケンス(not implemented)	
derivation method	文字列 ("restriction"|"extension")	
abstract	ブール値	
attribute uses	Attribute Use のシーケンス	
attribute wildcard	任意の属性のワイルドカード	
content type	プロパティを持つ関数: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	文字列のシーケンス ("restriction"|"extension")	
assertions	アサーションのシーケンス	

- Element Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
type definition	Simple Type または Complex Type	
type table	プロパティを持つ関数 ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Simple Type or Complex Type)	
scope	プロパティを持つ関数 ("class":"Scope", "variety": ("global"|"local"), "parent": optional Complex Type)	
value constraint	Attribute Declaration を参照	
nillable	ブール値	
identity-constraint	Identity Constraint のシーケンス	

definitions		
substitution group affiliations	Element Declaration のシーケンス	
substitution group exclusions	文字列のシーケンス ("restriction"|"extension")	
disallowed substitutions	文字列のシーケンス ("restriction"|"extension"|"substitution")	
abstract	ブール値	

Element Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 ("strict"|"lax"|"skip")	

Facet

プロパティ名	プロパティ型	プロパティ値
kind	文字列	ファセットの名前、例えば、"minLength" または"enumeration"
value	ファセットによる	ファセットの値
fixed	ブール値	
typed-value	ファセットの列挙のみ、Array(xs:anyAtomicType	列挙値を含む配列。それぞれがアトミック値のシーケンスである場合があります。(メモ 列挙ファセットに関しては、実際の型にかかわらず "value" 値プロパティは文字列のシーケンスです)

Identity Constraint

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Identity-Constraint Definition"
name	文字列	制約のローカル名
target namespace	文字列	制約の名前空間 URI
identity-constraint category	文字列 ("key"|"unique"|"keyRef")	
selector	XPath プロパティレコード	
fields	XPath プロパティレコード のシーケンス	

(C) 2015-2021 Altova GmbH

referenced key	(keyRef のためのみ): Identity Constraint	対応するキー制約

- Model Group

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group"
compositor	文字列 ("sequence"|"choice"|"all")	
particles	Particle のシーケンス	

- Model Group Definition

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group Definition"
name	文字列	モデルグループのローカル名
target namespace	文字列	モデルグループの名前空間 URI
model group	Model Group	

- Notation

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Notation Declaration"
name	文字列	表記のローカル名
target namespace	文字列	表記の名前空間 URI
system identifier	anyURI	
public identifier	文字列	

- Particle

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Particle"
min occurs	整数	
max occurs	整数または文字列("unbounded")	
term	Element Declaration、Element Wildcard、または ModelGroup	

- Simple Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Simple Type Definition"
name	文字列	(匿名の場合は空) 型のローカル名
target namespace	文字列	(匿名の場合は空) 型の名前空間 URI
final	文字列のシーケンス	

	("restriction"|"extension"|"list"|"union")	
context	含まれるコンポーネント	
base type definition	Simple Type	
facets	Sequence of Facet	
fundamental facets	(実装されていない) 空のシーケンス	
variety	文字列 ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(リスト型のみのための) Simple Type	
member type definitions	(ユニオン型のための) Simple Type のシーケンス	

Type Alternative

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Type Alternative"
test	XPath プロパティレコード	
type definition	Simple Type または Complex Type	

XPath Property Record

プロパティ名	プロパティ型	プロパティ値
namespace bindings	プロパティを持つ関数のシーケンス ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	XPath 式の静的ベース URI
expression	文字列	文字列としての XPath 式

型

altova:type(Node *as item?*) **as (function(xs:string) as item()*)? XP3.1 XQ3.1**
関数 **altova:type** は XML ドキュメントの要素または属性ノードを送信しノードの型情報を PSVI から返します。

メモ スキーマが参照可能になるために XML ドキュメントはスキーマ宣言を持つ必要があります。

サンプル

```
for $element in //Email
let $type := altova:type($element)
return $type
```
はノードの型情報を含む関数を返します。

```
for $element in //Email
let $type := altova:type($element)
return $type ("kind")
```

はノードの型のコンポーネント（単純型または複合型）を取りコンポーネントの`kind` property の値を返します。

コンポーネントとそのプロパティ

- Assertion

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Assertion"
test	XPath プロパティレコード	

- Attribute Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Declaration"
name	文字列	属性のローカル名
target namespace	文字列	属性の名前空間 URI
type definition	Simple Type または Complex Type	
scope	プロパティを持つ関数（"class": "Scope", "variety": "global" or "local", "parent": the containing Complex Type or Attribute Group)	
value constraint	存在する場合、プロパティを持つ関数（"class": "Value Constraint"、"variety": "fixed" または "default", "value": atomic value, "lexical form": string。"value" プロパティは namespace-sensitive 型のために使用することはできないことにご注意してください。	
inheritable	ブール値	

- Attribute Group Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Group Definition"
name	文字列	属性グループのローカル名
target namespace	文字列	属性グループの名前空間 URI
attribute uses	（Attribute Use）のシーケンス	
attribute wildcard	任意の属性のワイルドカード	

- Attribute Use

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Use"
required	ブール値	属性が必要な場合はtrue 、任意の場合はfalse

value constraint	Attribute Declaration を参照	
inheritable	ブール値	

- Attribute Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 ("strict"|"lax"|"skip")	

- Complex Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
base type definition	Complex Type Definition	
final	文字列のシーケンス ("restriction"|"extension")	
context	空のシーケンス(not implemented)	
derivation method	文字列 ("restriction"|"extension")	
abstract	ブール値	
attribute uses	Attribute Use のシーケンス	
attribute wildcard	任意の属性のワイルドカード	
content type	プロパティを持つ関数: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	文字列のシーケンス ("restriction"|"extension")	
assertions	アサーションのシーケンス	

- Element Declaration

(C) 2015-2021 Altova GmbH

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
type definition	Simple Type または Complex Type	
type table	プロパティを持つ関数 ("class": "Type Table", "alternatives": sequence of Type Alternative, "default type definition": Simple Type or Complex Type)	
scope	プロパティを持つ関数 ("class": "Scope", "variety": ("global"|"local"), "parent": optional Complex Type)	
value constraint	Attribute Declaration を参照	
nillable	ブール値	
identity-constraint definitions	Identity Constraint のシーケンス	
substitution group affiliations	Element Declaration のシーケンス	
substitution group exclusions	文字列のシーケンス ("restriction"|"extension")	
disallowed substitutions	文字列のシーケンス ("restriction"|"extension"|"substitution")	
abstract	ブール値	

- Element Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 ("strict"|"lax"|"skip")	

- Facet

プロパティ名	プロパティ型	プロパティ値
kind	文字列	ファセットの名前、例えば、"minLength" または "enumeration"
value	ファセットによる	ファセットの値
fixed	ブール値	

typed-value	ファセットの列挙のみ、Array(xs:anyAtomicType	列挙値を含む配列。それぞれがアトミック値のシーケンスである場合があります。(メモ 列挙ファセットに関しては、実際の型にかかわらず "value" 値プロパティは文字列のシーケンスです)

Identity Constraint

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Identity-Constraint Definition"
name	文字列	制約のローカル名
target namespace	文字列	制約の名前空間 URI
identity-constraint category	文字列 ("key"|"unique"|"keyRef")	
selector	XPath プロパティレコード	
fields	XPath プロパティレコード のシーケンス	
referenced key	(keyRef のためのみ): Identity Constraint	対応するキー制約

Model Group

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group"
compositor	文字列 ("sequence"|"choice"|"all")	
particles	Particle のシーケンス	

Model Group Definition

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group Definition"
name	文字列	モデルグループのローカル名
target namespace	文字列	モデルグループの名前空間 URI
model group	Model Group	

Notation

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Notation Declaration"
name	文字列	表記のローカル名
target namespace	文字列	表記の名前空間 URI
system identifier	anyURI	
public identifier	文字列	

Particle

(C) 2015-2021 Altova GmbH

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Particle"
min occurs	整数	
max occurs	整数または文字列("unbounded")	
term	Element Declaration、Element Wildcard、または ModelGroup	

Simple Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Simple Type Definition"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
final	文字列のシーケンス ("restriction"|"extension"|"list"|"union")	
context	含まれるコンポーネント	
base type definition	Simple Type	
facets	Sequence of Facet	
fundamental facets	(実装されていない)空のシーケンス	
variety	文字列 ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(リスト型のみのための) Simple Type	
member type definitions	(ユニオン型のための) Simple Type のシーケンス	

Type Alternative

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Type Alternative"
test	XPath プロパティレコード	
type definition	Simple Type または Complex Type	

XPath Property Record

プロパティ名	プロパティ型	プロパティ値
namespace bindings	プロパティを持つ関数のシーケンス("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	XPath 式の静的ベース URI
expression	文字列	文字列としての XPath 式

30.2.1.7 XPath/XQuery 関数: シーケンス

Altova のシーケンス拡張関数はXPath とXQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の**XPath 3.0** および**XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova 拡張関数名前空間、`http://www.altova.com/xslt-extensions` に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3.1

▼ attributes [altova:]

altova:attributes(AttributeName *as xs:string*) を**attribute()*** とする XP3.1 XQ3.1
入力引数 AttributeName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、attribute:: 軸に対して行われます。これは、コンテキストノードが親要素ノードである必要があることを意味します。

サンプル

- **altova:attributes**("MyAttribute") は MyAttribute()* を返します。

altova:attributes(AttributeName *as xs:string*, **SearchOptions** *as xs:string*) **asattribute()*** XP3.1 XQ3.1
入力引数 AttributeName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、attribute:: 軸に対して行われます。コンテキストノードが親要素ノードである必要があります。第 2 引数はオプションのフラグを含みます。使用することのできるフラグは以下の通りです:
r = 正規表現検索を切り替えます; AttributeName は正規表現検索文字列である必要があります;
f = このオプションが指定されている場合、AttributeName は完全一致を提供します。それ以外の場合は、AttributeName は属性名に部分的に一致するとその属性を返します。例えば **f** が指定されていない場合、MyAtt は MyAttribute を返します。
i = 大文字と小文字を一致させる検索に切り替えます。
p = 検索に名前空間プレフィックスを含みます。AttributeName は、名前空間プレフィックスを含みます。例えば altova:MyAttribute。
フラグは順序に関わりなく書き込むことができます。無効なフラグはエラーを生成します。1 つまたは複数のフラグを省略することができます。空の文字列は許可されていますが、引数を1 つしか持たない関数と同じ効果を持ちます(*前の署名*)、ですが、空のシーケンスは第 2 の引数としては許可されていません。

サンプル

- **altova:attributes**("MyAttribute", "rfip") は MyAttribute()* を返します。

- **altova:attributes**("MyAttribute", "pri") は MyAttribute()* を返します。
- **altova:attributes**("MyAtt", "rip") は MyAttribute()* を返します。
- **altova:attributes**("MyAttributes", "rfip") は不一致を返します。
- **altova:attributes**("MyAttribute", "") は MyAttribute()* を返します。
- **altova:attributes**("MyAttribute", "Rip") 認識されないフラグエラーを返します。
- **altova:attributes**("MyAttribute",) 見つからない第 2 引数を返します。

elements [altova:]

altova:elements(ElementName *as xs:string*) を **element()*** とする **XP3.1** **XQ3.1**
入力引数 ElementName で与えられた名前と同じローカル名を持つすべての要素を返します。検索は大文字と小文字を区別して child:: 軸に対して実行されます。コンテキストノードは、検索される要素の親ノードである必要があります。

サンプル

- **altova:elements**("MyElement") は MyElement()* を返します。

altova:elements(ElementName *as xs:string*, **SearchOptions** *as xs:string*) **aselement()*** **XP3.1** **XQ3.1**
入力引数 ElementName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、child:: 軸に対して行われます。コンテキストノードが親要素ノードである必要があります。第 2 引数はオプションのフラグを含みます。使用することのできるフラグは以下の通りです:
r = 正規表現検索を切り替えます; ElementName は正規表現検索文字列である必要があります;
f = このオプションが指定されている場合、ElementName は完全一致を提供します。それ以外の場合は、ElementName は属性名に部分的に一致するとその属性を返します。例えば **f** が指定されていない場合、MyElem は MyElement を返します。
i = 大文字と小文字を一致させる検索に切り替えます。
p = 検索に名前空間プレフィックスを含みます。ElementName は、名前空間プレフィックスを含みます。例えば altova:MyElement。
フラグは順序に関わりなく書き込むことができます。無効なフラグはエラーを生成します。1 つまたは複数のフラグを省略することができます。空の文字列は許可されていますが、引数を 1 つしか持たない関数と同じ効果を持ちます(*前の署名*)、ですが、空のシーケンスは第 2 の引数としては許可されていません。

サンプル

- **altova:elements**("MyElement", "rip") は MyElement()* を返します。
- **altova:elements**("MyElement", "pri") は MyElement()* を返します。
- **altova:elements**("MyElement", "") は MyElement()* を返します。
- **altova:attributes**("MyElem", "rip") は MyElement()* を返します。
- **altova:attributes**("MyElements", "rfip") 不一致を課wします。
- **altova:elements**("MyElement", "Rip") 認識されないフラグエラーを返します。
- **altova:elements**("MyElement",) 見つからない第 2 引数を返します。

find-first [altova:]

altova:find-first((Sequence *as item()**), **(Condition(Sequence-Item** *as xs:boolean*))) を **item()?** とする **XP3.1** **XQ3.1**
この関数は2 つの引数を必要とします。最初の引数は1 つ、または1 つ以上のデータ型のアイテムのシーケンスです。第 2 の引数 Condition は、(1 のアリティを持つ) 1 つの引数 を必要とし、boolean を返す XPath 関数に対する参照です。Condition で参照された関数の代わりに、**Sequence** の各アイテムが提出されます。(*注意:* この関数は1 つの引数のみを必要とします。) **Condition** 内の関数に **true()** と評価させる最初の **Sequence** アイテムは、**altova:find-first**、反復の終了の結果として返されます。

サンプル

- **altova:find-first**(5 to 10, function($a) {$a mod 2 = 0}) は xs:integer 6 を返します。**Condition** 引数は、**$a** という名のインライン関数を宣言し、定義します。XPath 3.0 インライン関数 **function()** を参照します。**altova: Sequence** 引数内の各アイテムでは、**find-first** がパスされ、代わりに、**$a** を入力値とします。入力値は関数定義($a mod 2 = 0)内の条件に対してテストされます。この条件を満たす最初の入力値が **altova:find-first**（この場合は 6）の結果として返されます。

- **altova:find-first**((1 to 10), (function($a) {$a+3=7})) は xs:integer 4 を返します。

更なるサンプル
ファイル **C:\Temp\Customers.xml** が存在する場合:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は xs:string C:\Temp\Customers.xml を返します。

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** が存在する場合:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は xs:string http://www.altova.com/index.html を返します。

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** も存在しない場合:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) 結果無しを返します。

上のサンプルについての注意点

- XPath 3.0 関数 doc-available は URI として使用され、ドキュメントノードが提出された URI で検出される場合 true を返する単一の引数を必要とします。（ですから、提出された URI でのドキュメントは XML ドキュメントである必要があります。）
- doc-available 関数は、altova:find-first の第 2 引数である、**Condition** で使用することができます。これは、1 つの引数（アリティ=1）のみを必要とするからであり、item() を入力（ URI として使用される文字列）として、boolean の値を返すからです。
- doc-available 関数は、参照されているだけで、呼び出されていない点に注意してください。アタッチされている #1 サフィックスは関数が 1 つのアリティであることを表示するためです。doc-available#1 の意味は以下のとおりです: アリティ=1 を持つ doc-availabe() 関数を使用し、最初のシーケンスの各アイテムの代わりに、単一引数としてパスします。この結果、2 つの文字列の各自つは、文字列を URI として使用し、URI にドキュメントノードが存在するかテストする **doc-available()** にパスされます。1 つが各当する場合、**doc-available()** 関数は true() を評価し、シーケンス内のその文字列のインデックス ポジションは、**altova:find-first** 関数の結果として返されます。doc-available() 関数に関しての注意点: 相対パスは、デフォルトで関数がロードされる XML ドキュメントの現在のベース URI に対して相対的に解決されます。

▼ find-first-combination [altova:]

altova:find-first-combination((Seq-01 *as item()**), **(Seq-02** *as item()**), **(Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean*)) を **item()*** とする **XP3.1** **XQ3.1**
この関数は 3 つの引数を必要とします:

- 最初の2 つの引数 Seq-01 とSeq-02, は、1 つまたは1 つ以上のデータ型のアイテムです。
- 第 3 の引数 Condition は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 とSeq-02 のアイテムは指定された組み合わせ(各シーケンスからの1 つずつにアイテムで構成されるペア)で、Condition 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X1 Y2), (X1 Y3) ... (X1 Yn), (X2 Y1), (X2 Y2) ... (Xn Yn)
```

Condition 関数にtrue() と評価するように指示する最初のペアはaltova:find-first-combination の結果として返されます。以下の点に注意してください: (i) Condition 関数が提出された引数ペア内で繰り返され、true() を評価しない場合、altova:find-first-combination は、結果を返しません; (ii) altova:find-first-combination の結果が常に(データ型の)アイテムのペアである場合またはアイテムでない場合。

サンプル

- altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) はxs:integers (11, 21) のシーケンスを返します。
- altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) はxs:integers (11, 22) のシーケンを返します。
- altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34}) は xs:integers (11, 23) のシーケンを返します。

find-first-pair [altova:]

altova:find-first-pair((Seq-01 *as item()**), (Seq-02 *as item()**), (Condition(Seq-01-Item, Seq-02-Item *as xs:boolean*)) をitem()* とする XP3.1 XQ3.1
この関数は3 つの引数を必要とします:

- 最初の2 つの引数 Seq-01 とSeq-02, は、1 つまたは1 つ以上のデータ型のアイテムです。
- 第 3 の引数 Condition は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 とSeq-02 のアイテムは指定された組み合わせで、Condition 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Condition 関数にtrue() と評価するように指示する最初のペアはaltova:find-first-pair の結果として返されます。以下の点に注意してください: (i) Condition 関数が提出された引数ペア内で繰り返され、true() を評価しない場合、altova:find-first-pair は、結果を返しません; (ii) altova:find-first-combination の結果が常に(データ型の) アイテムのペアである場合またはアイテムでない場合。

サンプル

- altova:find-first-pair(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) は xs:integers (11, 21) のシーケンスを返します。
- altova:find-first-pair(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) は*結果無し*を返します。

上の2 つのサンプルに表示される通り、ペアの順序は以下の通りです: (11, 21) (12, 22) (13, 23)...(20,

30) 。(33 を返す指示されたペアがないため) この理由で第 2 のサンプルは *結果無し*を返します。

find-first-pair-pos [altova:]

altova:find-first-pair-pos((Seq-01 *as item()**)**,* **(Seq-02** *as item()**)**,* **(Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean*)) を **xs:integer** とする **XP3.1** **XQ3.1**
この関数は3 つの引数を必要とします:

- 最初の2 つの引数 **Seq-01** と**Seq-02**, は、1 つまたは1 つ以上のデータ型のアイテムです。
- 第 3 の引数 **Condition** は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 と**Seq-02** のアイテムは指定された組み合わせで、**Condition** 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Condition 関数に**true()**を評価させる、最初に指示されたペアのインデックス ポジションは、は**altova:find-first-pair-pos** の結果として返されます。関数が提出された引数ペア内で繰り返され、**true()**を一度も評価しない場合、**altova:find-first-pair-pos** *結果無し*が返します。

サンプル

- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) は 1 を返します
- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) は*結果無し*を返します。

上の2 つのサンプルに表示される通り、ペアの順序は以下の通りです: (11, 21) (12, 22) (13, 23)...(20, 30)。最初のサンプルでは、最初のペアは**Condition** 関数に**true()**を評価させ、シーケンス内のインデックス ポジションには**1** が返されます。第 2 のサンプルでは、 33 を返すペアがないため、*結果無し*が返します。

find-first-pos [altova:]

altova:find-first-pos((Sequence *as item()**)**,* **(Condition(Sequence-Item** *as xs:boolean*)) を **xs:integer** とする **XP3.1** **XQ3.1**
この関数は2 つの引数を必要とします。最初の引数は1 つ、または1 つ以上のデータ型のアイテムのシーケンスです。第 2 の引数 Condition は、(1 のアリティを持つ) 1 つの引数 を必要とし、boolean を返す XPath 関数に対する参照です。Condition で参照された関数の代わりに、**Sequence** の各アイテムが提出されます。(*注意:* この関数は1 つの引数のみを必要とします。) **Condition** 内の関数に**true()**と評価させる最初の **Sequence** アイテムは、**altova:find-first-pos** の結果として返された**Sequence** 内インデックス ポジションを持ちます。

サンプル

- **altova:find-first-pos**(5 to 10, function($a) {$a mod 2 = 0}) は xs:integer 2 を返します。

Condition 引数は、**$a** という名のインライン関数を宣言し、定義します。XPath 3.0 インライン関数 **function()**を参照します。 **Sequence** 引数内の各アイテムでは、**find-first-pos** がパスされ、代わりに、**$a** を入力値とします。入力値は関数定義($a mod 2 = 0)内の条件に対してテストされます。この条件を満たす最初の入力値が **altova:find-first-pos** ((シーケンス内で条件を満たす最初の値である6 がシーケンスのインデクス位置 2 にあるためこの場合は2、) の結果として返されます。

- **altova:find-first-pos**((2 to 10), (function($a) {$a+3=7})) は xs:integer 3 を返します。

更なるサンプル
ファイル **C:\Temp\Customers.xml** が存在する場合:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は 1 を返します。

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** が存在する場合:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は 2 を返します。

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** も存在しない場合:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) 結果無しを返します。

上のサンプルについての注意点

- XPath 3.0 関数 doc-available は URI として使用され、ドキュメントノードが提出された URI で検出される場合 true を返する単一の引数を必要とします。(ですから、提出された URI でのドキュメントは XML ドキュメントである必要があります。)
- doc-available 関数は、altova:find-first-pos の第 2 引数である、**Condition** で使用することができます。これは、1 つの引数 (アリティ=1) のみを必要とするからであり、item() を入力 (URI として使用される文字列) として、boolean の値を返すからです。
- doc-available 関数は、参照されているだけで、呼び出されていない点に注意してください。アタッチされている #1 サフィックスは関数が 1 つのアリティであることを表示するためです。doc-available#1 の意味は以下のとおりです: アリティ=1 を持つ doc-availabe() 関数を使用し、最初のシーケンスの各アイテムの代わりに単一引数として パスします。この結果、2 つの文字列の各自つは、文字列を URI として使用し、URI にドキュメントノードが存在するかテストする **doc-available()** に パスされます。1 つが各当する場合、**doc-available()** 関数は true() を評価し、シーケンス内のその文字列のインデックス ポジションは、**altova:find-first-pos** 関数の結果として返されます。doc-available() 関数に関しての注意点: 相対パスは、デフォルトで関数がロードされる XML ドキュメントの現在のベース URI に対して相対的に解決されます。

for-each-attribute-pair [altova:]

altova:for-each-attribute-pair(Seq1 *as element()?*, **Seq2** *as element()?*, **Function** *as function()*) **asitem()*** **XP3.1** **XQ3.1**
ペアの一つの属性が最初の要素を取得し、もう一つの属性が 2 番目の要素から取得された箇所で、最初の 2 つの引数は、2 つの要素、および、属性ペアを作成するために使用された属性を識別します。要素ペアは同じ名前を持つことをベースに選択され、ペアは名前別に アルファベット順にセットに並べ替えられます。属性のために他の要素上に対応する属性が存在しない場合、ペアのジョイントは解除されます。これは、一つのメンバーにより飲み構成されることを意味します。関数アイテム (3 番目の引数 Function) はアイテムのシーケンスである出力であるペアのシーケンス内のペア (ジョイント、ジョイントの解除) に個別に適用されます。

サンプル

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, function($a, $b){$a+b}) は以下 を返します。...

```
(2, 4, 6) if
<Test-A att1="1" att2="2" att3="3" />
<Test-B att1="1" att2="2" att3="3" />

(2, 4, 6) if
<Test-A att2="2" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />

(2, 6) if
<Test-A att4="4" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />
```

メモ: 結果 (2, 6) は次のアクションにより取得されます: (1+1, ()+2, 3+3, 4+()) 。演算子の一つが空のシーケンスの場合、アイテム 2 と 4 の場合同様、結果は空のシーケンスになります。

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, concat#2) は以下を返します。

```
(11, 22, 33) if
<Test-A att1="1" att2="2" att3="3" />
<Test-B att1="1" att2="2" att3="3" />

(11, 2, 33, 4) if
<Test-A att4="4" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />
```

for-each-combination [altova:]

altova:for-each-combination(FirstSequence *as item()**, **SecondSequence** *as item()**, **function($i,$j){$i || $j})** を **item()*** とする **XP3.1** **XQ3.1**
第 3 の引数として与えられた、インライン関数を返します。FirstSequence 内の各 $i を SecondSequence内の各 $j と結合します。出力はこれらのアイテムのシーケンスです。

サンプル

- **altova:for-each-combination**(('a', 'b', 'c'), ('1', '2', '3'), function($i, $j) {$i || $j}) は ('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3') を返します。

for-each-combination [altova:]

altova:for-each-combination(FirstSequence *as item()**, **SecondSequence** *as item()**, **Function($i,$j){$i || $j}) asitem()*** **XP3.1** **XQ3.1**
最初の2 つの引数内の2 つのシーケンスのアイテムは最初のシーケンスの各アイテムが2 番目のシーケンスになるように結合されるように結合されます。3 番目の引数として与えられる関数は結果のシーケンス内の組み合わせに適用され、アイテムのシーケンスである出力内で出力されます（*サンプル参照*）。

サンプル

- **altova:for-each-combination**(('a', 'b', 'c'), ('1', '2', '3'), function($i, $j) {$i || $j}) は ('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3') を返します。

▼ substitute-empty [altova:]

altova:substitute-empty(FirstSequence *as item()**, **SecondSequence** *as item()*) を **item()*** とする XP3.1 XQ3.1
FirstSequence が空の場合、SecondSequence を返します。FirstSequence が空でない場合、FirstSequence を返します。

サンプル

- **altova:substitute-empty**((1,2,3), (4,5,6)) は (1,2,3) を返します。
- **altova:substitute-empty**((), (4,5,6)) は (4,5,6) を返します。

30.2.1.8 XPath/XQuery 関数: 文字列

Altova の文字列拡張関数は XPath と XQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の **XPath 3.0** および **XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数は XPath/XQuery 式で使用することができ、XPath、XQuery、および XSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は **Altova 拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内の XPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内の XPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内の XQuery 式で使用):	XQ1 XQ3.1

▼ camel-case [altova:]

altova:camel-case(InputString *as xs:string*) を **xs:string** とする XP3.1 XQ3.1
入力文字列を **InputString** をキャメルケースで返します。文字列は空白スペースのショートカットである) 正規表現 **'\s'** を使用して分析されます。空白文字または連続する空白文字のシーケンスの後の最初の非空白スペース文字は大文字です。出力文字列の最初の文字は大文字です。

サンプル

- **altova:camel-case**("max") Max を返します。
- **altova:camel-case**("max max") Max Max を返します。
- **altova:camel-case**("file01.xml") File01.xml を返します。
- **altova:camel-case**("file01.xml file02.xml") File01.xml File02.xml を返します。
- **altova:camel-case**("file01.xml file02.xml") File01.xml File02.xml を返します。
- **altova:camel-case**("file01.xml -file02.xml") File01.xml -file02.xml を返します。

altova:camel-case(InputString *as xs:string*, **SplitChars** *as xs:string*, **IsRegex** *as xs:boolean*) を **xs:string** とする XP3.1 XQ3.1
SplitChars を使用して、次の大文字をトリガーする文字を決定し、入力文字列を **InputString** キャメルケースに変換しま

す。 SplitChars は、IsRegex = true() の場合、または、IsRegex = false() の場合、プレーン文字は正規表現として使用されます。出力文字列の最初の文字は大文字です。

サンプル

- altova:camel-case("setname getname", "set|get", true()) setName getName を返します。
- altova:camel-case("altova\documents\testcases", "\", false()) Altova\Documents\Testcases を返します。

char [altova:]

altova:char(Position *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
xs:string.Position に対してのコンテキストアイテムの値を変換することにより得られた文字列内の Position 引数により指定されたポジションにある文字を含む文字列を返します。引数により提出されたインデックスに文字が存在しない場合、結果文字列は空です。

サンプル

コンテキスト アイテムが 1234ABCD の場合:

- altova:char(2) は 2 を返します。
- altova:char(5) は A を返します。
- altova:char(9) は空の文字列を返します。
- altova:char(-2) は空の文字列を返します。

altova:char(InputString *as xs:string*, **Position** *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
引数として提出された文字列内の Position 引数により指定されたポジションでの文字を含む文字列を返します。Position 引数により提出されたインデックスに文字が存在しない場合、結果文字列は空です。

サンプル

- altova:char("2014-01-15", 5) は - を返します。
- altova:char("USA", 1) は U を返します。
- altova:char("USA", 10) は空の文字列を返します。
- altova:char("USA", -2) は空の文字列を返します。

create-hash-from-string[altova:]

altova:create-hash-from-string(InputString *as xs:string*) **asxs:string** **XP2** **XQ1** **XP3.1** **XQ3.1**
altova:create-hash-from-string(InputString *as xs:string*, **HashAlgo** *as xs:string*) **asxs:string** **XP2** **XQ1** **XP3.1** **XQ3.1**
HashAlgo 引数により指定されているハッシュアルゴリズムを使用して InputString からハッシュ文字列を生成します。次のハッシュアルゴリズムは(大文字、または、小文字で)指定されている可能性があります: **MD5**, **SHA-224**, **SHA-256**, **SHA-384**, **SHA-512**。(最初の署名を参照してください) 2番目の引数が指定されていない場合、**SHA-256** ハッシュアルゴリズムが使用されます。

サンプル

- altova:create-hash-from-string('abc') は **SHA-256** ハッシュアルゴリズムを使用して生成されたハッシュ文字列 を返します。
- altova:create-hash-from-string('abc', 'md5') は **MD5** ハッシュアルゴリズムを使用して生成されたハッシュ文字列 を返します。
- altova:create-hash-from-string('abc', 'MD5') は **MD5** ハッシュアルゴリズムを使用して生成された

ハッシュ文字列 を返します。

first-chars [altova:]

altova:first-chars(X-Number *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
xs:string に対するコンテキストアイテムの値を変換することにより得られた最初の X-Number 文字を含む文字列を返します。

サンプル

コンテキストアイテムが **1234ABCD** の場合:

- **altova:first-chars**(2) は 12 を返します。
- **altova:first-chars**(5) は 1234A を返します。
- **altova:first-chars**(9) は 1234ABCD を返します。

altova:first-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
InputString 引数として提出された文字列の最初の文字を含む文字列を返します。

サンプル

- **altova:first-chars**("2014-01-15", 5) は 2014- を返します。
- **altova:first-chars**("USA", 1) は U を返します。

format-string [altova:]

altova:format-string(InputString *as xs:string*, **FormatSequence** *as item()**) **as xs:string** **XP3.1** **XQ3.1**
入力文字列（最初の引数）には、配置パラメーター（%1, %2, etc）が含まれています。各パラメーターは、（2番目の引数として提出されている）フォーマットシーケンス内の対応するポジションでロケートされる文字列アイテムと置き換えられます。フォーマットシーケンス内の最初のアイテムは、配置パラメーター %1 を置換、二番目のアイテムは %2 を置き換えます。関数は、書式設定された代替を持つ文字列を返します。配置パラメーターに文字列が存在しない場合は、配置パラメーターが返されます。これは、配置パラメーターのインデックスが書式シーケンス内のアイテムの数より大きい場合発生します。

サンプル

- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe')) は "Hello Jane, John, Joe" を返します。
- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe', 'Tom')) は"Hello Jane, John, Joe" を返します。
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe', 'Tom')) は "Hello Jane, John, Tom" を返します。
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe')) は "Hello Jane, John, %4" を返します。

last-chars [altova:]

altova:last-chars(X-Number *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
xs:string に対してのコンテキストアイテムの値の変換より取得された文字列の最後の X-Number 文字を含んでいる文字列を返します。

サンプル

コンテキストアイテムが **1234ABCD** の場合:

- **altova:last-chars**(2) は CD を返します。
- **altova:last-chars**(5) は 4ABCD を返します。
- **altova:last-chars**(9) は 1234ABCD を返します。

altova:last-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) **asxs:string** とする
XP3.1 **XQ3.1**
引数として提出された文字列の最後の X-Number 文字を含んでいる文字列を返します。

サンプル

- **altova:last-chars**("2014-01-15", 5) は 01-15 を返します。
- **altova:last-chars**("USA", 10) は USA を返します。

pad-string-left [altova:]

altova:pad-string-left(StringToPad *as xs:string*, **StringLength** *as xs:integer*, **PadCharacter** *as xs:string*) を**xs:string** とする **XP3.1** **XQ3.1**
PadCharacter 引数は1文字です。文字列の左側にパドされ、この数が、StringLength 引数の整数の値と等しくなるようにStringToPad の文字の数を増やします。StringLength 引数は任意の整数の値（正数または負数）を持つことができますが、パディングは、StringLength の値がStringToPad 内の文字数よりも多い場合の場合のみ発生します。もし、StringToPad がStringLength の値よりも多くの文字数も持つ場合、StringToPad は変更されません。

サンプル

- **altova:pad-string-left**('AP', 1, 'Z') は 'AP' を返します。
- **altova:pad-string-left**('AP', 2, 'Z') は 'AP' を返します。
- **altova:pad-string-left**('AP', 3, 'Z') は 'ZAP' を返します。
- **altova:pad-string-left**('AP', 4, 'Z') は 'ZZAP' を返します。
- **altova:pad-string-left**('AP', -3, 'Z') は 'AP' を返します。
- **altova:pad-string-left**('AP', 3, 'YZ') は[パド文字が長すぎます] エラーを返します。

pad-string-right [altova:]

altova:pad-string-right(StringToPad *as xs:string*, **StringLength** *as xs:integer*, **PadCharacter** *as xs:string*) を**xs:string** とする **XP3.1** **XQ3.1**
PadCharacter 引数は1文字です。文字列の右側にパドされ、この数が、StringLength 引数の整数の値と等しくなるようにStringToPad の文字の数を増やします。StringLength 引数は任意の整数の値（正数または負数）を持つことができますが、パディングは、StringLength の値がStringToPad 内の文字数よりも多い場合の場合のみ発生します。もし、StringToPad がStringLength の値よりも多くの文字数も持つ場合、StringToPad は変更されません。

サンプル

- **altova:pad-string-right**('AP', 1, 'Z') を 'AP' を返します。
- **altova:pad-string-right**('AP', 2, 'Z') を 'AP' を返します。
- **altova:pad-string-right**('AP', 3, 'Z') を 'APZ' を返します。
- **altova:pad-string-right**('AP', 4, 'Z') を 'APZZ' を返します。
- **altova:pad-string-right**('AP', -3, 'Z') を 'AP' を返します。
- **altova:pad-string-right**('AP', 3, 'YZ') は[パド文字が長すぎます] エラーを返します。

repeat-string [altova:]

altova:repeat-string(InputString *as xs:string*, **Repeats** *as xs:integer*) を**xs:string** とする

XP2 XQ1 XP3.1 XQ3.1
最初の InputString 引数により構成される文字列、Repeats 回繰り返してを生成します。

サンプル

- **altova:repeat-string**("Altova #", 3) は "Altova #Altova #Altova #" を返します。

substring-after-last [altova:]

altova:substring-after-last(MainString *as xs:string*, **CheckString** *as xs:string*) を **xs:string** とする XP3.1 XQ3.1
CheckString が MainString 内で検出された場合、MainString 内の CheckString が発生した後のサブ文字列が返されます。MainString 内で CheckString が検出されない場合、空の文字列が返されます。CheckString が空の文字列の場合、MainString 全体が返されます。一度以上発生する場合、CheckString の最後の発生の後の サブ文字列が返されます。

サンプル

- **altova:substring-after-last**('ABCDEFGH', 'B') は 'CDEFGH' を返します。
- **altova:substring-after-last**('ABCDEFGH', 'BC') は 'DEFGH' を返します。
- **altova:substring-after-last**('ABCDEFGH', 'BD') は '' を返します。
- **altova:substring-after-last**('ABCDEFGH', 'Z') は '' を返します。
- **altova:substring-after-last**('ABCDEFGH', '') は 'ABCDEFGH' を返します。
- **altova:substring-after-last**('ABCD-ABCD', 'B') は 'CD' を返します。
- **altova:substring-after-last**('ABCD-ABCD-ABCD', 'BCD') は '' を返します。

substring-before-last [altova:]

altova:substring-before-last(MainString *as xs:string*, **CheckString** *as xs:string*) を **xs:string** とする XP3.1 XQ3.1
CheckString が MainString 内で検出された場合、MainString 内の CheckString が発生する前のサブ文字列が返されます。MainString 内で CheckString が一度以上発生する場合、CheckString の最後の発生の前の サブ文字列が返されます。

サンプル

- **altova:substring-before-last**('ABCDEFGH', 'B') は 'A' を返します。
- **altova:substring-before-last**('ABCDEFGH', 'BC') は 'A' を返します。
- **altova:substring-before-last**('ABCDEFGH', 'BD') は '' を返します。
- **altova:substring-before-last**('ABCDEFGH', 'Z') は '' を返します。
- **altova:substring-before-last**('ABCDEFGH', '') は '' を返します。
- **altova:substring-before-last**('ABCD-ABCD', 'B') は 'ABCD-A' を返します。
- **altova:substring-before-last**('ABCD-ABCD-ABCD', 'ABCD') は 'ABCD-ABCD-' を返します。

substring-pos [altova:]

altova:substring-pos(StringToCheck *as xs:string*, **StringToFind** *as xs:string*) を **xs:integer** とする XP3.1 XQ3.1
StringToCheck 内での StringToFind の最初の発生の文字位置を整数として返します。StringToCheck の最初の文字は、位置 1 にあります。StringToFind が StringToCheck 内で発生しない場合、整数 0 が返されます。第 2 またはその後の StringToCheck, 発生を確認するには、この関数の次の署名を確認してください。

サンプル

- **altova:substring-pos**('Altova', 'to') は 3 を返します。

- **altova:substring-pos**('Altova', 'tov') は 3 を返します。
- **altova:substring-pos**('Altova', 'tv') は 0 を返します。
- **altova:substring-pos**('AltovaAltova', 'to') は 3 を返します。

altova:substring-pos(StringToCheck *as xs:string*, **StringToFind** *as xs:string*, **Integer** *as xs:integer*) を **xs:integer** とする **XP3.1** **XQ3.1**
StringToCheck 内での StringToFind の最初の発生の文字位置を返します。Integer 引数により与えられた、文字位置から StringToFind の検索が開始されます。この位置の前の文字サブ文字列は検索されません。しかし、返された整数は、全体文字列 StringToCheck の検索された文字列の位置です。この署名は、StringToCheck 内で複数回発生する、第 2 または後の発生を検索する際に役に立ちます。 StringToFind が StringToCheck 内で発生しない場合、整数 0 が返されます。

サンプル

- **altova:substring-pos**('Altova', 'to', 1) は 3 を返します。
- **altova:substring-pos**('Altova', 'to', 3) は 3 を返します。
- **altova:substring-pos**('Altova', 'to', 4) は 0 を返します。
- **altova:substring-pos**('Altova-Altova', 'to', 0) は 3 を返します。
- **altova:substring-pos**('Altova-Altova', 'to', 4) は 10 を返します。

trim-string [altova:]

altova:trim-string(InputString *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
この関数は xs:string 引数を必要とし、先頭および後続の空白を削除し、トリミングされた xs:string を返します。

サンプル

- **altova:trim-string**(" Hello World ")) は "Hello World" を返します。
- **altova:trim-string**("Hello World ")) は "Hello World" を返します。
- **altova:trim-string**(" Hello World")) は "Hello World" を返します。
- **altova:trim-string**("Hello World")) は "Hello World" を返します。
- **altova:trim-string**("Hello World")) は "Hello World" を返します。

trim-string-left [altova:]

altova:trim-string-left(InputString *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
この関数は xs:string 引数を必要とし、先頭および後続の空白を削除し、トリミングされた xs:string を返します。

サンプル

- **altova:trim-string-left**(" Hello World ")) は "Hello World " を返します。
- **altova:trim-string-left**("Hello World ")) は "Hello World " を返します。
- **altova:trim-string-left**(" Hello World")) は "Hello World" を返します。
- **altova:trim-string-left**("Hello World")) は "Hello World" を返します。
- **altova:trim-string-left**("Hello World")) は "Hello World" を返します。

trim-string-right [altova:]

altova:trim-string-right(InputString *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
この関数は xs:string 引数を必要とし、先頭および後続の空白を削除し、トリミングされた xs:string を返します。

サンプル

(C) 2015-2021 Altova GmbH

- **altova:trim-string-right**(" Hello World ")は" Hello World" を返します。
- **altova:trim-string-right**("Hello World ")は"Hello World" を返します。
- **altova:trim-string-right**(" Hello World")) は" Hello World" を返します。
- **altova:trim-string-right**("Hello World")) は"Hello World" を返します。
- **altova:trim-string-right**("Hello World")) は"Hello World" を返します。

30.2.1.9 XPath/XQuery 関数: その他

XPath/XQuery 拡張関数の以下の一般的な目的は現在のXMLSpy バージョンでサポートされており(i) XSLT コンテキスト内の XPath 式 または(ii) Xquery ドキュメント内のXQuery 式内で使用することができます。。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova 拡張関数名前空間、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3.1**

▼ decode-string [altova:]

altova:decode-string(Input *as xs:base64Binary*) **as xs:string** **XP3.1** **XQ3.1**
altova:decode-string(Input *as xs:base64Binary*, **Encoding** *as xs:string*) **as xs:string** **XP3.1** **XQ3.1**
指定されたエンコードを使用して送信されたbase64Binary 入力を文字列にデコードします。エンコードが指定されていない場合 UTF-8 エンコードが使用されます。以下のエンコードがサポートされます: US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4

▣ *サンプル*

- **altova:decode-string**($XML1/MailData/Meta/b64B) はbase64Binary 入力をUTF-8 エンコード済み文字列として返します。
- **altova:decode-string**($XML1/MailData/Meta/b64B, "UTF-8") はbase64Binary 入力をUTF-8-エンコード済み文字列として返します。
- **altova:decode-string**($XML1/MailData/Meta/b64B, "ISO-8859-1") はbase64Binary 入力をISO-8859-1 エンコード済み文字列として返します。

▼ encode-string [altova:]

altova:encode-string(InputString *as xs:string*) **as xs:base64Binaryinteger** **XP3.1** **XQ3.1**
altova:encode-string(InputString *as xs:string*, **Encoding** *as xs:string*) **as**

xs:base64Binaryinteger XP3.1 XQ3.1
与えらえている場合指定されているエンコードを使用して送信された文字列をエンコードします。エンコードが与えられていない場合 UTF-8 エンコードが使用されます。エンコードされた文字列はbase64Binary 文字に変換され、エンコードされたbase64Binary 値が返されます。最初にUTF-8 エンコードがサポートされ、サポートは以下のエンコードに拡張されます: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

サンプル

- **altova:encode-string**("Altova") はUTF-8 エンコード済み文字列 "Altova" の等価である base64Binary を返します。
- **altova:encode-string**("Altova", "UTF-8") はUTF-8 エンコード済み文字列 "Altova" の等価であるbase64Binary のを返します。

get-temp-folder [altova:]

altova:get-temp-folder() を xs:string とする XP2 XQ1 XP3.1 XQ3.1
この関数は引数を必要としません。この関数は現在のユーザーの一時的なフォルダーへのパスを返します。

サンプル

- **altova:get-temp-folder**() は、マシン上で、xs:string として C:\Users\<UserName>\AppData\Local\Temp\ と類似したパスを返します。

generate-guid [altova:]

altova:generate-guid() asxs:string XP2 XQ1 XP3.1 XQ3.1
ユニークな文字列 GUID 文字列を生成します。

サンプル

- **altova:generate-guid**() は例えば 85F971DA-17F3-4E4E-994E-99137873ACCD を返します。

high-res-timer [altova:]

altova:high-res-timer() asxs:double XP3.1 XQ3.1
秒数でシステムハイレゾリューションタイマーの値を返します。ハイレゾリューションタイマーがシステム内で存在すると、必要とされる場合、高度に正確な時間の計算を有効化します（例えば、アニメーションと正確なコード実行の時間の決定などが例として挙げられます）。この関数は、システムのタイマーに精度を与えます。

サンプル

- **altova:high-res-timer**() は '1.16766146154566E6' などを返します。

parse-html [altova:]

altova:parse-html(HTMLText *as xs:string*) **asnode()** XP3.1 XQ3.1
HTMLText 引数は、HTML ドキュメントのテキストを含む文字列です。関数は、文字列からHTML ツリーを作成します。提供された文字列は、HTML 要素を含む、または、含まない場合があります。いずれの場合でも、ツリーのルート要素は、**HTML** と名付けられます。提出された文字列内のHTML コードが有効な HTMLであることを確認することが奨励されます。.

サンプル

- **altova:parse-html**("<html><head/><body><h1>Header</h1></body></html>") は提供された文字列からHTML ツリーを作成します。

▼ sleep[altova:]

altova:sleep(Millisecs *as xs:integer*) **asempty-sequence()** XP2 XQ1 XP3.1 XQ3.1

Millisecs により与えられるミリ秒で示される期間のための実行を延期します。

サンプル

- **altova:sleep**(1000) は1000 ミリ秒のための実行を延期します。

[[トップ] 2051]

30.2.1.10 チャート関数

以下に示されるチャート関数により、チャートをイメージとして作成、生成、保存することができます。これらの機能は、お使いの Altova 製品で、以下に記される方法によりサポートされます。しかしながら、将来使用されるバージョンの製品において、以下にある関数のサポートが打ち切られたり、個々の振る舞いが変化する可能性があることに留意してください。将来のリリースにおける Altova 拡張関数のサポートについては、そのバージョンのドキュメンテーションを参照ください。

(XSLT 関数ではない) チャートに関する XPath 関数は、2 つのグループに分けられています::

- チャートの生成と保存を行う関数 2053
- チャートの作成を行う関数 2054

メモ: チャート関数は、**Enterprise** ならびに**サーバー エディション**の Altova 製品でしかサポートされていないことにご注意してください。

メモ: サーバーエディションのチャートでサポートされるイメージのフォーマットは jpg、png、および bmp です。無損失圧縮のため、推奨されるオプションは png です。Enterprise エディションでサポートされるフォーマットは jpg. png、bmp、および gif です。

チャートの生成と保存を行う関数

これらの関数は、(チャート作成関数により得られた) チャートオブジェクトを受け取り、イメージの生成を行うか、イメージをファイルへ保存します。

altova:generate-chart-image ($chart, $width, $height, $encoding) as atomic

以下の説明を参照ください

- $chart は altova:create-chart 関数により得られたチャート拡張アイテムです。
- $width ならびに $height により大きさを指定する必要があります。
- $encoding は x-binarytobase64 または x-binarytobase16 であることができます。

関数からは、指定されたエンコーディングならびにイメージフォーマットにてチャートのイメージが返されます。

altova:generate-chart-image ($chart, $width, $height, $encoding, $imagetype) as atomic

以下の説明を参照ください

- $chart は altova:create-chart 関数により得られたチャート拡張アイテムです。

- $width ならびに$height により大きさを指定する必要があります。
- $encoding はx-binarytobase64 またはx-binarytobase16 であることができます。
- $imagetype は次のフォーマットから選択することができます: png、gif、bmp、jpg、jpeg。gif はサーバー製品でサポートされていないことにご注意してください。*ページの上のメモを参照してください*。

関数からは、指定されたエンコーディングならびにイメージフォーマットにてチャートのイメージが返されます。

altova:save-chart-image ($chart, $filename, $width, $height) as empty() *(Windows のみ)*

以下の説明を参照ください

- $chart はaltova:create-chart 関数により得られたチャート拡張アイテムです。
- $filename は保存されるチャートイメージのファイルパスならびにファイル名です。
- $width ならびに$height により大きさを指定する必要があります。

関数により$filename で指定されたファイルにチャートイメージが保存されます。

altova:save-chart-image ($chart, $filename, $width, $height, $imagetype) as empty() *(Windows のみ)*

以下の説明を参照ください

- $chart はaltova:create-chart 関数により得られたチャート拡張アイテムです。
- $filename は保存されるチャートイメージのファイルパスならびにファイル名です。
- $width ならびに$height により大きさを指定する必要があります。
- $imagetype は次のフォーマットから選択することができます: png、gif、bmp、jpg、jpeg。gif はサーバー製品でサポートされていないことにご注意してください。*ページの上のメモを参照してください*。

指定されたイメージフォーマットで$filename により指定されたファイルにチャートイメージが保存されます。

チャートの作成を行う関数

以下の関数によりチャートの作成を行うことができます

altova:create-chart($chart-config, $chart-data-series*) as chart extension item

以下の説明を参照ください

- $chart-config はaltova:create-chart-config 関数またはaltova:create-chart-config-from-xml 関数により取得されたchart-config 拡張アイテムとなります。
- $chart-data-series はaltova:create-chart-data-series 関数またはaltova:create-chart-data-series-from-rows 関数により取得されたchart-data-series 拡張アイテムとなります。

関数により、引数により与えられたデータから作成されたチャート拡張アイテムが返されます。

altova:chart($chart-config, $chart-data-series*) as chart extension item

以下の説明を参照ください

- $chart-config はチャート構成拡張アイテムです。順序の決まっていない4つのキーのシリーズです: 4つのキーが"width"、"height"、"title"、および"kind" である値ペアです。width とheight の値は整数で、チャートの幅と高さを

ピクセルで指定します。kind の値は以下のとおりです: Pie、 Pie3d、 BarChart、 BarChart3d、 BarChart3dGrouped、 LineChart、 ValueLineChart、 RoundGauge、 BarGauge。

- $chart-data-series は各配列がチャートデータ系列を定義するサイズ 3 の配列です。各配列は以下により構成されています: (i) データ系列の名前 (ii) X-軸の値 (iii) Y-軸の値。複数のデータ系列は提供される場合があります。このサンプルでは、例えば、2 つのデータ配列はそれぞれ月々の最高および最低気温を提供します。

巻数は引数を介して提供されるデータから作成されたチャート拡張アイテムを返します。

サンプル:
```
altova:chart( map{'width':800, 'height':600, "kind":"LineChart", "title":"Monthly Temperatures"}, (['Min', $temps/Month, $temps/Month/@min], ['Max', $temps/Month, $temps/Month/@max]) )
```

altova:create-chart-config($type-name, $title) as chart-config extension item

以下の説明を参照ください

- $type-name により、以下から作成される種類のチャートが選択されます: Pie、Pie3d、 BarChart、BarChart3d、BarChart3dGrouped、 LineChart、ValueLineChart、RoundGauge、BarGauge
- $title によりチャートの名前が指定されます。

関数により、チャートの構成情報が含まれる chart-config 拡張アイテムが返されます。

altova:create-chart-config-from-xml($xml-struct) as chart-config extension item

以下の説明を参照ください

- $xml-struct はチャートの構成情報が含まれる XML 構造です。

関数により、チャートの構成情報が含まれる chart-config 拡張アイテムが返されます。この情報は XML データフラグメント 2057 から与えられます。

altova:create-chart-data-series($series-name?, $x-values*, $y-values*) as chart-data-series extension item

以下の説明を参照ください

- $series-name により、系列の名前が指定されます。
- $x-values により、X-軸値のリストが与えられます。
- $y-values により、Y-軸値のリストが与えられます。

関数により、チャートの作成に必要なデータの情報(系列の名前と軸のデータ)が含まれる chart-data-series 拡張アイテムが返されます。

altova:create-chart-data-row(x, y1, y2, y3, ...) as chart-data-x-*N*y-row extension item

以下の説明を参照ください

- x はチャートデータ行の X-軸カラムにおける値です。
- yN は Y-軸カラムの値です。

関数により、1 つの系列の X-軸カラムならびに Y-軸カラムに対するデータが含まれる chart-data-x-*N*y-row 拡張アイテムが返されます。

altova:create-chart-data-series-from-rows($series-names as xs:string*, $row*) as chart-data-series extension item

以下の説明を参照ください

- $series-name は作成される系列の名前です。
- $row は系列として作成される chart-data-x-*N*y-row 拡張アイテムです。

関数により、系列のX-軸ならびにY-軸に関するデータが含まれる chart-data-series 拡張アイテムが返されます。

altova:create-chart-layer($chart-config, $chart-data-series*) as chart-layer extension item

以下の説明を参照ください

- $chart-config は altova:create-chart-config 関数または altova:create-chart-config-from-xml 関数により取得された $chart-config 拡張アイテムです。
- $chart-data-series は altova:create-chart-data-series 関数または altova:create-chart-data-series-from-rows 関数により取得された chart-data-series 拡張アイテムです。

関数により、チャートのレイヤーデータが含まれる **chart-layer** 拡張アイテムが返されます。

altova:create-multi-layer-chart($chart-config, $chart-data-series*, $chart-layer*)

以下の説明を参照ください

- $chart-config は altova:create-chart-config 関数または altova:create-chart-config-from-xml 関数により取得された chart-config 拡張アイテムです。
- $chart-data-series は altova:create-chart-data-series 関数または altova:create-chart-data-series-from-rows 関数により取得された chart-data-series 拡張アイテムです。
- $chart-layer は altova:create-chart-layer 関数により取得された **chart-layer** 拡張アイテムです。

関数により、複数レイヤーのチャートアイテムが返されます。

altova:create-multi-layer-chart($chart-config, $chart-data-series*, $chart-layer*, xs:boolean $mergecategoryvalues)

以下の説明を参照ください

- $chart-config は altova:create-chart-config 関数または altova:create-chart-config-from-xml 関数により取得された chart-config 拡張アイテムです。
- $chart-data-series は altova:create-chart-data-series 関数または altova:create-chart-data-series-from-rows 関数により取得された chart-data-series 拡張アイテムです。
- $chart-layer は altova:create-chart-layer 関数により取得された **chart-layer** 拡張アイテムです。
- $mergecategoryvalues は true の場合、複数のデータシリーズをマージします。false の場合はマージしません。

関数により、複数レイヤーのチャートアイテムが返されます。

(C) 2015-2021 Altova GmbH

30.2.1.10.1 チャートデータの XML 構造

以下にチャートデータの XML 構造、ならびにそれが Altova のチャート拡張関数 2053 においてどのように表示されるかを示します。チャートの種類によっては、ここでの記述内容により外観が変化します。全ての要素が全種類のチャートに対して使用されるわけではなく、例えば <Pie> 要素は棒グラフに対して無視されます。

メモ チャート関数は、**Enterprise** ならびに**サーバーエディション**の Altova 製品でしかサポートされていないことにご注意してください。

```
<chart-config>
    <General
    SettingsVersion="1" must be provided
    ChartKind="BarChart" Pie, Pie3d, BarChart, StackedBarChart, BarChart3d,
BarChart3dGrouped, LineChart, ValueLineChart, AreaChart, StackedAreaChart, RoundGauge, BarGauge,
CandleStick
    BKColor="#ffffff" Color
    BKColorGradientEnd="#ffffff" Color. In case of a gradient, BKColor and BKColorGradientEnd define
the gradient's colors
    BKMode="#ffffff" Solid, HorzGradient, VertGradient
    BKFile="Path+Filename" String. If file exists, its content is drawn over the background.
    BKFileMode="Stretch" Stretch, ZoomToFit, Center, Tile
    ShowBorder="1" Bool
    PlotBorderColor="#000000" Color
    PlotBKColor="#ffffff" Color
    Title="" String
    ShowLegend="1" Bool
    OutsideMargin="3.%" PercentOrPixel
    TitleToPlotMargin="3.%" PercentOrPixel
    LegendToPlotMargin="3.%" PercentOrPixel
    Orientation="vert" Enumeration: possible values are: vert, horz
    >
    <TitleFont
        Color="#000000" Color
        Name="Tahoma" String
        Bold="1" Bool
        Italic="0" Bool
        Underline="0" Bool
        MinFontHeight="10.pt" FontSize (only pt values)
        Size="8.%" FontSize />
    <LegendFont
        Color="#000000"
        Name="Tahoma"
        Bold="0"
        Italic="0"
        Underline="0"
        MinFontHeight="10.pt"
        Size="3.5%" />
    <AxisLabelFont
        Color="#000000"
        Name="Tahoma"
        Bold="1"
        Italic="0"
        Underline="0"
```

```
      MinFontHeight="10.pt"
      Size="5.%" />
</General>

<Line
      ConnectionShapeSize="1.%"   PercentOrPixel
      DrawFilledConnectionShapes="1"   Bool
      DrawOutlineConnectionShapes="0"   Bool
      DrawSlashConnectionShapes="0"   Bool
      DrawBackslashConnectionShapes="0"   Bool
/>

<Bar
      ShowShadow="1"   Bool
      ShadowColor="#a0a0a0"   Color
      OutlineColor="#000000"   Color
      ShowOutline="1"   Bool
/>

<Area
      Transparency="0"   UINT ( 0-255 ) 255 is fully transparent, 0 is opaque
      OutlineColor="#000000"   Color
      ShowOutline="1"   Bool
/>

<CandleStick
      FillHighClose="0"   Bool. If 0, the body is left empty. If 1, FillColorHighClose is used for
the candle body
      FillColorHighClose="#ffffff"   Color. For the candle body when close > open
      FillHighOpenWithSeriesColor="1"   Bool. If true, the series color is used to fill the
candlebody when open > close
      FillColorHighOpen="#000000"   Color. For the candle body when open > close and
FillHighOpenWithSeriesColor is false
/>

<Colors   User-defined color scheme: By default this element is empty except for the style and
has no Color attributes
      UseSubsequentColors ="1"   Boolean. If 0, then color in overlay is used. If 1, then
subsequent colors from previous chart layer is used
      Style="User"   Possible values are: "Default", "Grayscale", "Colorful", "Pastel", "User"
      Colors="#52aca0"   Color: only added for user defined color set
      Colors1="#d3c15d"   Color: only added for user defined color set
      Colors2="#8971d8"   Color: only added for user defined color set
      ...
      ColorsN=""   Up to ten colors are allowed in a set: from Colors to Colors9
</Colors>

<Pie
      ShowLabels="1"   Bool
      OutlineColor="#404040"   Color
      ShowOutline="1"   Bool
      StartAngle="0."   Double
      Clockwise="1"   Bool
      Draw2dHighlights="1"   Bool
      Transparency="0"   Int (0 to 255: 0 is opaque, 255 is fully transparent)
```

```
            DropShadowColor="#c0c0c0"   Color
            DropShadowSize="5.%"   PercentOrPixel
            PieHeight="10.%"   PercentOrPixel. Pixel values might be different in the result because
of 3d tilting
            Tilt="40.0"   Double (10 to 90: The 3d tilt in degrees of a 3d pie)
            ShowDropShadow="1"   Bool
            ChartToLabelMargin="10.%"   PercentOrPixel
            AddValueToLabel="0"   Bool
            AddPercentToLabel="0"   Bool
            AddPercentToLabels_DecimalDigits="0"  UINT ( 0 – 2 )
            >
            <LabelFont
                  Color="#000000"
                  Name="Arial"
                  Bold="0"
                  Italic="0"
                  Underline="0"
                  MinFontHeight="10.pt"
                  Size="4.%"    />
      </Pie>

      <XY>
            <XAxis   Axis
                  AutoRange="1"   Bool
                  AutoRangeIncludesZero="1"   Bool
                  RangeFrom="0."   Double: manual range
                  RangeTill="1."   Double : manual range
                  LabelToAxisMargin="3.%"   PercentOrPixel
                  AxisLabel=""   String
                  AxisColor="#000000"   Color
                  AxisGridColor="#e6e6e6"   Color
                  ShowGrid="1"   Bool
                  UseAutoTick="1"   Bool
                  ManualTickInterval="1."   Double
                  AxisToChartMargin="0.px"   PercentOrPixel
                  TickSize="3.px"   PercentOrPixel
                  ShowTicks="1"   Bool
                  ShowValues="1"   Bool
                  AxisPosition="LeftOrBottom"   Enums: "LeftOrBottom", "RightOrTop", "AtValue"
                  AxisPositionAtValue = "0"   Double
                  >
                  <ValueFont
                        Color="#000000"
                        Name="Tahoma"
                        Bold="0"
                        Italic="0"
                        Underline="0"
                        MinFontHeight="10.pt"
                        Size="3.%"    />
            </XAxis>
            <YAxis   Axis (same as for XAxis)
                  AutoRange="1"
                  AutoRangeIncludesZero="1"
                  RangeFrom="0."
                  RangeTill="1."
                  LabelToAxisMargin="3.%"
                  AxisLabel=""
```

```
                AxisColor="#000000"
                AxisGridColor="#e6e6e6"
                ShowGrid="1"
                UseAutoTick="1"
                ManualTickInterval="1."
                AxisToChartMargin="0.px"
                TickSize="3.px"
                ShowTicks="1"   Bool
                ShowValues="1"   Bool
                AxisPosition="LeftOrBottom"   Enums: "LeftOrBottom", "RightOrTop", "AtValue"
                AxisPositionAtValue = "0"   Double
                >
                <ValueFont
                      Color="#000000"
                      Name="Tahoma"
                      Bold="0"
                      Italic="0"
                      Underline="0"
                      MinFontHeight="10.pt"
                      Size="3.%"/>
          </YAxis>
    </XY>

    <XY3d
          AxisAutoSize="1"   Bool: If false, XSize and YSize define the aspect ration of x and y
axis. If true, aspect ratio is equal to chart window
          XSize="100.%"   PercentOrPixel. Pixel values might be different in the result because of
3d tilting and zooming to fit chart
          YSize="100.%"   PercentOrPixel. Pixel values might be different in the result because of
3d tilting and zooming to fit chart
          SeriesMargin="30.%"   PercentOrPixel. Pixel values might be different in the result
because of 3d tilting and zooming to fit chart
          Tilt="20."   Double. -90 to +90 degrees
          Rot="20."   Double. -359 to +359 degrees
          FoV="50.">   Double. Field of view: 1-120 degree
          >
          <ZAxis
                AutoRange="1"
                AutoRangeIncludesZero="1"
                RangeFrom="0."
                RangeTill="1."
                LabelToAxisMargin="3.%"
                AxisLabel=""
                AxisColor="#000000"
                AxisGridColor="#e6e6e6"
                ShowGrid="1"
                UseAutoTick="1"
                ManualTickInterval="1."
                AxisToChartMargin="0.px"
                TickSize="3.px" >
                <ValueFont
                      Color="#000000"
                      Name="Tahoma"
                      Bold="0"
                      Italic="0"
                      Underline="0"
                      MinFontHeight="10.pt"
                      Size="3.%"/>
          </ZAxis>
```

```
        </XY3d>

        <Gauge
                MinVal="0."   Double
                MaxVal="100."   Double
                MinAngle="225"   UINT: -359-359
                SweepAngle="270"   UINT: 1-359
                BorderToTick="1.%"   PercentOrPixel
                MajorTickWidth="3.px"   PercentOrPixel
                MajorTickLength="4.%"   PercentOrPixel
                MinorTickWidth="1.px"   PercentOrPixel
                MinorTickLength="3.%"   PercentOrPixel
                BorderColor="#a0a0a0"   Color
                FillColor="#303535"   Color
                MajorTickColor="#a0c0b0"   Color
                MinorTickColor="#a0c0b0"   Color
                BorderWidth="2.%"   PercentOrPixel
                NeedleBaseWidth="1.5%"   PercentOrPixel
                NeedleBaseRadius="5.%"   PercentOrPixel
                NeedleColor="#f00000"   Color
                NeedleBaseColor="#141414"   Color
                TickToTickValueMargin="5.%"   PercentOrPixel
                MajorTickStep="10."   Double
                MinorTickStep="5."   Double
                RoundGaugeBorderToColorRange="0.%"   PercentOrPixel
                RoundGaugeColorRangeWidth ="6.%"   PercentOrPixel
                BarGaugeRadius="5.%"   PercentOrPixel
                BarGaugeMaxHeight="20.%"   PercentOrPixel
                RoundGaugeNeedleLength="45.%"   PercentOrPixel
                BarGaugeNeedleLength="3.%"   PercentOrPixel
                >
                <TicksFont
                        Color="#a0c0b0"
                        Name="Tahoma"
                        Bold="0"
                        Italic="0"
                        Underline="0"
                        MinFontHeight="10.pt"
                        Size="4.%"
                />
                <ColorRanges>   User-defined color ranges. By default empty with no child element entries
                        <Entry
                                From="50. "   Double
                                FillWithColor="1"   Bool
                                Color="#00ff00"   Color
                        />
                        <Entry
                                From="50.0"
                                FillWithColor="1"
                                Color="#ff0000"
                        />
                        ...
                </ColorRanges>
        </Gauge>
</chart-config>
```

30.2.1.10.2 チャート関数

以下のサンプルXSLTドキュメントでは、Altova のチャート拡張関数[2053]の使用方法が示されます。更に下には、XMLドキュメントと、Altova XSLT 2.0 または3.0 エンジンとXSLTドキュメントによりXMLドキュメントが処理された結果生成される出力イメージのスクリーンショットが示されます。

メモ: チャート関数は、**Enterprise** ならびに**サーバー エディション**のAltova 製品でしかサポートされていないことにご注意してください。

メモ: チャートデータテーブルの作成に関する詳しい情報については、Altova XMLSpy ならびにStyleVision 製品のドキュメンテーションを参照ください。

XSLT ドキュメント

以下にあるXSLTドキュメントでは、Altova チャート拡張関数を使用することで円グラフを生成します。更に下に記されているXMLドキュメントを処理するために使用することができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:altovaext="http://www.altova.com/xslt-extensions"
    exclude-result-prefixes="#all">
    <xsl:output version="4.0" method="html" indent="yes" encoding="UTF-8"/>
    <xsl:template match="/">
        <html>
            <head>
                <title>
                    <xsl:text>HTML Page with Embedded Chart</xsl:text>
                </title>
            </head>
            <body>
                <xsl:for-each select="/Data/Region[1]">
                    <xsl:variable name="extChartConfig" as="item()*">
                        <xsl:variable name="ext-chart-settings" as="item()*">
                            <chart-config>
                                <General
                                    SettingsVersion="1"
                                    ChartKind="Pie3d"
                                    BKColor="#ffffff"
                                    ShowBorder="1"
                                    PlotBorderColor="#000000"
                                    PlotBKColor="#ffffff"
                                    Title="{@id}"
                                    ShowLegend="1"
                                    OutsideMargin="3.2%"
                                    TitleToPlotMargin="3.%"
                                    LegendToPlotMargin="6.%"
                                    >
                                    <TitleFont
                                        Color="#023d7d"
                                        Name="Tahoma"
                                        Bold="1"
                                        Italic="0"
                                        Underline="0"
                                        MinFontHeight="10.pt"
                                        Size="8.%" />
```

```
                                    </General>
                                </chart-config>
                            </xsl:variable>
                            <xsl:sequence select="altovaext:create-chart-config-from-xml( $ext-chart-settings )"/>
                        </xsl:variable>
                        <xsl:variable name="chartDataSeries" as="item()*">
                            <xsl:variable name="chartDataRows" as="item()*">
                                <xsl:for-each select="(Year)">
                                    <xsl:sequence select="altovaext:create-chart-data-row( (@id), ( .) )"/>
                                </xsl:for-each>
                            </xsl:variable>
                            <xsl:variable name="chartDataSeriesNames" as="xs:string*" select=" ( (&quot;Series 1&quot;),
&apos;&apos; )[1]"/>
                            <xsl:sequence
                                select="altovaext:create-chart-data-series-from-rows( $chartDataSeriesNames,
$chartDataRows)"/>
                        </xsl:variable>
                        <xsl:variable name="ChartObj" select="altovaext:create-chart( $extChartConfig, ( $chartDataSeries),
false() )"/>
                        <xsl:variable name="sChartFileName" select="'mychart1.png'"/>
                        <img src="{$sChartFileName, altovaext:save-chart-image( $ChartObj, $sChartFileName, 400,
400 ) }"/>
                    </xsl:for-each>
                </body>
            </html>
        </xsl:template>
</xsl:stylesheet>
```

XML ドキュメント

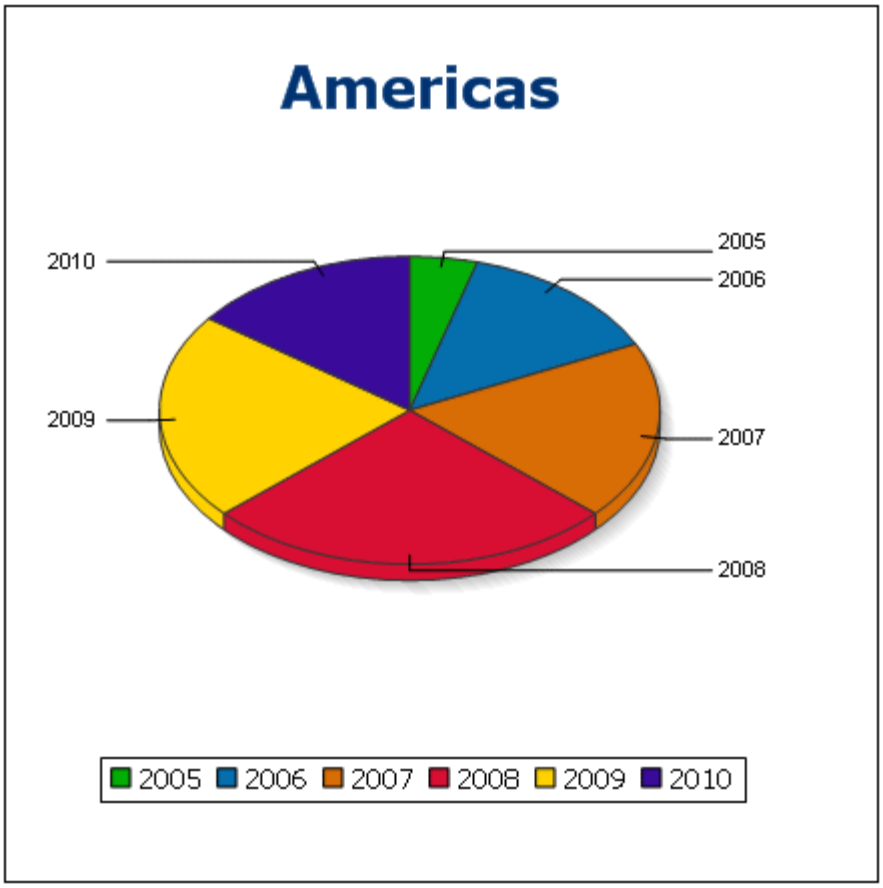
上にある XSLT ドキュメントにより以下の XML ドキュメントを処理することができます。XML ドキュメント内にあるデータを使用することで、下に示される円グラフを生成することができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="YearlySales.xsd">
    <ChartType>Pie Chart 2D</ChartType>
    <Region id="Americas">
        <Year id="2005">30000</Year>
        <Year id="2006">90000</Year>
        <Year id="2007">120000</Year>
        <Year id="2008">180000</Year>
        <Year id="2009">140000</Year>
        <Year id="2010">100000</Year>
    </Region>
    <Region id="Europe">
        <Year id="2005">50000</Year>
        <Year id="2006">60000</Year>
        <Year id="2007">80000</Year>
        <Year id="2008">100000</Year>
        <Year id="2009">95000</Year>
        <Year id="2010">80000</Year>
    </Region>
    <Region id="Asia">
        <Year id="2005">10000</Year>
        <Year id="2006">25000</Year>
        <Year id="2007">70000</Year>
        <Year id="2008">110000</Year>
        <Year id="2009">125000</Year>
        <Year id="2010">150000</Year>
    </Region>
```

</Data>

出力イメージ

上にある XSLT ドキュメントを使って XML ドキュメントを処理することで、以下にある円グラフが生成されます



30.2.2 その他の拡張関数

Java や C# などのプログラミング言語には、XPath 2.0 / XQuery 関数、または XSLT 2.0 関数として利用できない関数がいくつかあります。そのような関数の良い例として、Java で利用することのできる `sin()` や `cos()` といった数学関数があります。XSLT スタイルシートや XQuery のクエリにてこれらの関数が利用できるのであれば、スタイルシートやクエリの適用範囲を大幅に拡張することができ、スタイルシート作成タスクの負担が大幅に軽減されます。Altova 製品で使用されている Altova エンジン(XSLT 1.0、XSLT 2.0、XQuery 1.0)では、Java[2065] や .NET[2073] および MSXSL scripts for XSLT[2078] における拡張関数の使用がサポートされます。 このセクションでは、拡張機能および XSLT スタイルシート内で MSXSL スクリプト、および XQuery ドキュメントを使用する方法について記述します。使用できる拡張関数は以下のように構成されます:

- Java 拡張関数[2065]
- .NET 拡張関数[2073]
- XSLT に対する MSXSL スクリプト[2078]

記述の中では特に、(i) 関連するライブラリ内の関数がどのように呼ばれるか、(ii) 関数呼び出しを行う際に入力として使用される引数を変換するのにどのようなルールが適用され、return により値が返される際にどのような変換ルールが適用されるのか(XSLT/XQuery データオブジェクトに対する関数の結果)について説明されます。

必要条件

拡張関数のサポートを有効にするには、XSLT 変換や XQuery の実行を行うコンピューターに Java Runtime Environment (Java 関数にアクセスする場合)ならびに .NET Framework 2.0 以上(.NET 関数にアクセスする場合)がインストールされている、またはアクセスできる環境が整っている必要があります。

30.2.2.1 Java 拡張関数

Java 拡張関数は、XPath または XQuery 条件式にて使用することができるほか、Java のコンストラクターを呼び出したり、Java の(静的またはインスタンス)メソッドを呼び出すことができます。

Java クラスのフィールドは、引数を持たないメソッドとして扱われます。フィールドは静的またはインスタンスとして存在することができます。フィールドへのアクセス方法については、静的とインスタンスの両方について、以下のサブセクションにて記述されます。

このセクションは以下のサブセクションにより構成されます:

- Java : コンストラクター 2070
- Java : 静的メソッドと静的フィールド 2070
- Java : インスタンスメソッドとインスタンスフィールド 2071
- データ型 : XPath/XQuery から Java へ 2072
- データ型 : Java から XPath/XQuery へ 2073

以下の点に注意してください:

- Altova デスクトップ製品を使用している場合、Altova アプリケーションは、Java 仮想マシンへのパスを(以下の順序で)読み取ることにより自動的に検知しようとします: (i) Windows レジストリ (ii) **JAVA_HOME** 環境変数。アプリケーションのオプションダイアログ内にカスタムパスを追加することもできます。このエントリは自動的に検知された他の Java VM パス以上の優先順位を有します。
- Altova サーバー製品が Windows マシン上で作動している場合、Java 仮想マシンへのパスは Windows レジストリから最初に読み取られます。成功しない場合、**JAVA_HOME** 環境変数が使用されます。
- Altova サーバー製品を作動する場合、および、Linux または macOS マシン上でサーバー製品を作動する場合、Java 仮想マシンへのパスが **JAVA_HOME** 環境変数内に保管されていることを確認してください。**\bin\server** または **\bin\client** ディレクトリ内の **jvm.dll** ファイルを指している必要があります。

拡張関数のフォーム

XPath/XQuery 条件式における拡張関数では、`prefix:fname()` の形式を取る必要があります。

- `prefix:` 部により拡張関数が Java 関数として認識されます。`java:` から始まる URI のスコープ内の名前空間宣言に拡張関数を関連付けることで Java 関数であるという認識が行われます。名前空間の宣言により、例えば `xmlns:myns="java:java.lang.Math"` という Java クラスが特定されます。名前空間の宣言は、(コロン無しの) `xmlns:myns="java"` という形式で、Java クラスの識別子を拡張関数にある `fname()` 部の左型に配置することでも行うことができます。

- `fname()` 部により、呼び出されている Java メソッドが識別され、メソッドの引数が提供されます（以下の例を参照ください）。`prefix:` 部にて識別された名前空間 URI が Java クラスを識別できなかった場合、Java クラスの識別はクラスの前にくる `fname()` 部にて行うことになり、ピリオドによりクラスから分離されることになります（以下にある2番目の XSLT サンプルを参照）。

メモ 呼び出されるクラスはコンピューターのクラスパス上にある必要があります。

XSLT サンプル

以下に静的メソッドを呼び出す2つのサンプルを示します。最初のサンプルでは、クラス名（`java.lang.Math`）が名前空間 URI に加えられており、`fname()` へ加えることはできません。2 番目のサンプルでは、`prefix:` 部に `java:` が与えられており、`fname()` 部にてクラスとメソッドが識別されます。

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jmath="java"
              select="jmath:java.lang.Math.cos(3.14)" />
```

拡張関数内にあるメソッド名（上の例では `cos()` ）は、名前付き Java クラス（上の例では `java.lang.Math` ）の public な静的メソッドの名前に一致する必要があります。

XQuery サンプル

以下に XSLT のサンプルに似た XQuery のサンプルを示します:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

ユーザー定義された Java クラス

独自の Java クラスやメソッドを作成した場合、(i) JAR ファイル（または class ファイル）を介してこれらクラスファイルへアクセスしているか、(ii) これら（JAR または class）ファイルが、カレントディレクトリ（XSLT や XQuery ドキュメントが存在するディレクトリ）に配置されているかにより、これらクラスの呼び出し方法が変わってきます。これらファイルの特定方法については、ユーザー定義クラスファイル[2066]ならびにユーザー定義 JAR ファイル[2069]を参照ください。カレントディレクトリには無いクラスファイルや JAR ファイルへのパスは指定しなければならないことご注意してください。

30.2.2.1.1 ユーザー定義のクラスファイル

アクセスがクラスファイルを介したものである場合、4 つのケースが考えられます:

- クラスファイルはパッケージである。XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている。（下のサンプルを参照[2067]）
- クラスファイルはパッケージでは無い。XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている。（下のサンプルを参照[2067]）
- クラスファイルはパッケージである。XSLT または XQuery ファイルがランダムな場所に収められている。（下のサンプルを参照[2068]）
- クラスファイルはパッケージである。XSLT または XQuery ファイルがランダムな場所に収められている。（下のサンプルを参照[2067]）

クラスファイルがパッケージではなく、XSLT または XQuery ドキュメントと同じ場所に収められているケースを考えてみましょう。この場合、フォルダー内の全クラスを発見することができるため、ファイルの場所を指定する必要はありません。クラスの識別を行う構文は以下のようになります:

```
java:classname
```

ここで、

`java:` によりユーザー定義の Java 関数が呼ばれていることが示されます(デフォルトでカレントディレクトリにある Java クラスがロードされます)。
`classname` は目的となるメソッドのクラスが含まれているクラスの名前です。

クラスが名前空間 URI にて識別され、名前空間がメソッド呼び出しにて使用されます。

クラスファイルがパッケージで、XSLT/XQuery ファイルが Java パッケージと同じ場所に収められている

以下の例では、`com.altova.extfunc` パッケージにある `Car` クラスの `getVehicleType()` メソッドが呼び出されています。`com.altova.extfunc` パッケージは `JavaProject` という名前のフォルダーに置かれており、XSLT ファイルも同じフォルダーに配置されています。

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java:com.altova.extfunc.Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

クラスファイルが参照され、XSLT/XQuery ファイルがクラスファイルと同じフォルダーに収められている

下のサンプルでは `Car` クラスの `getVehicleType()` メソッドが呼び出されます。以下を述べることができます: (i) `Car` クラスは次のフォルダー内にあります: `JavaProject/com/altova/extfunc` 。(ii) このフォルダーは下のサンプルの現在のフォルダー内に存在します。XSLT ファイルもフォルダー `JavaProject/com/altova/extfunc` 内にあります。

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java:Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

クラスファイルがパッケージされXSLT/XQuery ファイルがランダムな場所に収められている

以下の例では、com.altova.extfunc パッケージにある Car クラスの getVehicleColor() メソッドが呼び出されています。com.altova.extfunc パッケージは JavaProject という名前のフォルダーに置かれており、XSLT ファイルが任意の場所に配置されています。この場合、以下のような構文でパッケージの場所をクエリ文字列として URI 内にて指定する必要があります:

```
java:classname[?path=uri-of-package]
```

ここで、

java: によりユーザー定義の Java 関数が呼ばれていることを表します。
uri-of-package は Java パッケージの URI です。
classname は目的のメソッドが含まれているクラス名です。

クラスは名前空間 URI により特定され、名前空間がメソッド呼び出しのプレフィックスで使用されます。以下の例ではカレントディレクトリ以外にあるクラスファイルへのアクセスを行うことができます。

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java:com.altova.extfunc.Car?path=file:///C:/JavaProject/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

クラスファイルがパッケージではなく、XSLT/XQuery ファイルがランダムな場所に収められている

以下の例では、com.altova.extfunc パッケージにある Car クラスの getCarColor() メソッドが呼び出されています。com.altova.extfunc パッケージは JavaProject という名前のフォルダーに置かれており、XSLT ファイルが任意の場所に配置されています。以下のような構文で、クラスファイルの場所をクエリ文字列として URI 内にて指定する必要があります:

```
java:classname[?path=uri-of-classfile]
```

ここで

java: によりユーザー定義の Java 関数が呼ばれていることを表します。
uri-of-classfile は Java パッケージの URI です。
classname は目的のメソッドが含まれているクラス名です。

クラスは名前空間 URI により特定され、名前空間はメソッド呼び出しのプレフィックスで使用されます。以下の例ではカレントディレクトリ以外にあるクラスファイルへのアクセスを行うことができます。

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java:Car?path=file:///C:/JavaProject/com/altova/extfunc/" >
```


(C) 2015-2021 Altova GmbH

```
<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

メモ パスが外部関数により与えられている場合、ClassLoader により パスが追加されます。

30.2.2.1.2 ユーザー定義の JAR ファイル

JAR ファイル経由でアクセスが行われた場合、以下の構文により JAR ファイルの URI を指定する必要があります:

```
xmlns:classNS="java:classname?path=jar:uri-of-jarfile!/"
```

クラスの識別を行う名前空間 URI のプレフィックスを使用してメソッドが呼び出されます: `classNS:method()`

上の例に対する説明は以下のとおりです:

`java:` 関数が呼び出されていることを表します。
`classname` がユーザー定義されたクラスの名前になります。
`?` はクラス名と パスを分離するために使用されます。
`path=jar:` により、JAR ファイルへの パスが与えられていることを示します。
`uri-of-jarfile` は JAR ファイルの URI となります。
`!/` は、終了を表すデリミタとなります。
`classNS:method()` により、メソッドの呼び出しが行われます。

その他にも、メソッド名とともにクラス名を与えることができます。構文の例を以下に示します:

```
xmlns:ns1="java:docx.layout.pages?
path=jar:file:///c:/projects/docs/docx.jar!/"
ns1:main()

xmlns:ns2="java?path=jar:file:///c:/projects/docs/docx.jar!/"
ns2:docx.layout.pages.main()
```

以下に JAR ファイルを使った Java 拡張関数を呼び出す XSLT サンプルを記します:

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java?path=jar:file:///C:/test/Car1.jar!/" >
<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
      <xsl:variable name="myCar" select="car:Car1.new('red')" />
      <a><xsl:value-of select="car:Car1.getCarColor($myCar)"/></a>
</xsl:template>

<xsl:template match="car"/>
```

```
</xsl:stylesheet>
```

メモ: 拡張関数によりパスが与えられている場合、ClassLoader にパスが追加されます。

30.2.2.1.3 Java：コンストラクター

拡張関数を使用することで Java コンストラクターを呼び出すことができます。`new()` により全てのコンストラクターを呼び出すことができます。

Java コンストラクターの呼び出し結果を、[黙示的にXPath/XQuery データ型へ変換](2073)できる場合、Java 拡張関数により XPath/XQuery データ型のシーケンスが返されます。Java コンストラクターの呼び出し結果がXPath/XQuery データ型へ変換できない場合、値を返すクラス名でラップした Java オブジェクトがコンストラクターにより作成されます。例えば、`java.util.Date` クラスに対するコンストラクターが呼び出された場合（`java.util.Date.new()`）、`java.util.Date` を持つオブジェクトが返されます。返されたオブジェクトのレキシカルフォーマットは、XPath データ型のレキシカルフォーマットにマッチしない場合もあり、目的のXPath データ型に対するレキシカルフォーマットへ値の変換を行い、その後目的のXPath データ型へ変換を行う必要があります。

コンストラクターにより作成された Java オブジェクトにより2つのことが行えます:

- 変数への割り当てを行うことができます:
  ```
  <xsl:variable name="currentdate" select="date:new()"
  xmlns:date="java:java.util.Date" />
  ```
- 拡張関数への受け渡しを行うことができます（[インスタンスメソッドならびにインスタンスフィールド](2071)を参照ください）:
  ```
  <xsl:value-of select="date:toString(date:new())"
  xmlns:date="java:java.util.Date" />
  ```

30.2.2.1.4 Java：静的メソッドと静的フィールド

静的メソッドは、Java 名ならびにメソッドの引数により直接呼び出すことができます。`E` や `PI` といった定数の静的フィールド（引数を持たないメソッド）は、引数を指定することなくアクセスすることができます。

XSLT の例

静的メソッドならびにフィールドを呼び出す例を以下に示します:

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos( jMath:PI() )" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:E() * jMath:cos(3.14)" />
```

上の拡張関数は `prefix:fname()` という形式を使用していることに注意してください。3つの例にあるプレフィックスは全て jMath: となっており、このプレフィックスは、`java:java.lang.Math` という名前空間 URI に関連付けられています。名前空間 URI は java: で開始しなければなりません。上の例では、クラス名（`java.lang.Math`）を含むように拡張されています。拡張関数の fname() 部は（`java.lang.Math` のような）public クラスにマッチする必要があり、その後 public な静的メソッドが引数とともに続くか（例: `cos(3.14)`）、public な静的フィールドが続きます（例: PI()）。

上の例では、クラス名が名前空間 URI に含まれています。クラス名が名前空間 URI に含まれていない場合、以下の例にあるように、拡張関数の `fname()` 部にて追加する必要があります:

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math.cos(3.14)" />
```

XQuery の例

XQuery における似たようなサンプルを以下に示します:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

30.2.2.1.5 Java：インスタンスメソッドとインスタンスフィールド

メソッド呼び出しの第一引数としてパースされる Java オブジェクトが、インスタンスメソッドには与えられています。このような Java オブジェクトは通常、拡張関数を使うことで作成される(例：コンストラクターの呼び出し)か、スタイルシートパラメーター/変数により作成されます。以下に XSLT サンプルを示します:

```
<xsl:stylesheet version="1.0" exclude-result-prefixes="date"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:date="java:java.util.Date"
  xmlns:jlang="java:java.lang">
  <xsl:param name="CurrentDate" select="date:new()"/>
  <xsl:template match="/">
      <enrollment institution-id="Altova School"
                  date="{date:toString($CurrentDate)}"
                  type="{jlang:Object.toString(jlang:Object.getClass( date:new() ))}">
      </enrollment>
  </xsl:template>
</xsl:stylesheet>
```

上の例では、ノード `enrollment/@type` の値が以下のように作成されます:

1. `java.util.Date` クラスに対するオブジェクトがコンストラクター(`date:new()` コンストラクター)とともに作成されます。
2. `jlang.Object.getClass` メソッドの引数として Java オブジェクトがパースされます。
3. `getClass` メソッドにより得られたオブジェクトが `jlang.Object.toString` メソッドの引数としてパースされます。

結果(`@type` の値)は `java.util.Date` を持った文字列となります。

インスタンスフィールドは、引数としてインスタンスフィールドへ渡される Java オブジェクトではないという点で、理論的にはインスタンスメソッドと異なります。パラメーターや変数がその代わりに引数として渡されますが、パラメーター/変数そのものに Java オブジェクトから返された値が含まれている場合もあります。例えば、`CurrentDate` パラメーターには `java.util.Date` クラスのコンストラクターから返された値が含まれます。この値は引数として、`date:toString` インスタンスメソッドへ渡され、`/enrollment/@date` の値として使用されます。

30.2.2.1.6 データ型：XPath/XQuery から Java へ

XPath/XQuery 条件式内部から Java 関数が呼び出された場合、複数ある同名の Java クラスのうち、どのクラスが呼び出されたのか決定するのに、関数へ渡される引数のデータ型が重要になります。

Java では、以下のルールが適用されます:

- 同名の Java メソッドが2つ以上あり、それぞれが違う数の引数を受け取る場合、呼び出しに使用されている引数の数に一番マッチするメソッドが選択されます。
- XPath/XQuery の文字列、数値、boolean データ型は、黙示的に対応する Java データ型へ変換されます（以下のリストを参照）。与えられた XPath/XQuery 型が2つ以上の Java 型へ変換できる場合（例: `xs:integer`）、選択されたメソッドにて宣言されている Java 型が使用されます。例えば、呼び出された Java メソッドが `fx(decimal)` で、与えられた XPath/XQuery データ型が `xs:integer` の場合、`xs:integer` が Java の `decimal` データ型へ変換されます。

以下のテーブルに、XPath/XQuery の文字列、数値、boolean 型から Java データ型への黙示的な変換リストを示します。

`xs:string`	`java.lang.String`
`xs:boolean`	`boolean`（プリミティブ型）, `java.lang.Boolean`
`xs:integer`	`int`, `long`, `short`, `byte`, `float`, `double`, ならびに、`java.lang.Integer` のようなこれらのラッパークラス
`xs:float`	`float`（プリミティブ型）, `java.lang.Float`, `double`（プリミティブ型）
`xs:double`	`double`（プリミティブ型）, `java.lang.Double`
`xs:decimal`	`float`（プリミティブ型）, `java.lang.Float`, `double`（プリミティブ型）, `java.lang.Double`

上のリストにある XML スキーマデータ型（ならびに XPath や XQuery で使用されているデータ型）のサブタイプも、対応する祖先のサブタイプとして Java のデータ型へ変換されます。

場合によっては、与えられた情報から正しい Java メソッドを選択することができない場合もあります。例えば、以下のような場合を考えてみましょう:

- 与えられた引数が 10 という値を持った xs:untypedAtomic 型で、`mymethod(float)` メソッドへ渡されるのを意図している。
- しかし、そのクラスには別のデータ型を取る `mymethod(double)` というメソッドも存在する。
- メソッド名が同じで、与えられた型（`xs:untypedAtomic`）も `float` と `double` の両方に変換することができるため、`xs:untypedAtomic` が `float` ではなく `double` に変換される可能性もある。
- 結果として、意図したメソッドは選択されず、予期しない動作結果が招く可能性がある。この問題を回避するには、意図したメソッドを使用するユーザー定義のメソッドを別の名前で新たに作成する必要があります。

上のリストでカバーされていない型（例: `xs:date`）は変換されず、エラーとなります。しかし場合によっては、Java コンストラクターを使用して、目的の Java データ型を作成することが可能だということも留意してください。

30.2.2.1.7 データ型：Java から XPath/XQuery へ

Java メソッドにより値が返され、値のデータ型が文字列、数値、または boolean 型の場合、対応する XPath/XQuery 型への変換が行われます。例えば、Java の `java.lang.Boolean` や `boolean` データ型は `xsd:boolean` へ変換されます。

関数から返された一次元配列は、シーケンス（sequence）に展開されます。2 次元以上の配列は変換されることが無いため、ラップして使用するべきでしょう。

Java オブジェクトや文字列、数値、boolean 以外のデータ型がラップされて返された場合、最初に（例えば `toString` といった）Java メソッドを使用して Java オブジェクトを文字列へ変換することで、目的の XPath/XQuery 型への変換を行います。XPath/XQuery では、文字列を目的となる型のレキシカルフォーマットへ変換し、目的の型への変換を（例えば `cast as` 式を使用することで）行うことができます。

30.2.2.2 .NET 拡張関数

.NET プラットフォームにて作業を行なっている場合、.NET 言語（例えば C#）で記述された拡張関数を使用することができます。.NET 拡張関数は XPath や XQuery 条件式内部から使用することができ、.NET クラス内部にあるコンストラクターや（static または インスタンス変数）プロパティを呼び出すことができます。

`get_PropertyName()` 構文を使用することにより .NET クラスのプロパティを呼び出すことができます。

このセクションは、以下のサブセクションにより構成されています:

- .NET コンストラクター 2075
- .NET：静的メソッドならびに静的フィールド 2076
- .NET：インスタンスメソッドとインスタンスフィールド 2076
- データ型：XPath/XQuery から .NET へ 2077
- データ型：.NET から XPath/XQuery へ 2078

拡張関数のフォーム

XPath/XQuery 条件式にある拡張関数は、`prefix:fname()` の形式を取る必要があります。

- `prefix:` 部は、呼び出されている .NET クラスを特定する URI となります。
- `fname()` 部により、.NET クラス内にあるコンストラクター、プロパティ、または（静的または インスタンス）メソッドが特定され、必要な場合は引数が与えられます。
- URI は `clitype:` で開始する必要があり、これにより関数が .NET 拡張関数であることが認識されます。
- 拡張関数の `prefix:fname()` 形式は、システムクラスならびにロードされたアセンブリとともに使用することもできます。しかしクラスをロードする必要がある場合、必要な情報が含まれるパラメーターが必要になります。

パラメーター

アセンブリをロードするには以下のパラメーターを使用してください:

`asm`	ロードするアセンブリの名前。
`ver`	バージョン番号（ピリオドにより分離された最大 4 桁の整数）。

sn	アセンブリ厳密名のキートークン(16新数の数値)。
from	ロードするアセンブリ(DLL) の場所を特定するURI。URI が相対パスの場合、XSLT やXQuery ドキュメントに対して相対的になります。このパラメーターが指定された場合、その他のパラメーターが無視されます。
partialname	アセンブリ名の一部。Assembly.LoadWith.PartialName() へ渡され、アセンブリへのロードが試みられます。 partialname が指定された場合、その他のパラメーターが無視されます。
loc	例えばen-US というロケール。デフォルトはneutral です。

アセンブリがDLL からロードされる場合、 from パラメータが使用して、sn パラメーターは使用しないでください。アセンブリがグローバルアセンブリキャッシュ(GAC) からロードされる場合、sn パラメータを使用して from パラメーターは使用しないでください。

最初のパラメーターの前に疑問符 (?) を挿入し、パラメーター同士はセミコロンで分離する必要があります。パラメーター名へ値を受け渡すには、統合符号 (=) を使用します(以下の例を参照ください)。

名前空間宣言の例

XSLT において、システムクラス System.Environment を特定する名前空間宣言の例を以下に示します:

```
xmlns:myns="clitype:System.Environment"
```

XSLT において、ロードするクラスを Trade.Forward.Scrip として特定する名前空間宣言の例を以下に示します:

```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

XQuery において、システムクラス MyManagedDLL.testClass を特定する名前空間宣言の例を以下に示します。2つのケースが考えられます:

1. アセンブリがGAC からロードされた場合:
```
declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL;
       ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";
```
2. アセンブリがDLL からロードされた場合(完全参照と一部の参照):
```
       declare namespace cs="clitype:MyManagedDLL.testClass?from=file:///C:/Altova
       Projects/extFunctions/MyManagedDLL.dll;

   declare namespace cs="clitype:MyManagedDLL.testClass?from=MyManagedDLL.dll;
```

XSLT の例

システムクラス System.Math 内の関数を呼び出すための完全な XSLT の例を以下に示します:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes" />
   <xsl:template match="/">
      <math xmlns:math="clitype:System.Math">
         <sqrt><xsl:value-of select="math:Sqrt(9)"/></sqrt>
         <pi><xsl:value-of select="math:PI()"/></pi>
         <e><xsl:value-of select="math:E()"/></e>
```

```
        <pow><xsl:value-of select="math:Pow(math:PI(), math:E())"/></pow>
      </math>
   </xsl:template>
</xsl:stylesheet>
```

math 要素にある名前空間宣言により、math: プレフィックスと clitype:System.Math URI が関連付けられます。URI の最初にある clitype: により、それ以降の記述がシステムクラスまたはロードされたクラスを特定するものであることが示されます。XPath 条件式にある math: プレフィックスにより、拡張関数が URI（そしてクラス） System.Math に関連付けられます。拡張関数により、System.Math クラス内のメソッドが特定され、必要な場所に引数が与えられます。

XQuery の例

上の XSLT に対する例と同様の XQuery 例を以下に示します:

```
<math xmlns:math="clitype:System.Math">
   {math:Sqrt(9)}
</math>
```

上の XSLT と同様に、名前空間宣言により .NET クラス（この場合はシステムクラス）が特定されます。XQuery 式により呼び出されるメソッドが特定され、引数が与えられます。

30.2.2.2.1 .NET コンストラクター

拡張関数を使用することで、.NET コンストラクターを呼び出すことができます。new() により全てのコンストラクターを呼び出すことができます。クラス内に2つ以上のコンストラクターがある場合、与えられた引数の数が最もマッチするコンストラクターが選択されます。与えられた引数に対してマッチするコンストラクターが見つからない場合、"No constructor found'" エラーが返されます。

XPath/XQuery データ型を返すコンストラクター

.NET コンストラクター呼び出しの結果が XPath/XQuery データ型へ黙示的に変換[2073]することができる場合、.NET 拡張関数から XPath/XQuery データ型のシーケンスが返されます。

.NET オブジェクトを返すコンストラクター

.NET コンストラクター呼び出しの結果が XPath/XQuery データ型へ適切に変換できない場合、値を返すクラス名でラップした .NET オブジェクトがコンストラクターにより作成されます。例えば、System.DateTime クラスのコンストラクターが（System.DateTime.new() により）呼ばれた場合 System.DateTime 型を持つオブジェクトが返されます。

返されたオブジェクトのレキシカルフォーマットは、目的の XPath データ型と違っている場合があります。その場合、返された値を: (i) 目的の XPath データ型のレキシカルフォーマットへ変換し、(ii) 目的の XPath データ型へキャストする必要があります。

コンストラクターにより作成された .NET オブジェクトに対して3つのことを行うことができます:

- 変数内で使用することができます:
  ```
  <xsl:variable name="currentdate" select="date:new(2008, 4, 29)"
  xmlns:date="clitype:System.DateTime" />
  ```
- 拡張関数へ渡すことができます（インスタンスメソッドとインスタンスフィールド[2071] を参照ください）:
  ```
  <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
  xmlns:date="clitype:System.DateTime" />
  ```
- 文字列、数値、または boolean へ変換することができます:

```
<xsl:value-of select="xs:integer(date:get_Month(date:new(2008, 4, 29)))"
  xmlns:date="clitype:System.DateTime" />
```

30.2.2.2.2 .NET : 静的メソッドと静的フィールド

メソッド名と引数を与えることで、静的メソッドを直接呼び出すことができます。呼び出しに使用される名前は、クラス内にある public static メソッドと完全に一致する必要があります。関数の呼び出しに使用されたメソッド名と引数の数にマッチするものがクラス内に複数ある場合、与えられた引数が評価され、最もマッチするものが選択されます。マッチする結果が得られない場合、エラーが返されます。

メモ: .NET クラス内にあるフィールドは、引数を持たないメソッドとしてみなされます。プロパティは `get_PropertyName()` 構文により呼び出されます。

例

1つの引数とともにメソッド (`System.Math.Sin(arg)`) を呼び出す XSLT サンプルを以下に示します:

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

(引数なしのメソッドとしてみなされる) フィールド (`System.Double.MaxValue()`) を呼び出す XSLT サンプルを以下に示します:

```
<xsl:value-of select="double:MaxValue()" xmlns:double="clitype:System.Double"/>
```

(`get_PropertyName()` 構文を使って) プロパティ (`System.String()`) を呼び出す XSLT サンプルを以下に示します:

```
<xsl:value-of select="string:get_Length('my string')"
xmlns:string="clitype:System.String"/>
```

1つの引数とともにメソッド ((`System.Math.Sin(arg)`) を呼び出す XQuery サンプルを以下に示します:

```
<sin xmlns:math="clitype:System.Math">
   { math:Sin(30) }
</sin>
```

30.2.2.2.3 .NET : インスタンスメソッドとインスタンスフィールド

インスタンスメソッドには、メソッド呼び出しの第一引数として .NET オブジェクトが渡されます。通常この .NET オブジェクトは、拡張関数 (例えばコンストラクター呼び出し) またはスタイルシート パラメーター/変数により作成されます。以下に XSLT の例を示します:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes"/>
   <xsl:template match="/">
      <xsl:variable name="releasedate"
         select="date:new(2008, 4, 29)"
         xmlns:date="clitype:System.DateTime"/>
```

```
      <doc>
         <date>
            <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
               xmlns:date="clitype:System.DateTime"/>
         </date>
         <date>
            <xsl:value-of select="date:ToString($releasedate)"
               xmlns:date="clitype:System.DateTime"/>
         </date>
      </doc>
   </xsl:template>
</xsl:stylesheet>
```

上の例では、System.DateTime コンストラクター(`new(2008, 4, 29)`) が、System.DateTime 型の.NET オブジェクトの作成に使用されます。このオブジェクトは、最初に`releasedate` 変数の値として、次に`System.DateTime.ToString()` メソッドの引数として作成されます。`System.DateTime.ToString()` インスタンスメソッドは、`System.DateTime` コンストラクターの (new(2008, 4, 29)) における引数として2度呼び出されます。これらインスタンスにおいて `releasedate` 変数が.NET オブジェクトを取得するのに使用されます。

インスタンスメソッドとインスタンスフィールド

インスタンスメソッドとインスタンスフィールドの違いは理論的なものです。インスタンスメソッドでは.NET オブジェクトが直接引数に渡され、インスタンスフィールドではパラメーターや変数が(.NET オブジェクトそのものを含めることはできるものの)代わりに渡されます。例えば上の例では、`releasedate` 変数に.NET オブジェクトが含まれており、この変数が2番目のdate 要素コンストラクターにて `ToString()` の引数として渡されます。そのため、最初のdate 要素にある `ToString()` インスタンスがインスタンスメソッドであるのに対し、2番目はインスタンスフィールドとしてみなされます。両方のインスタンスで求められる結果は等価です。

30.2.2.2.4 データ型:XPath/XQuery から .NET へ

.NET 拡張関数がXPath/XQuery 条件式内部で使用された場合、複数ある.NET メソッドのうち、どれが呼び出されたのか決定するのに関数の引数に使用されるデータ型が重要になります。

.NET では、以下のルールが適用されます:

- クラス内に同名のメソッドが2つ以上ある場合、呼び出しに使用された引数の数がマッチするメソッドだけが、呼び出される関数の候補に狭められます。
- XPath/XQuery の文字列、数値、boolean データ型は黙示的に対応する.NET データ型へ変換されます(以下のリストを参照)。与えられたXPath/XQuery 型が2つ以上の.NET 型へ変換できる場合(例: `xs:integer`)、選択されたメソッドにて宣言されている.NET 型が使用されます。例えば、呼び出された.NET メソッドがfx(fouble) で、与えられたXPath/XQuery データ型がxs:integer の場合、`xs:integer` が.NET の`double` データ型へ変換されます。

以下のテーブルに、XPath/XQuery の文字列、数値、boolean 型から.NET データ型へ行われる黙示的な変換リストを示します。

`xs:string`	`StringValue, string`
`xs:boolean`	`BooleanValue, bool`
`xs:integer`	`IntegerValue, decimal, long, integer, short, byte, double, float`
`xs:float`	`FloatValue, float, double`
`xs:double`	`DoubleValue, double`

xs:decimal	DecimalValue, decimal, double, float

上のリストにある XML スキーマ型(ならびに XPath や XQuery で使用されているデータ型)のサブタイプも、対応する祖先のサブタイプとして .NET のデータ型へ変換されます。

場合によっては、与えられた情報から正しい .NET メソッドを選択することができない場合もあります。例えば、以下のような場合を考えてみましょう:

- 与えられた引数が 0 ある xs:untypedAtomic 値で、mymethod(float) メソッドへ渡されるのを意図している。
- しかし、そのクラスには別のデータ型をとる mymethod(double) というメソッドも存在する。
- メソッド名が同じで、与えられた型 (xs:untypedAtomic) も float と double の両方に変換することができるため、xs:untypedAtomic が float ではなく double に変換される可能性もある。
- 結果として、意図したメソッドは選択されず、予期しない動作結果が招く可能性がある。この問題を回避するには、意図したメソッドを使用するユーザー定義のメソッドを別の名前で新たに作成する必要があります。

上のリストでカバーされていない型(例: xs:date)は変換されずエラーとなります。

30.2.2.2.5 データ型 : .NET から XPath/XQuery へ

.NET メソッドにより値が返される際に値のデータ型が文字列、数値、または boolean 型の場合、対応する XPath/XQuery 型への変換が行われます。例えば、.NET の decimal データ型は xsd:decimal へ変換されます。

.NET オブジェクトや文字列、数値、boolean 以外のデータ型が返された場合、最初に(例えば System.DateTime.ToString() といった).NET メソッドを使用して .NET オブジェクトを文字列へ変換します。XPath/XQuery では、文字列を目的となる型のレキシカルフォーマットへ変換し、目的の型への変換を(例えば cast as 式を使用することで)行うことができます。

30.2.2.3 XSLT に対する MSXSL スクリプト

<msxsl:script> 要素にはユーザー定義の関数や変数が含まれており、XSLT スタイルシート内の XPath 条件式内部から呼び出しを行うことができます。<msxsl:script> はトップレベル要素で、<xsl:stylesheet> または <xsl:transform> の子要素である必要があります。

<msxsl:script> 要素は urn:schemas-microsoft-com:xslt 名前空間内に存在する必要があります(以下を参照ください)。

スクリプト言語と名前空間

ブロック内で使用されるスクリプト言語は <msxsl:script> 要素の language 属性にて指定され、XPath 条件式における関数の呼び出しに対して使用される名前空間は implements-prefix 属性により特定されます(以下を参照)。

```
<msxsl:script language="scripting-language" implements-prefix="user-namespace-prefix">

   function-1 or variable-1
   ...
   function-n or variable-n

</msxsl:script>
```

<msxsl:script> 要素は Windows Scripting Runtime を使ったやりとりを行うため、お使いのコンピューターにインストールされた言語だけか <msxsl:script> 要素では使用することができません。MSXSL スクリプトを使用するには .NET Framework 2.0 以上のプラットフォームをインストールする必要があります。結果として <msxsl:script> 言語から .NET スクリプト言語を使用することができます。

HTML の <script> 要素における language 属性と同じ値が language 属性では受理されます。language 属性が指定されていない場合、Microsoft JScript がデフォルトとして想定されます。

implements-prefix 属性には名前空間スコープ内で宣言されたプレフィックスが与えられます。通常この名前空間は関数ライブラリのために予約されたユーザーの名前空間となります。<msxsl:script>要素内で定義された全ての関数ならびに変数は、implements-prefix 属性にて指定されたプレフィックスで特定される名前空間に収められます。XPath 条件式内部から関数が呼ばれる場合、完全修飾関数名が同じ名前空間内に関数として定義されていなければなりません。

サンプル

<msxsl:script> 要素内で定義された関数を使用する XSLT スタイルシートの例を以下に示します:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   xmlns:msxsl="urn:schemas-microsoft-com:xslt"
   xmlns:user="http://mycompany.com/mynamespace">

  <msxsl:script language="VBScript" implements-prefix="user">
    <![CDATA[
    ' Input:A currency value: the wholesale price
    ' Returns:The retail price: the input value plus 20% margin,
    ' rounded to the nearest cent
    dim a as integer  = 13
    Function AddMargin(WholesalePrice) as integer
      AddMargin = WholesalePrice * 1.2 + a
    End Function
  ]]>
  </msxsl:script>

  <xsl:template match="/">
    <html>
      <body>
        <p>
          <b>Total Retail Price =
            $<xsl:value-of select="user:AddMargin(50)"/>
          </b>
          <br/>
          <b>Total Wholesale Price =
            $<xsl:value-of select="50"/>
          </b>
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

データ型

スクリプトブロックとのやりとりに使用されるパラメーターの値は XPath データ型に限定されます。スクリプトブロック内にある関数にてやりとりされるデータや変数に、この制限はありません。

アセンブリ

msxsl:assembly 要素を使用することで、アセンブリをスクリプト内部へインポートすることができます。アセンブリは名前やURI により特定されます。アセンブリのインポートは、コンパイル時に行われます。以下に msxsl:assembly 要素の簡単な使用例を示します:

```
<msxsl:script>
       <msxsl:assembly name="myAssembly.assemblyName" />
       <msxsl:assembly href="pathToAssembly" />

       ...

</msxsl:script>
```

アセンブリ名は、以下のような完全な名前でも::

```
"system.Math, Version=3.1.4500.1 Culture=neutral PublicKeyToken=a46b3f648229c514"
```

`"myAssembly.Draw"` のような短い名前でも指定できます。

名前空間

msxsl:using 要素により名前空間の宣言を行うことができます。これにより、スクリプト内において名前空間無しでアセンブリクラスを使用することができ、タイピングの手間を軽減することができます。以下に `msxsl:using` 要素の簡単な使用例を示します:

```
<msxsl:script>
       <msxsl:using namespace="myAssemblyNS.NamespaceName" />

       ...

</msxsl:script>
```

`namespace` 属性の値は名前空間の名前となります。

(C) 2015-2021 Altova GmbH

30.3 DB 内で生成された XML スキーマのデータ型

XML スキーマがデータベース(DB)から生成されると、そのデータベース特有のデータ型がXML スキーマデータ型に変換されます。一般に使用されるDB データ型からXML スキーマデータ型へのマッピングは付録に記載されています。以下のリストから選択してください。

- ADO 2081
- MS Access 2082
- MS SQL Server 2082
- MySQL 2083
- ODBC 2084
- Oracle 2085
- Sybase 2085

30.3.1 ADO

XML スキーマがADO データベース(DB) から生成される場合、ADO DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

ADO データ型	XML スキーマ データ型
adGUID	xs:ID
adChar	xs:string
adWChar	xs:string
adVarChar	xs:string
adWVarChar	xs:string
adLongVarChar	xs:string
adWLongVarChar	xs:string
adVarWChar	xs:string
adBoolean	xs:boolean
adSingle	xs:float
adDouble	xs:double
adNumeric	xs:decimal
adCurrency	xs:decimal
adDBTimeStamp	xs:dateTime
adDate	xs:date
adBinary	xs:base64Binary
adVarBinary	xs:base64Binary
adLongVarBinary	xs:base64Binary
adInteger	xs:Integer

adUnsignedInt	xs:unsignedInt
adSmallInt	xs:short
adUnsignedSmallInt	xs:unsignedShort
adBigInt	xs:long
adUnsignedBigInt	xs:unsignedLong
adTinyInt	xs:byte
adUnsignedTinyInt	xs:unsignedByte

30.3.2 MS Access

XML スキーマがMS Access データベース(DB) から生成される場合、MS Access DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

MS Access データ型	XML スキーマ データ型
GUID	xs:ID
char	xs:string
varchar	xs:string
memo	xs:string
bit	xs:boolean
Number(single)	xs:float
Number(double)	xs:double
Decimal	xs:decimal
Currency	xs:decimal
Date/Time	xs:dateTime
Number(Long Integer)	xs:integer
Number(Integer)	xs:short
Number(Byte)	xs:byte
OLE Object	xs:base64Binary

30.3.3 MS SQL Server

XML スキーマがMS SQL Server データベース(DB) から生成される場合、MS SQL Server DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

MS SQL Server データ型	XML スキーマ データ型
uniqueidentifier	xs:ID

(C) 2015-2021 Altova GmbH

char	xs:string
nchar	xs:string
varchar	xs:string
nvarchar	xs:string
text	xs:string
ntext	xs:string
sysname	xs:string
bit	xs:boolean
real	xs:float
float	xs:double
decimal	xs:decimal
money	xs:decimal
smallmoney	xs:decimal
datetime	xs:dateTime
smalldatetime	xs:dateTime
binary	xs:base64Binary
varbinary	xs:base64Binary
image	xs:base64Binary
integer	xs:integer
smallint	xs:short
bigint	xs:long
tinyint	xs:byte

30.3.4 MySQL

XML スキーマがMySQL データベース(DB) から生成される場合、MySQL DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

MySQL データ型	XML スキーマ データ型
char	xs:string
varchar	xs:string
text	xs:string
tinytext	xs:string
mediumtext	xs:string
longtext	xs:string

tinyint(1)	xs:boolean
float	xs:float
double	xs:double
decimal	xs:decimal
datetime	xs:dateTime
blob	xs:base64Binary
tinyblob	xs:base64Binary
mediumblob	xs:base64Binary
longblob	xs:base64Binary
smallint	xs:short
bigint	xs:long
tinyint	xs:byte

30.3.5 ODBC

XML スキーマがODBC データベース(DB) から生成される場合、ODBC DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

ODBC データ型	XML スキーマ データ型
SQL_GUID	xs:ID
SQL_CHAR	xs:string
SQL_VARCHAR	xs:string
SQL_LONGVARCHAR	xs:string
SQL_BIT	xs:boolean
SQL_REAL	xs:float
SQL_DOUBLE	xs:double
SQL_DECIMAL	xs:decimal
SQL_TIMESTAMP	xs:dateTime
SQL_DATE	xs:date
SQL_BINARY	xs:base64Binary
SQL_VARBINARY	xs:base64Binary
SQL_LONGVARBINARY	xs:base64Binary
SQL_INTEGER	xs:integer
SQL_SMALLINT	xs:short
SQL_BIGINT	xs:long

(C) 2015-2021 Altova GmbH

SQL_TINYINT	xs:byte

30.3.6 Oracle

XML スキーマがOracle データベース(DB) から生成される場合、Oracle DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

Oracle データ型	XML スキーマ データ型
ROWID	xs:ID
CHAR	xs:string
NCHAR	xs:string
VARCHAR2	xs:string
NVARCHAR2	xs:string
CLOB	xs:string
NCLOB	xs:string
NUMBER (with check constraint applied)*	xs:boolean
NUMBER	xs:decimal
FLOAT	xs:double
DATE	xs:dateTime
INTERVAL YEAR TO MONTH	xs:gYearMonth
BLOB	xs:base64Binary

* データ型 NUMBER の列にチェック制約が適用されている場合、またチェック制約が0 または1 の値をチェックする場合、この列の NUMBER データ型は、 xs:boolean のXML スキーマデータ型に変換されます。このメカニズムは、生成されたXML スキーマ内にxs:boolean データ型を生成する際に役に立ちます。

30.3.7 Sybase

XML スキーマがSybase データベース(DB) から生成される場合、Sybase DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

Sybase データ型	XML スキーマ データ型
char	xs:string
nchar	xs:string
varchar	xs:string
nvarchar	xs:string
text	xs:string

sysname-varchar(30)	xs:string
bit	xs:boolean
real	xs:float
float	xs:float
double	xs:double
decimal	xs:decimal
money	xs:decimal
smallmoney	xs:decimal
datetime	xs:dateTime
smalldatetime	xs:dateTime
timestamp	xs:dateTime
binary<=255	xs:base64Binary
varbinary<=255	xs:base64Binary
image	xs:base64Binary
integer	xs:integer
smallint	xs:short
tinyint	xs:byte

(C) 2015-2021 Altova GmbH

30.4 XML スキーマから生成された DB のデータ型

データベース構造がXML スキーマがから生成されると、そのデータベース特有のデータ型がXML スキーマデータ型から生成されます。一般に使用されるXML スキーマデータ型からDB データ型へのマッピングは付録に記載されています。以下のリストから選択してください。

- MS Access 2087
- MS SQL Server 2088
- MySQL 2090
- Oracle 2092

30.4.1 MS Access

MS Access データベース(DB) がXML スキーマから生成される場合、MS Access データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

XML スキーマ データ型	MS Access データ型
xs:ID	GUID
xs:string	If no facets varchar (255)
	Size = either length or maxLength
	If Size <= 255 varchar (n)
	else memo
xs:normalizedString	Same as xs:string
xs:token	Same as xs:string
xs:Name	Same as xs:string
xs:NCName	Same as xs:string
xs:anyURI	Same as xs:string
xs:QName	Same as xs:string
xs:NOTATION	Same as xs:string
xs:boolean	bit
xs:float	Number (single)
xs:double	Number (double)
xs:decimal	Decimal
xs:duration	Date/Time
xs:dateTime	Date/Time
xs:time	Date/Time
xs:date	Date/Time
xs:gYearMonth	Date/Time

xs:gYear	Date/Time
xs:gMonthDay	Date/Time
xs:gDay	Date/Time
xs:gMonth	Date/Time
xs:hexBinary	If no facets varbinary (255)
	Size = either length or maxLength
	If Size <= 8000 varbinary
	else image (OLE Object)
xs:base64Binary	Same as xs:hexBinary
xs:integer	Number (Long Integer)
xs:int	Number (Long Integer)
xs:negativeInteger	Number (Long Integer); value constraint
xs:positiveInteger	Number (Long Integer); value constraint
xs:nonNegativeInteger	Number (Long Integer); value constraint
xs:nonPositiveInteger	Number (Long Integer); value constraint
xs:unsignedInt	Number (Long Integer)
xs:short	-- no equivalent --
xs:unsignedShort	-- no equivalent --
xs:long	-- no equivalent --
xs:unsignedLong	-- no equivalent --
xs:byte	Number (Byte)
xs:unsignedByte	Number (Byte)

30.4.2 MS SQL Server

MS SQL Server データベース(DB) がXML スキーマから生成される場合、MS SQL Server データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

XML スキーマ データ型	MS SQL Server データ型
ID	uniqueidentifier
xs:string	If no facets
	{ if UNICODE nvarchar (255)
	else varchar (255) }
	else
	{ if UNICODE

	(Size = either length or maxLength)
	If Size <= 4000
	if FacetLengthIsSet then nChar
	else nVarChar
	if Size <= 1073741823 then nText }
	else
	{ if NON-UNICODE
	(Size = either length or maxLength)
	If Size <= 8000
	if FacetLengthIsSet then char
	else varchar
	if Size <= 2147483647 then text }
xs:normalizedString	Same as xs:string
xs:token	Same as xs:string
xs:Name	Same as xs:string
xs:NCName	Same as xs:string
xs:anyURi	Same as xs:string
xs:QName	Same as xs:string
xs:NOTATION	Same as xs:string
xs:boolean	bit
xs:float	real
xs:double	float
xs:decimal	decimal
xs:duration	datetime
xs:dateTime	datetime
xs:time	datetime
xs:date	datetime
xs:gYearMonth	datetime
xs:gYear	datetime
xs:gMonthDay	datetime
xs:gDay	datetime
xs:gMonth	datetime
xs:hexBinary	If no facets varbinary (255)
	(Size = either length or maxLength
	If Size <= 8000

	if FacetLengthIsSet then binary
	else varbinary
	if Size <= 2147483647 then image
xs:base64Binary	Same as xs:hexBinary
xs:integer	int
xs:int	int
xs:negativeInteger	Int (constrained to {...,-2,-1})
xs:positiveInteger	Int (constrained to {1,2,...})
xs:nonNegativeInteger	int (constrained to {0,1,2,...})
xs:nonPositiveInteger	int (constrained to {...,-2,-1,0})
xs:unsignedInt	int (additional constraints)
xs:short	smallint
xs:unsignedShort	smallint (additional constraints)
xs:long	bigint
xs:unsignedLong	bigint (additional constraints)
xs:byte	tinyint
xs:unsignedByte	tinyint (additional constraints)

30.4.3 MySQL

MySQL データベース(DB) がXML スキーマから生成される場合、MySQL データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

XML スキーマ データ型	MySQL データ型
xs:ID	varchar(255)
xs:string	If no facets then varchar (255)
	else if facet length is set and <= 255 then char
	else if facet maxLength set and <= 255 then varchar
	else if maxLength is set and <= 65545 then text
	else if maxlength is set and <= 16777215 then mediumtext
	else if maxlength is set and <= 429496295 then longtext
xs:normalizedString	*Same as* xs:string

(C) 2015-2021 Altova GmbH

xs:token	*Same as* xs:string
xs:Name	*Same as* xs:string
xs:NCName	*Same as* xs:string
xs:anyURI	*Same as* xs:string
xs:QName	*Same as* xs:string
xs:NOTATION	*Same as* xs:string
xs:boolean	tinyint(1)
xs:float	float
xs:double	double
xs:decimal	decimal
xs:duration	timestamp
xs:dateTime	datetime
xs:time	time
xs:date	date
xs:gYearMonth	timestamp(4)
xs:gYear	year(4)
xs:gMonthDay	timestamp(8); constraints to check month, day
xs:gDay	timestamp(8); constraints to check day
xs:gMonth	timestamp(8); constraints to check month
xs:hexBinary	If no facets then blob (255)
	else if facet length is set and <= 255 then blob
	else if facet maxLength is set and <= 255 then tinyblob
	else if maxlength is set and <= 65545 then blob
	else if maxlength is set and <= 16777215 then mediumblob
	else if maxlength is set and <= 429496295 then longblob
xs:base64Binary	*Same as* xs:hexBinary
xs:integer	Integer
xs:int	int
xs:negativeInteger	Integer (constrained to {...,-2,-1})
xs:positiveInteger	Integer (constrained to {1,2,...})
xs:nonNegativeInteger	Integer (constrained to {0,1,2,...})

xs:nonPositiveInteger	Integer (constrained to {...,-2,-1,0})
xs:unsignedInt	Int (additional constraints)
xs:short	Smallint
xs:unsignedShort	Smallint (additional constraints)
xs:long	Bigint
xs:unsignedLong	Bigint (additional constraints)
xs:byte	Tinyint
xs:unsignedByte	Tinyint (additional constraints)

30.4.4 Oracle

Oracle データベース(DB) がXML スキーマから生成される場合、Oracle データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

XML スキーマ データ型	Oracle データ型
xs:ID	ROWID
xs:string	If no facets
	if UNICODE then NVARCHAR2 (255)
	else VARCHAR2 (255)
	else if UNICODE
	(Size = either length or maxLength)
	If Size <= 2000 then NCHAR
	if Size <= 4000 then NVARHCAR2
	if Size <= 4 Gigabytes then NCLOB
	else if NON-UNICODE
	(Size = either length or maxLength)
	If Size <= 2000 then CHAR
	if Size <= 4000 then VARCHAR2
	if Size <= 4 Gigabytes then CLOB
xs:normalizedString	*Same as* xs:string
xs:token	*Same as* xs:string
xs:Name	*Same as* xs:string
xs:NCName	*Same as* xs:string
xs:anyURI	*Same as* xs:string
xs:QName	*Same as* xs:string

(C) 2015-2021 Altova GmbH

xs:NOTATION	*Same as* xs:string
xs:boolean	NUMBER with constraint Boolean
xs:float	FLOAT
xs:double	FLOAT
xs:decimal	NUMBER
xs:duration	TIMESTAMP
xs:dateTime	TIMESTAMP
xs:time	DATE
xs:date	DATE
xs:gYearMonth	INTERVAL YEAR TO MONTH
xs:gYear	DATE
xs:gMonthDay	DATE
xs:gDay	DATE
xs:gMonth	DATE
xs:hexBinary	if no facets then RAW (255)
	(Size = either length or maxLength)
	If Size <= 2000 then RAW (X)
	else Size <= 2 Gigabytes then LONG RAW (X)
	if Size <= 4 Gigabytes then BLOB (X)
xs:base64Binary	BLOB
xs:integer	NUMBER
xs:int	NUMBER
xs:negativeInteger	NUMBER (constrained to {...,-2,-1})
xs:positiveInteger	NUMBER (constrained to {1,2,...})
xs:nonNegativeInteger	NUMBER (constrained to {0,1,2,...})
xs:nonPositiveInteger	NUMBER (constrained to {...,-2,-1,0})
xs:unsignedInt	NUMBER (additional constraints)
xs:short	NUMBER
xs:unsignedShort	NUMBER (additional constraints)
xs:long	NUMBER
xs:unsignedLong	NUMBER (additional constraints)
xs:byte	BLOB
xs:unsignedByte	BLOB (additional constraints)

30.5 技術データ

このセクションは、ソフトウェアの技術面に関する役に立つ背景情報を含んでいます。以下のように整理されています:
OS とメモリ要件[2094] Altova XML バリデーター[2094] Altova XSLT とXQuery エンジン[2094] Unicode のサポート[2094] インターネットの使用[2095]

30.5.1 OS とメモリ要件

オペレーティングシステム

Altova ソフトウェア アプリケーションは、以下のプラットフォームでご使用いただけます:
プラットフォーム更新済みのW indows 7 SP1、W indows 8、W indows 10プラットフォーム更新済みのW indows Server 2008 R2 SP1 または以降 **メモリ**
ソフトウェアがC++ で書かれているため、Java Runtime Environment をダウンロードする必要はなく、Java ベースのアプリケーションに比べ、通常少ないメモリを必要とします。しかしながら、各ドキュメントは完全に解析するため、また、ビューと編集の速度を向上するためにメモリにダウンロードされます。メモリの要件は、ドキュメントのサイズを増やします。

メモリ要件は、制限のない「元に戻す」履歴により影響を受けます。大きなドキュメントの大きなセクションの切り取り、貼り付け操作を繰り返し行うと、使用できるメモリがすぐに消費されます。

30.5.2 Altova XML バリデーター

XML ドキュメントを開くと、アプリケーションは、内蔵のXML バリデーターを使用して、指定されている場合、スキーマに対して整形式をチェックし、ツリーとインフォセットを作成します。XML バリデーターは、ドキュメントを編集する際にインテリジェントな編集ヘルプを提供し、発生する検証エラーを表示するために使用されます。

内蔵のXML バリデーターは、W 3C のXML スキーマ1.0 と1.1 仕様の最終勧告を実装しています。. New developments recommended by the W 3C XML スキーマ作業グループにより勧告される新しい項目は、XML バリデーターに継続的に組み込まれるため、Altova 製品は最高水準の開発環境を届けることができます。

30.5.3 Altova XSLT と XQuery エンジン

Altova 製品 は、Altova XSLT 1.0、2.0、および3.0 エンジンとAltova XQuery 1.0 と3.1 エンジンを使用しています。各エンジンのためのドキュメントと実装に固有の振る舞いに関しては、製品で使用されるエンジンの各ドキュメントの付属書(エンジン情報) で確認することができます。

メモ Altova MapForce は、XSLT 1.0、2.0 およびXQuery 1.0 エンジンを使用したコードを生成します。

30.5.4 Unicode のサポート

Altova XML 製品は、Unicode を完全にサポートします。XML ドキュメントを編集するには、ドキュメント内で使用されているUnicode 文字をサポートするフォントが必要です。

フォントの多くは、Unicode 範囲全体の特定のサブセットを含む場合があり、このため、通常は対応する表記システムをターゲットとします。テキストの一部が、文字化けして表示された場合、理由としては、選択されたフォントが必要とする字形を含まない場合があげられます。です

から、特に、異なる言語、または、異なる言語システムのXMLドキュメントを編集する場合、範囲全体をカバーするフォントを使用することが役にたちます。典型的なUnicode フォントは、W indows PC のArial Unicode MS で確認することができます。

アプリケーションフォルダーの/Examples フォルダー内で、異なる言語システムで表記された次の文章を含むUnicodeUTF-8.html というXHTML ファイルを確認してください:
When the world wants to talk, it speaks Unicode Wenn die Welt miteinander spricht, spricht sie Unicode
世界的に話すなら、Unicode です。) XHTML ファイルを開くと、Unicode の可能性を確認することができ、使用中のPC の使用することのできるフォントによりサポートされている表記システムが表示されます。

30.5.5 インターネットの使用

Altova アプリケーションは、次の状況でインターネット 接続を開始します:

- 登録ダイアログ(「**ヘルプ | ソフトウェアのライセンス認証**」)内の「評価キーコードをリクエスト」をクリックした場合、登録ダイアログボックス内の3つのフィールドが通常のhttp (ポート 80)接続を使用し、サーバーに転送され、無料の評価キーが顧客に通常のSMTP 電子メールを使用して送り返されます。
- Altova 製品の一部では、インターネットからファイルを開くことができます(「**ファイル | 開く | URL に切り替える**」)。この場合、ドキュメントは、次のプロトコルメソッドと接続の1つを使用して取得されます: HTTP (通常、ポート 80)、FTP (通常、ポート 20/21)、HTTPS (通常、ポート 443)。HTTP サーバーをポート 8080 で作動することもできます(URL ダイアログ内で、サーバー名とコロンの後にポートを指定します)。
- XML スキーマ、または、DTD を参照するXMLドキュメント、と、URL により指定されているドキュメントを開くと、参照されているスキーマドキュメントは、HTTP 接続(ポート 80) またはURL により指定されている他のプロトコル(上のポイント 2 参照)により抽出されます。XML ファイルが検証されている場合、スキーマドキュメントも抽出されます(オプションダイアログのファイルタブ内の(「**ツール | オプション**」))。アプリケーションに命令している場合、ドキュメントが開かれると検証が自動的に行われる場合もあります。
- W SDL とSOAP を使用するAltova アプリケーションでは、W eb サービスを使用する接続は、W SDLドキュメントにより定義されています。
- XMLSpy 内で、「**電子メールで送信**」コマンドを使用する場合、(「**ファイル | 電子メールで送信**」) 現在選択されている範囲、または、ファイルは、ユーザーのマシンにインストールされているMAPI コンプライアント電子メールプログラムにより送信されます。
- ソフトウェアの荒いセンス認証 とLiveUpdate の一部として、Altova ソフトウェア使用許諾書内で更に詳しい説明を確認することができます。

30.6 ライセンス情報

このセクションには以下の内容が含まれています:

- ソフトウェアの配布に関する情報
- ソフトウェアのアクティベーションとライセンスの計測
- ソフトウェアの使用に関する使用許諾契約書

本製品を使用する前に、上記の情報をよくお読みください。ソフトウェアのインストール時に上記のすべての条件に同意したとみなされ、お客様は上記の条件に拘束されることを同意したとみなされます。

Altova ライセンスの内容を確認するには、Altova Web サイト の Altova 法的な情報のページ に移動してください。

30.6.1 電子的なソフトウェアの配布

この製品は電子的なソフトウェアの配布により利用することが可能で、この配布方法により、以下のユニークなメリットがあります:

- 購入を決定する前に、無料でソフトウェアを試用することができます。*(Note: Altova MobileTogether Designer に対してライセンスを無料で割り当てることができます)。*
- Once ソフトウェアの購入を決定した際には、Altova Web サイト にて注文を行います。すぐにライセンス登録された製品の使用を開始することができます。
- オンラインにて注文を行うと、常に最新のソフトウェアをご利用いただけます。
- 製品パッケージには包括的なヘルプシステムが画面上に表示されます。最新バージョンのユーザーマニュアルは https://www.altova.com/ja/ 上にあり、(i) HTML フォーマットによる閲覧、ならびに(ii) PDF フォーマットのダウンロードと印刷に対応しております。

30 日間の評価期間

この製品をダウンロードした後は、最大で30 日の間無料で製品の評価を行うことができます。20 日間を超えた頃から、製品がライセンス登録されていないことがソフトウェアにより表示されます。このメッセージはアプリケーションが起動されるたびに表示され、30 日間を超えてプログラムを使用するには、キーコードを含むライセンスファイルから提供される製品のライセンスを購入します。ライセンスファイルを製品のソフトウェアアクティベーションダイアログにアップロードして、製品をアンロックします。

https://shop.altova.com/ でライセンスを購入することができます

組織内でソフトウェアの評価を行う

評価版のソフトウェアを組織内のネットワークにて配布したい場合、またはインターネットに接続されていないコンピューターにてソフトウェアを使用する場合、どのような状態でも改変されていないことを条件に、セットアッププログラムおよびの配布を行うことが可能です。ソフトウェアインストーラーへアクセスした人は、例外なく30 日間の評価ライセンスキーコードをリクエストして、試用期間が経過した後は、製品を使い続けるためにライセンスの購入を行う必要があります。

30.6.2 ソフトウェアのアクティベーションとライセンスの計測

Altova のソフトウェアアクティベーションの一部として、ソフトウェアにより内部ネットワークまたはインターネットへの接続を行い、インストール時、登録時、Altova により使用されるライセンスサーバーの更新やライセンスの正当性を検証することで、ソフトウェアの不正な使用を防ぎ、顧客サービスを向上するため、ライセンスに関する情報を送信することがあります。アクティベーションにより、オペレーティングシステムや IP アドレス、

(C) 2015-2021 Altova GmbH

日付/時刻、ソフトウェアのバージョン、コンピュータの名前などのライセンスに関する情報が、お使いのコンピューターとAltova ライセンスサーバー間にてやり取りされます。

お使いのAltova 製品にはライセンス計測モジュールが内蔵されており、エンドユーザー使用許諾契約書の意図しない違反を防ぎます。お使いの製品はシングルユーザーまたはマルチユーザーとしてインストールされており、ライセンス計測モジュールにより、ライセンスされている数を超えたユーザーが同時に製品を使用することが無いことが保証されます。

このライセンス計測技術により、ローカルエリア接続（LAN）において、別々のコンピューター間で動作しているアプリケーションインスタンス間の通信が行われます。

シングルライセンス

ライセンス計測プロセスの一部としてアプリケーションが起動すると、ソフトウェアにより短いデータグラムがブロードキャストにより送信され、同一のネットワークセグメントにある他のコンピューターにてプログラムが動作していないかのチェックが行われます。応答が無い場合は、アプリケーションの他インスタンスから送信される信号に応えるため、ポートが開かれます。

マルチユーザーライセンス

同一のLAN 内にて2つ以上のアプリケーションインスタンスが使用された場合、スタートアップ時に、これらインスタンス間において通信が行われます。これらのインスタンス間にてキーコードのやりとりが行われ、購入された数のライセンスを超えてインスタンスが起動しないように保証することができます。このようなライセンス計測システムはUNIX やデータベース開発ツールにて広く使用されているもので、Altova ユーザーはリーズナブルな価格にて同時使用マルチユーザーライセンスを購入することができます。

弊社はアプリケーションのデザインも行っており、少数の小さなネットワークパケットを送信することで、ネットワークに対する負荷を最小限に抑えております。Altova により使用される2799番 TCP/IP ポートはIANA により公式登録されており（詳細は（ IANA W eb サイト (http://www.iana.org/) を参照ください）、弊社のライセンス計測モジュールも既にテストされたものです。

ファイヤーウォールを使用している場合、2799番ポートにてAltova 製品が動作しているコンピューター同士が通信しているのに気づかれるかも知れません。その他の手段によりライセンス使用許諾書の内容が守られることを保証できる限り、組織間の異なるグループにおいてこのようなトラフィックをブロックすることは勿論可能です。

証明書に関するメモ

Altova アプリケーションはHTTPS を介してAltova ライセンスサーバー（`link.altova.com`）に通信します。この通信のためにAltova は登録済みのSSL 証明書を使用します。（例えば、社内IT 部署または外部エージェンシーにより）この証明書が置き換えられている場合、使用中のAltova アプリケーションは接続が安全でないことを警告します。Altova アプリケーションを開始するために代替の証明書を使用することができますが、自己責任で行ってください。*安全ではない接続* の警告メッセージが表示されると、証明書の発行元を確認して（ Altova 証明書の代替証明書の使用の継続または停止を決定することができる）社内IT チームと相談してください。

（例えば、クライアントマシンへの、または、クライアントマシンへの通信を監視するために）自身の証明書の使用が必要な場合 Altova の無料管理ソフトウェアである Altova LicenseServer を使用中のネットワークにインストールすることが奨励されます。このセットアップでは、Altova LicenseServer はAltova との通信のためにAltova 証明書の使用を許可しつつクライアントマシンが所属機関の証明書の使用を継続することができます。

30.6.3 エンドユーザー使用許諾契約書

- Altova エンドユーザー使用許諾契約書: http://www.altova.com/ja/legal/eula
- Altova プライバシーポリシー: http://www.altova.com/ja/privacy

インデックス

, 262

.

.docx, 257, 834
.NET,
XMLSpy との統合, 1005
スタンドアロン XMLSpy との違い, 1007
と XMLSpy デバッガー, 1009
.NET 拡張関数,
XSLT と XQuery, 2073
インスタンスメソッド、インスタンスフィールド, 2076
コンストラクター, 2075
データ型変換、.NET から XPath/XQuery へ, 2078
データ型変換、XPath/XQuery から .NET へ, 2077
概要, 2073
静的メソッド、静的フィールド, 2076
.NET 内のXSLT と XQuery のための拡張機能,
.NET 拡張関数を参照する, 2073
.pptx, 257, 834
.xslx, 257, 834

1

1行の長さ,
テキストビューにてワードラップ, 1362

A

AAIDC ペイン, 195
ActiveX,
アプリケーションレベルの統合, 1908
ドキュメントレベルの統合, 1910
統合の必要条件, 1905
ActiveX controls,
support, 1531
ActiveX コントロール,
Visual Studio ツールボックスへの追加, 1906
ADO,
データ接続インターフェイスとして, 847
接続のセットアップ, 852
ADO.NET,
接続のセットアップ, 859
Altova XML パーサー,
について, 2094
Altova エンジン,
Altova 製品内で, 2094
Altova グローバルリソース,
グローバルリソース, 934
グローバルリソース を参照する, 934
Altova サポート, 125
Altova ソフトウェアの注文, 1498
Altova ソフトウェアの登録, 1498
Altova 拡張関数,
チャート関数 (チャート関数を参照), 1981
Altova 製品, 125
Ant,
setting the environment variables for,, 1107
API,
documentation, 1544
JAVA, 1857
JAVA Classpath, 1857
overview, 1545
Application,
ActiveDocument, 1583
AddMacroMenuItem, 1583
AddXSLT_XQParameter, 1584
Application, 1584
ClearMacroMenu, 1584
CurrentProject, 1585
Dialogs, 1585
Documents, 1585
GetDatabaseImportElementList, 1586
GetDatabaseSettings, 1587
GetDatabaseTables, 1587
GetExportSettings, 1588
GetTextImportElementList, 1588
GetTextImportExportSettings, 1589
GetXSLT_XQParameterCount, 1589
GetXSLT_XQParameterName, 1590
GetXSLT_XQParameterXPath, 1590
ImportFromDatabase, 1590
ImportFromSchema, 1591

(C) 2015-2021 Altova GmbH

Application,
ImportFromText, 1592
ImportFromWord, 1593
NewProject, 1594
OnBeforeOpenDocument, 1581
OnBeforeOpenProject, 1582
OnDocumentOpened, 1582
OnProjectOpened, 1583
OpenProject, 1594
Parent, 1594
Quit, 1595
ReloadSettings, 1595
RemoveXSLT_XQParameter, 1595
RunMacro, 1596
ScriptingEnvironment, 1596
ShowApplication, 1597
ShowForm, 1597
URLDelete, 1598
URLMakeDirectory, 1598
WarningNumber, 1599
WarningText, 1599
Archive View, 257, 834
と EPUB ファイル, 842
と OOXML ファイル, 836
と ZIP ファイル, 840
ATL,
plug-in sample files, 1534
ATTLIST 宣言,
グリッドビューにて子として追加, 1219
グリッドビューにて挿入, 1204
グリッドビューにて追加, 1212
グリッドビューにて変換, 1222
AttributeFormDefault,
スキーマデザインビューにおける設定, 1256
Authentic View, 558, 1297
CDATA セクション, 535
DB データの編集, 1295
GUI の概要, 540
PXF ファイルから出力ドキュメントを生成, 1303
SPS テーブル, 557
XML テーブル, 558
XML テーブルアイコン, 562
XML テーブルの使用, 558
XML ドキュメントの印刷, 538
XML ドキュメントを開く, 529
XML/テキストとして貼り付け, 550
エンティティ, 535
エンティティの挿入, 537
コンテキストメニュー, 530
ツールバーアイコン, 541
データエントリデバイス, 535
データの挿入, 535
テーブル (SPS と XML), 557
テーブル内, 532
テキストの書式設定, 541
テキストの範囲, 550
ドキュメントの表示, 544
ノードの削除, 532
ノードの挿入, 532
ノードの追加, 532
マークアップタグの表示, 530
マークアップの表示, 541, 544
メインウィンドウ, 544
主要な機能の使用方法, 553
新たな XML ファイルを開く, 1294
切り替え, 1360
属性値の挿入, 536
特殊文字, 535
入力ヘルパー, 530, 546
要素のクリア, 532
要素の適用, 532
Authentic View テンプレート, 529
Authentic XML, 526
Authentic スクリプト,
セキュリティー設定, 1303
信頼された場所, 1303
Authentic スクリプトの信頼された場所, 1303
Authentic ビュー,
XML 列の編集, 299
Authentic メニュー, 1294
マークアップの表示, 541
動的なテーブルの編集, 541
AuthenticDataTransfer,
dropEffect, 1602
getData, 1602
ownDrag, 1602
type, 1602
AuthenticRange,
AppendRow, 1607
Application, 1608
CanPerformAction, 1608
CanPerformActionWith, 1609
Close, 1609
CollapsToBegin, 1609
CollapsToEnd, 1610
Copy, 1610

AuthenticRange,
Cut, 1610
Delete, 1610
DeleteRow, 1611
DuplicateRow, 1611
ExpandTo, 1612
FirstTextPosition, 1613
FirstXMLData, 1613
FirstXMLDataOffset, 1614
GetElementAttributeNames, 1615
GetElementAttributeValue, 1616
GetElementHierarchy, 1616
GetEntityNames, 1617
Goto, 1618
GotoNext, 1618
GotoNextCursorPosition, 1619
GotoPrevious, 1619
GotoPreviousCursorPosition, 1620
HasElementAttribute, 1620
InsertEntity, 1620
InsertRow, 1621
IsCopyEnabled, 1622
IsCutEnabled, 1622
IsDeleteEnabled, 1622
IsEmpty, 1622
IsEqual, 1623
IsFirstRow, 1623
IsInDynamicTable, 1623
IsLastRow, 1623
IsPasteEnabled, 1624
IsTextStateApplied, 1624
LastTextPosition, 1624
LastXMLData, 1625
LastXMLDataOffset, 1626
MoveBegin, 1627
MoveEnd, 1627
MoveRowDown, 1628
MoveRowUp, 1628
Parent, 1628
Paste, 1628
PerformAction, 1629
Select, 1630
SelectNext, 1630
SelectPrevious, 1631
SetElementAttributeValue, 1632
SetFromRange, 1633
Text, 1633
AuthenticView, 1651
Application, 1643
AsXMLString, 1644
DocumentBegin, 1645
DocumentEnd, 1645
Event, 1646
Goto, 1647
IsRedoEnabled, 1648
IsUndoEnabled, 1648
MarkupVisibility, 1649
OnBeforeCopy, 1635
OnBeforeCut, 1635
OnBeforeDelete, 1636
OnBeforeDrop, 1636
OnBeforePaste, 1637
OnDragOver, 1638
OnKeyboardEvent, 1639
OnMouseEvent, 1640
OnSelectionChanged, 1641
Parent, 1649
Print, 1649
Redo, 1650
Selection, 1650
Undo, 1651
WholeDocument, 1652
XMLDataRoot, 1652
Avro,
JSON 内のデータ構造, 680
と RaptorXML, 675
バイナリ内のデータ構造, 681
ファイルの型, 675
概要, 675
Avro スキーマ, 587
説明, 678
用語, 678
Avro バイナリファイル, 681
Avro ビュー, 681

B

build.xml,
enabling the zip64mode in,, 1107

C

C#,
code, 1017
code generation settings, 1108
reference to generated classes, 1077
XMLSpy の統合, 1913
C++,
code, 1017
code generation settings, 1108
reference to generated classes, 1062
CDATA,
グリッドビューにて子として追加, 1215
グリッドビューにて挿入, 1200
グリッドビューにて追加, 1208
グリッドビューにて変換, 1221
CDATA セクション,
Authentic View 内に挿入, 553
Class,
JAVA, 1857
Class ID,
XMLSpy 統合, 1917
ClassPath statement, 1857
Code,
built in types, 1122
SPL, 1110
Code Generator, 1017
CodeGeneratorDlg,
Application, 1653
CPPSettings_DOMType, 1654
CPPSettings_LibraryType, 1655
CPPSettings_UseMFC, 1655
CSharpSettings_ProjectType, 1655
OutputPath, 1656
OutputPathDialogAction, 1656
OutputResultDialogAction, 1656
Parent, 1657
ProgrammingLanguage, 1657
PropertySheetDialogAction, 1657
TemplateFileName, 1658
COM API,
in Scripting Editor, 1514
COM-API,
documentation, 1544
Configure,
XMLSPY UI, 1531
CoreCatalog.xml, 410
CR&LF, 1455
CSS, 579
ドキュメントアウトライン, 582
プロパティ, 582
情報ウィンドウ, 582
文法色分け, 582
CSS 情報ウィンドウ, 582
CSV ファイル,
XML としてインポート, 1331
CustomCatalog.xml, 410

D

Database connection,
setup examples, 877
DatabaseConnection,
ADOConnection, 1659
AsAttributes, 1659
CreateMissingTables, 1660
CreateNew, 1660
DatabaseKind, 1660
ExcludeKeys, 1661
File, 1661
IncludeEmptyElements, 1662
NumberDateTimeFormat, 1662
ODBCConnection, 1662
SQLSelect, 1663
TextFieldLen, 1664
DatabaseSpy, 344
3D チャート, 355
チャート X-軸, 350
チャート Y-軸, 353
チャート Z-軸, 354
チャートグリッド, 350, 353, 354
チャートサイズ, 355
チャートのタイトル, 342
チャートの色, 348
チャートの背景, 342
チャートの凡例, 342
チャートフォント, 357
チャートフォントオプション, 357
DB, 565, 566
Authentic View 内のクエリ, 565
Authentic View 内の編集, 565, 570

DB, 565, 566
Authentic View内のテーブルのナビゲート, 565
DB クエリ内のパラメーター, 566
クエリの作成, 566
内の Authentic Viewの表示のフィルター, 566
DB XML,
IBM DB2 にて XML スキーマを割り当て, 1322
IBM DB2 に対して、XML スキーマの割り当て, 1318
db2-fn:sqlquery, 462
db2-fn:xmlcolumn, 462
defaultOpenContent, 169
Delete,
Application.URLDelete, 1598
DF,
XMLSpy における変換, 431
Dialogs,
Application, 1665
CodeGeneratorDlg, 1665
DTDSchemaGeneratorDlg, 1666
FileSelectionDlg, 1665
GenerateSampleXMLDlg, 1666
Parent, 1665
SchemaDocumentationDlg, 1666
directories,
creating with Application.URLMakeDirectory, 1598
DocEditEvent (obsolete),
altKey (obsolete), 1807
altLeft (obsolete), 1808
button (obsolete), 1809
cancelBubble (obsolete), 1810
clientX (obsolete), 1811
clientY (obsolete), 1811
ctrlKey (obsolete), 1812
ctrlLeft (obsolete), 1813
dataTransfer (obsolete), 1813
fromElement (obsolete), 1814
keyCode (obsolete), 1815
propertyName (obsolete), 1815
repeat (obsolete), 1816
returnValue (obsolete), 1816
shiftKey (obsolete), 1816
shiftLeft (obsolete), 1817
srcElement (obsolete), 1818
type (obsolete), 1818
DocEditView (obsolete),
ApplyTextState (obsolete), 1822
CurrentSelection (obsolete), 1823
EditClear (obsolete), 1824
EditCopy (obsolete), 1824
EditCut (obsolete), 1825
EditPaste (obsolete), 1825
EditRedo (obsolete), 1826
EditSelectAll (obsolete), 1826
EditUndo (obsolete), 1826
event (obsolete), 1827
GetAllowedElements (obsolete), 1827
GetNextVisible (obsolete), 1829
GetPreviousVisible (obsolete), 1830
IsEditClearEnabled (obsolete), 1830
IsEditCopyEnabled (obsolete), 1831
IsEditCutEnabled (obsolete), 1831
IsEditPasteEnabled (obsolete), 1832
IsEditRedoEnabled (obsolete), 1832
IsEditUndoEnabled (obsolete), 1833
IsRowAppendEnabled (obsolete), 1833
IsRowDeleteEnabled (obsolete), 1834
IsRowDuplicateEnabled (obsolete), 1834
IsRowInsertEnabled (obsolete), 1835
IsRowMoveDOwnEnabled (obsolete), 1835
IsRowMoveUpEnabled (obsolete), 1836
IsTextStateApplied (obsolete), 1836
IsTextStateEnabled (obsolete), 1837
LoadXML (obsolete), 1837
RowAppend (obsolete), 1838
RowDelete (obsolete), 1839
RowDuplicate (obsolete), 1839
RowInsert (obsolete), 1840
RowMoveDown (obsolete), 1840
RowMoveUp (obsolete), 1841
SaveXML (obsolete), 1841
SelectionMoveTabOrder (obsolete), 1842
SelectionSet (obsolete), 1842
XMLRoot (obsolete), 1843
DOCTYPE 宣言,
グリッドビューにて子として追加, 1218
グリッドビューにて追加, 1203, 1211
グリッドビューにて変換, 1222
Document, 1679
Application, 1672
AssignDTD, 1672
AssignSchema, 1673
AssignXSL, 1673
AssignXSLFO, 1673
AuthenticView, 1674
Close, 1675
ConvertDTDOrSchema, 1675

(C) 2015-2021 Altova GmbH

Document, 1679
CreateChild, 1677
CreateSchemaDiagram, 1677
CurrentViewMode, 1678
DataRoot, 1678
DocEditView, 1678
Encoding, 1679
EndChanges, 1679
ExecuteXQuery, 1679
ExportToDatabase, 1680
ExportToText, 1681
FullName, 1682
GenerateDTDOrSchema, 1683
GenerateProgramCode, 1684
GenerateSampleXML, 1684
GenerateSchemaDocumentation, 1684
GetExportElementList, 1686
GetPathName, 1687
GridView, 1687
IsModified, 1687
IsValid, 1688
IsWellFormed, 1690
Name, 1690
OnBeforeCloseDocument, 1670
OnBeforeSaveDocument, 1670
OnBeforeValidate, 1671
OnCloseDocument, 1671
OnViewActivation, 1672
Path, 1691
RootElement, 1691
Save, 1691
SaveAs, 1692
Saved, 1692
SaveInString, 1692
SaveToURL, 1693
SetActiveDocument, 1693
SetEncoding, 1693
SetExternalIsValid, 1694
SetPathName, 1694
StartChanges, 1695
SwitchViewMode, 1695
Title, 1696
TransformXSL, 1696
TransformXSLFO, 1696
UpdateViews, 1697
UpdateXMLData, 1697
XQuery, 1679
Documents,
Count, 1699
Item, 1699
NewAuthenticFile, 1699
NewFile, 1699
NewFileFromText, 1700
OpenAuthenticFile, 1700
OpenFile, 1700
OpenURL, 1701
OpenURLDialog, 1701
DTD, 383, 1455, 1458
ATTLIST 宣言, 1204, 1212, 1219, 1222
ELEMENT 宣言, 1204, 1212, 1218, 1222
ENTITY 宣言, 1204, 1212, 1219, 1222
NOTATION 宣言, 1205, 1213, 1219, 1222
UML へ変換, 1250
XML から外部 DTD への参照, 1203
XML スキーマから生成(Enterprise ならびに Professional エディションのみ), 386
XML スキーマに変換する, 1247
XML スキーマへの変換(Enterprise と Professional エディション), 384
XML ドキュメントから移動, 1244
XML ドキュメントから生成, 1245
XML ドキュメントから定義へ移動, 1245
XML ドキュメントへ割り当て, 1243, 1244
XML ファイルから生成する, 384
XML ファイルのアウトラインを生成する, 1251
エンティティの読み込み, 1244
からコードの生成, 1255
グリッドビュー内の編集 (Enterprise と Professional エディション), 384
テキストビュー内での編集, 384
関連するメニューコマンド, 1243
DTD とカタログ, 409
DTD における ELEMENT 宣言,
グリッドビューにて挿入, 1204
DTD/スキーマメニュー, 1243
DTDSchemaGeneratorDlg,
Application, 1702
AttributeTypeDefinition, 1703
DTDSchemaFormat, 1703
FrequentElements, 1703
GlobalAttributes, 1704
MaxEnumLength, 1704
MergeAllEqualNamed, 1704
OnlyStringEnums, 1704
OutputPath, 1705
OutputPathDialogAction, 1705
Parent, 1705

DTDSchemaGeneratorDlg,
ResolveEntities, 1705
TypeDetection, 1706
ValueList, 1706

E

Eclipse における XMLSpy パースペクティブ, 1016
Eclipse のための XMLSpy プラグイン,
インストール, 1011
Eclipse プラットフォーム,
Eclipse における, 1013
XMLSpy デバッガーパースペクティブ, 1016
と XMLSpy, 1010
と XMLSpy Integration Package, 1011
Eclipse 内部で XMLSpy を使用する, 1010
EDGAR 検証, 1410
ELEMENT 宣言,
グリッドビューにて子として追加, 1218
グリッドビューにて追加, 1212
グリッドビューにて変換, 1222
ElementFormDefault,
スキーマデザインビューにおける設定, 1256
ElementList,
Count, 1707
Item, 1707
RemoveElement, 1707
ElementListItem,
ElementKind, 1707
FieldCount, 1708
Name, 1708
RecordCount, 1708
Enter キー, 576
Enter キー参照, 576
使用の効果, 576
ENTITY 宣言,
グリッドビューにて子として追加, 1219
グリッドビューにて挿入, 1204
グリッドビューにて追加, 1212
グリッドビューにて変換, 1222
Enumerations,
SPYAttributeTypeDefinition, 1844
SPYAuthenticActions, 1844
SPYAuthenticDocumentPosition, 1844
SpyAuthenticElementActions, 1845
SPYAuthenticElementKind, 1845
SPYAuthenticMarkupVisibility, 1845
SPYDatabaseKind, 1846
SPYDialogAction, 1846
SPYDOMType, 1847
SPYDTDSchemaFormat, 1847
SPYEncodingByteOrder, 1847
SPYExportNamespace, 1847
SPYFrequentElements, 1848
SPYKeyEvent, 1849
SPYLibType, 1849
SPYLoading, 1849
SPYMouseEvent, 1850
SPYNumberDateTimeFormat, 1850
SPYProgrammingLanguage, 1851
SPYProjectItemTypes, 1851
SPYProjectType, 1851
SPYSampleXMLGenerationOptimization, 1852
SPYSampleXMLGenerationSchemaOrDTDAssignment, 1852
SPYSchemaDefKind, 1853
SPYSchemaDocumentationFormat, 1853
SPYTextDelimiters, 1854
SPYTextEnclosing, 1854
SPYTypeDetection, 1855
SPYURLTapes, 1855
SPYViewModes, 1856
SPYVirtualKeyMask, 1856
SPYXMLDataKind, 1857
Environment variables,
ANT_OPS, 1107
EPUB ファイル, 842
Event, 1581, 1582, 1583, 1635, 1636, 1637, 1638, 1639, 1640, 1641, 1670, 1671, 1672, 1730, 1731, 1732
Events, 1551
Excel 2007, 257, 834
ExportSettings,
CreateKeys, 1709
ElementList, 1709
EntitiesToText, 1709
ExportAllElements, 1709
FromAttributes, 1710
FromSingleSubElements, 1710
FromTextValues, 1710
IndependentPrimaryKey, 1710
Namespace, 1710
SubLevelLimit, 1711
External ID 宣言,
グリッドビューにて子として追加, 1218

(C) 2015-2021 Altova GmbH

External ID 宣言,
グリッドビューにて挿入, 1203
グリッドビューにて追加, 1211
グリッドビューにて変換, 1222

F

FileSelectionDlg,
Application, 1711
DialogAction, 1711
FullName, 1712
Parent, 1712
Firebird,
ODBC を使用しての接続, 879
FO に対するXSLT スタイルシート,
XML ドキュメントへ割り当て, 1288
FOP,
フォント, 308

G

Generate,
code from schema, 1017
Generate Sample XML, 1844, 1852
GenerateSampleXMLDlg,
Application, 1725
FillWithSampleData, 1726
NonMandatoryAttributes, 1727
NonMandatoryElements, 1727
Parent, 1728
RepeatCount, 1728
TakeFirstChoice, 1729
Global objects,
in SPL, 1114
GridView,
CurrentFocus, 1732
Deselect, 1733
IsVisible, 1733
OnBeforeDrag, 1730
OnBeforeDrop, 1731
OnBeforeStartEditing, 1731
OnEditingFinished, 1732
OnFocusChanged, 1732
Select, 1733
SetFocus, 1733
GUI 詳細, 108

H

HTML, 579
XMLSpy の統合, 1919
HTML サンプル,
of XMLSpyControl 統合, 1916
XMLSpyControl 統合, 1917, 1918
HTML ドキュメント,
情報ウィンドウ, 580
編集, 580
HTML 出力,
PXF ファイルからAuthentic View 内に生成, 1303
HTML 情報 ウィンドウ, 580
HTTP, 711
HTTP メッセージウィンドウ,
HTTP メソッド, 712
HTTP 状態コード, 718
と WADL, 716
ヘッダーの受け入れ, 718
リクエストのインポート, 716
リクエストの送信, 712
レスポンスの受け入れ, 718

I

IBM DB2,
JDBC を介した接続, 881
ODBC を使用しての接続, 883
XML スキーマの割り当て, 1318
XML スキーマを XML ファイルへ割り当て, 1322
スキーマ管理ならびに割り当て, 1318
IBM DB2 for i,
ODBC を使用しての接続, 891
IBM Informix,
connecting through JDBC, 890
JDBC を使用しての接続, 894
Insert コマンド,
グリッドビューにて, 1198

J

Java, 1923
API, 1857
avoiding exceptions in generated code, 1107
ClassPath, 1857
code, 1017
reference to generated classes, 1092
Java 仮想マシン,
パス設定, 1483
Java 拡張関数,
XSLT と XQuery, 2065
インスタンスメソッド、インスタンスフィールド, 2071
コンストラクター, 2070
データ型変換、 Java から XPath/XQuery へ, 2073
データ型変換、XPath/XQuery から Java へ, 2072
ユーザー定義の JAR ファイル, 2069
ユーザー定義のクラスファイル, 2066
概要, 2065
静的メソッド、静的フィールド, 2070
Java 設定, 1483
JDBC,
Teradata への接続, 926
データ接続インターフェイスとして, 847
接続の設定 (Windows), 868
JRE,
Eclipse のための XMLSpy プラグイン, 1011
JScript,
scripting with XMLSpy, 1505
JSON, 579
JSON インスタンスを XML インスタンスに変換, 1352
XML スキーマ を JSON スキーマに変換する(JSON スキーマから XML スキーマに変換する), 1355
XQuery 式, 668
と XPath, 666
と XPath/XQuery 出力ウィンドウ, 666
と XQuery, 666
JSON グリッドビュー, 600
XQuery 式, 617, 618
イメージの表示, 622
オブジェクト内でコンポーネントをラップ, 1240
コンテキストメニューコマンド, 625
コンポーネントの挿入, 1240
コンポーネントの追加, 1240
セル内のコマンドボタン, 606, 607
チャートの作成, 623
テーブルアイコン, 613
テーブルの行と列をフリップする, 1242
テーブルの表示, 613
テーブルの表示コマンド, 1242
テーブル内の列を並べ替える, 1242
ドキュメントの表示, 603
ドキュメントの編集, 607
ドキュメント構造の編集, 606
ドラッグアンドドロップ, 610
ビューのフィルター, 617
メタデータファイル, 617, 618
異なる型のシンボル, 603
型コマンド, 1239
子としてコンポーネントを追加する, 1240
自動補完, 607
出力を計算するためのフォーミュラ, 617, 618
配列内でコンポーネントをラップ, 1240
JSON グリッドビュー内でのドラッグとドロップ, 610
JSON グリッドビュー内の XQuery 式, 617
JSON グリッドビュー内のイメージ, 622
JSON グリッドビュー内のチャート, 623
JSON グリッドビュー内のテーブル,
TSV または JSON としてコピー, 627
JSON グリッドビュー内のテーブル表示,
スプレッドシートのインポート/エクスポート, 613
編集, 613
JSON グリッドビュー内のフィルター, 617
の設定, 627
JSON グリッドビュー内のフォーミュラ, 617, 618
の設定, 627
JSON スキーマ, 587, 594
allOf, 655
any, 652
anyOf, 655
if-then-else, 656
JSON インスタンスからの生成, 670
multiple, 652
not, 655
oneOf, 655
アトミック方, 651
オブジェクト, 640, 642
オブジェクトと依存性, 646
オブジェクトプロパティ, 640
演算子, 655
基本型, 651
型セレクター(any, multiple), 652
指定されていないプロパティ, 642
条件, 656

(C) 2015-2021 Altova GmbH

JSON スキーマ, 587, 594
数値定義, 651
説明, 592
単純型, 651
配列, 649
文字列定義, 651
用語, 592
JSON スキーマバージョン, 631
JSON スキーマビュー, 630
デザインビュー, 638
外部スキーマの追加, 634
構成, 657
詳細入力ヘルパー, 634
制約入力ヘルパー, 634
入力ヘルパー, 634
JSON データ,
オブジェクト, 589
サンプル, 589
型, 589
配列, 589
JSON ドキュメント,
XML から、または、へ変換, 674
XML からの変換, 595
XMLSpy 内で開く, 587
XQuery式のための, 522
グリッドビュー内での編集, 600
テキストビューでの編集, 595
検証, 662
新規作成, 587
JSON ドキュメントのグリッドビュー,
JSON グリッドビューを参照, 600
JSON ドキュメントの構文チェック, 595, 600
JSON ドキュメントの整形式チェック, 595
JSON ドキュメントの整形式のチェック, 600
JSON のための XQuery 式, 668
JSON メニュー, 1239
JSON ライン, 594

L

Library, 1123
loading, 1701

M

Macros,
developing, 1505, 1510
enabling, 1516, 1528
running, 1529
MapForce, 1251
MariaDB,
ODBC を使用した接続, 896
Method names in generating code,
reserving, 1107
Microsoft Access,
ADO を介しての接続, 852, 898
Microsoft Office 2007, 257, 834
Microsoft SQL Server,
ADO を介しての接続, 901
ODBC を使用しての接続, 903
Microsoft® SharePoint® Server, 1189
MIME, 1458
MS SQL Server,
スキーマ拡張, 1270
スキーマ設定, 1271
MSXML, 1479
generating code for, 1108
MSXSL スクリプト内の拡張関数, 2078
msxsl:script, 2078
MySQL,
ODBC を使用しての接続, 909

N

NOTATION 宣言,
グリッドビューにて子として追加, 1219
グリッドビューにて挿入, 1205
グリッドビューにて追加, 1213
グリッドビューにて変換, 1222

O

OASIS,
XML カタログ, 1227
ODBC,

ODBC,
MariaDB への接続, 896
Teradata への接続, 928
データ接続インターフェイスとして, 847
接続のセットアップ, 865
ODBC ドライバー,
使用可能か確認する, 865
Office Open XML, 257, 834
OLE DB,
データ接続インターフェイスとして, 847
OOXML,
Office Open XML を参照, 257
Office Open XML 以下を参照, 834
Open Office XML,
Archive View にて作成, 836
Archive View にて編集, 836
サンプルファイル, 838
openContent, 169
Oracle,
スキーマ拡張, 1269
スキーマ設定, 1270
Oracle XML Db, 1326
データベースの参照, 1329
Oracle XML データベース,
XML スキーマの管理, 1326
Oracle データベース,
JDBC を使用しての接続, 911
ODBC を使用しての接続, 912
OS,
Altova 製品のための, 2094
Out of memory exceptions,
resolving, 1107
Overview,
of XMLSpy API, 1545

P

Parent, 1690
PDF フォント, 308
PDF 出力,
PXF ファイルからAuthentic View 内に生成, 1303
Plug-in,
ATL sample files, 1534
registration, 1530
User interface configuration, 1531
XMLSPY, 1530
PostgreSQL,
ODBC を使用しての接続, 918
直接 (ネイティブ)接続, 870
直接接続する(ネイティブ), 872
PowerPoint 2007, 257, 834
Progress OpenEdge データベース,
JDBC を使用しての接続, 921
ODBC を使用しての接続, 922
PUBLIC,
識別子 - カタログ, 1227
PXF ファイル,
Authentic View から出力ドキュメントを生成, 1303

R

Register,
plug-in, 1530
Return キー,
Enter キーを参照, 576
RootCatalog.xml, 410
RTF 出力,
PXF ファイルからAuthentic View 内に生成, 1303

S

save, 1693
schema, 1591
code generator, 1017
SchemaAgent, 406
XMLSpy からサーバーへ接続, 1272
XMLSpy からスキーマを開く, 406
サーバーから切断, 1273
スキーマの表示, 1273
拡張検証, 1273
作業する, 400, 401, 406
SchemaAgent Server,
接続, 398
SchemaAgent Server へ接続, 398, 1272
SchemaDocumentationDlg,
AllDetails, 1734
Application, 1735
IncludeAll, 1737
IncludeAttributeGroups, 1737
IncludeComplexTypes, 1737

(C) 2015-2021 Altova GmbH

SchemaDocumentationDlg,
IncludeGlobalElements, 1738
IncludeGroups, 1738
IncludeIndex, 1738
IncludeLocalElements, 1739
IncludeRedefines, 1739
IncludeSimpleTypes, 1740
OptionsDialogAction, 1740
OutputFile, 1741
OutputFileDialogAction, 1741
OutputFormat, 1741
Parent, 1742
ShowAnnotations, 1742
ShowAttributes, 1742
ShowChildren, 1742
ShowConstraints, 1743
ShowDiagram, 1743
ShowEnumerations, 1743
ShowNamespace, 1744
ShowPatterns, 1744
ShowProgressBar, 1744
ShowProperties, 1745
ShowResult, 1745
ShowSingleFacets, 1745
ShowSourceCode, 1745
ShowType, 1746
ShowUsedBy, 1746
schemanativetype, 1111
Scripting Editor,
overview, 1505, 1507
SharePoint® Server, 1189
SOAP, 683, 695, 1382
WSDL からリクエストを送信, 691
サーバーにリクエストを送信する, 1382
デバッガーオプション, 1389
デバッグセッション, 1387
プロキシサーバーの開始, 1388
プロキシサーバーの停止, 1389
リクエスト, 1382, 1389
リクエストパラメーター, 1383
新規リクエストの作成, 1380
SOAP コミュニケーション処理, 698
SOAP デバッガー, 696
Visual Studio .NET 内の SOAP デバッガー, 1009
ブレークポイントのセット, 707
SOAP メニュー, 1380
SOAP 検証, 695
Source control,
installing a source-control plug-in, 980
SPL, 1110
code blocks, 1110
conditions, 1117
foreach, 1118
global objects, 1114
subroutines, 1119
using files, 1115
variables, 1113
SPP ファイルの位置, 1168
SPS,
新たな XML ファイルに関連付ける, 1128
SPS テーブル,
動的なテーブルの編集, 541
SPS テーブル 内の Authentic View,
使用方法, 557
SPS ファイル,
XML ファイルに割り当てる, 1296
SpyProject,
CloseProject, 1747
ProjectFile, 1748
RootItems, 1748
SaveProject, 1748
SaveProjectAs, 1748
SpyProjectItem,
ChildItems, 1749
FileExtensions, 1749
ItemType, 1749
Name, 1749
Open, 1749
ParentItem, 1750
Path, 1750
ValidateWith, 1750
XMLForXSLTransformation, 1750
XSLForXMLTransformation, 1750
XSLTransformationFileExtension, 1750
XSLTransformationFolder, 1751
SpyProjectItems,
AddFile, 1751
AddFolder, 1751
AddURL, 1752
Count, 1752
Item, 1752
RemoveItem, 1752
SQL Server,
ADO.NET を使用して接続する, 859
XML スキーマの管理, 1324
SQL エディター,

SQL エディター,
クエリの作成, 1315
その説明, 1313
データベースクエリウィンドウにて, 1313
SQL サーバー,
ADO を介しての接続, 852
SQL スクリプトにおけるブックマーク, 1313
SQL スクリプトにおける自動補完, 1313
SQL スクリプト内のコマンド, 1313
SQL スクリプト内の領域, 1313
SQLite,
外部キーの無効化, 875
接続のセットアップ (Windows), 875
StyleVision, 1251
StyleVision Power Stylesheetの編集, 1296
と XBRL, 1403
StyleVision Power Stylesheet,
StyleVision内の編集, 1296
XML ファイルへ割り当てる, 1296
StyleVision Power Stylesheet を XML ファイルに割り当てる, 1296
Sybase,
JDBC を使用しての接続, 925

T

Teradata,
JDBC を使用した接続, 926, 928
terminate, 1595
TextImportExportSettings,
DestinationFolder, 1753
EnclosingCharacter, 1753
Encoding, 1754
EncodingByteOrder, 1754
FieldDelimiter, 1754
FileExtension, 1754
HeaderRow, 1754
ImportFile, 1754
TSV,
JSON グリッドビューテーブルを以下としてコピー, 627
XML グリッドビューテーブルを以下としてコピー, 296
Types,
built in, 1122

U

UCS-2, 1460
UML,
スキーマから変換, 1250
Unicode のサポート,
Altova 製品内で, 2094
URL, 1598, 1693, 1701
メールで送信, 1143
User interface,
configure using plug-in, 1531
UTF-16, 1460

V

Variables,
in SPL, 1113
VBScript,
scripting with XMLSpy, 1505
Visual Studio,
generating code for, 1108
XMLSpy ActiveX コントロールをツールボックスへ追加, 1906
Visual Studio .Net,
と XMLSpy, 1005
と XMLSpy デバッガー, 1009
と XMLSpy の違い, 1007
VS .NET,
と XMLSpy 統合パッケージ, 1006
VS .NET に対応した XMLSpy プラグイン,
インストール, 1006

W

WADL,
HTTP リクエストのために使用, 716
Web サーバー, 1501
Windows,
Altova 製品のためのサポート, 2094
Word 2007, 257, 834
Word 2007+ 出力,
PXF ファイルからAuthentic View 内に生成, 1303

(C) 2015-2021 Altova GmbH

Word ドキュメント,
XML としてインポート, 1339
Wrapper classes,
in generated code, 1108
WSDL, 683, 690
1.1 から 2.0 への変換, 1379
1.1 コンポーネント, 1367
1.1 におけるオペレーション, 1368
1.1 におけるサービス, 1370
1.1 におけるバインディング, 1369
1.1 におけるポート型, 1369
1.1 におけるメッセージ, 1367
2.0 コンポーネント, 1370
2.0 におけるインターフェイス, 1371
2.0 におけるサービス, 1373
2.0 におけるバインディング, 1372
PortTypes の作成, 685
SOAP デバッガー, 696
SOAP リクエストの送信, 691
ウェブサービス, 1380
ウェブサービスへ接続, 690
オペレーションの作成, 685
サービスの作成, 688
スキーマを開く, 690
ドキュメンテーションの作成, 692
ドキュメンテーションの生成, 1375
ドキュメントの再解析, 1378
バインディングの作成, 687
パラメーターの作成, 685
ポートの作成, 688
メッセージの作成, 685
型, 1374
検証, 689
新規ドキュメントの作成, 684
名前空間, 684
WSDL チュートリアル, 684
WSDL デザインビュー,
WSDL の記述, 231
サービス, 232
バインディング, 232
ファイルの閲覧, 231
ポートタイプ, 232
メインウィンドウ, 232
機能, 231
WSDL ビュー,
WSDL ドキュメントにインポート, 236
概要入力ヘルパー, 236
詳細入力ヘルパー, 236
入力ヘルパー, 236
WSDL ファイル,
拡張性要素, 1232
検証, 1232
WSDL フォント, 1477
WSDL メニュー, 1367

X

XBRL, 721
XULE, 821
アークロール, 1391
ターゲット名前空間, 1395
と MapForce, 1403
と StyleVision, 1403
ドキュメンテーションの生成, 1398
リンクロール, 1393
名前空間, 1394
XBRL タクソノミ, 738
タクソノミも参照, 739
XBRL タクソノミ マネジャー, 722
XBRL タクソノミにおけるリンクロール, 756
XBRL テーブル定義エディター, 791
XBRL におけるアークロール, 1391
XBRL におけるターゲット名前空間, 1395
XBRL におけるタクソノミ, 738, 739
XBRL におけるリンクロール, 1393
XBRL における検索,
XBRL 内を検索を参照, 829
XBRL における名前空間, 1394
XBRL ビュー, 243
メインウィンドウ内の プレゼンテーション タブ, 247
メインウィンドウ内の 計算 タブ, 247
メインウィンドウ内の 定義 タブ, 247
メインウィンドウ内の要素, 243
設定, 1401
入力ヘルパー, 250
XBRL ビュー、検索,
XBRL 内を検索を参照, 829
XBRL フォーミュラエディター, 767
XBRL フォント, 1478
XBRL メニュー, 1391
XBRL 検証オプション, 1485
XBRL 内を検索, 829
ウィンドウ, 833
結果, 833

XBRL 内を検索, 829
  検索語, 829
XBRL 内を検索ウィンドウ, 117
XBRLのための XULE, 821
Xerces,
  generating code for, 1108
XInclude,
  XML ドキュメントの挿入, 266
  グリッドビューにて子として追加, 1216
  グリッドビューにて挿入, 1201
  グリッドビューにて追加, 1154, 1209
  テキストビューにて挿入, 1154
XML,
  Oasis カタログ, 1227
  スペルチェッカー, 1414
  構造化テキストとしてコピー, 1150
XML DB,
  新たにデータ行を Authentic View へロード, 1297
  新規 XML データ行のロード, 565
XML DB 内のデータを編集, 1297
XML インポート,
  スキーマをベースに, 1343
XML グリッドビュー,
  グリッドビューを参照, 268
XML グリッドビュー内でのドラッグとドロップ, 278
XML グリッドビュー内の XQuery 式, 285, 286
XML グリッドビュー内のテーブル,
  TSV または XML としてコピー, 296
XML グリッドビュー内のテーブル表示,
  スプレッドシートのインポート/エクスポート, 281
  編集, 281
XML グリッドビュー内のフィルター, 285
  の設定, 296
XML スキーマ, 46, 383
  <alternative> 要素, 189
  DB XML へ割り当て, 1322
  DTD, 264
  DTD スキーマの生成(Enterprise と Professional エディション), 384
  DTD に変換する, 1247
  IBM DB2 データベースにおける管理と割り当て, 1318
  IBM DB2 にて管理, 1318
  MS SQL Server スキーマ設定, 1271
  MS SQL Server 拡張, 1270
  Oracle スキーマ設定, 1270
  Oracle 拡張, 1269
  XML スキーマへの変換(Enterprise ならびに Professional エディションのみ), 386
  XML ドキュメントから移動, 1245
  XML ドキュメントから生成, 1245
  XML ドキュメントから定義へ移動, 1245
  XML ドキュメントの生成, 386
  XML ドキュメントを編集中に修正する, 98
  XML ファイルのアウトラインを生成する, 1251
  インクルードされたスキーマのフラット化, 387
  インクルードされたスキーマをフラット化, 1275
  からコードの生成, 1255
  グリッドビューで編集(Enterprise ならびに Professional エディションのみ), 386
  グローバル コンポーネント, 169
  グローバルリソース, 264
  コンテンツ モデルの編集, 176
  コンテンツモデルダイアグラム, 1259
  コンテンツモデルビューの設定, 1264
  コンポーネンツの追加, 49
  サブセットへ分割, 1274
  スキーマデザインビューにおける設定, 1256
  スキーマデザインビューにおける名前空間の設定, 1256
  スキーマも参照, 1243
  スマートフィックス, 219
  チュートリアル, 46
  テキストビューで編集, 386
  デザインビューでのナビゲーション, 73
  ドキュメンテーションの生成, 1259
  ビューの構成, 55
  関連するメニューコマンド, 1243
  基本的なスキーマの作成, 46
  検証, 219
  新しいファイルの作成, 46
  他のスキーマを含める, 387
  名前空間の定義, 48
  要素を追加する, 54
XML スキーマのスマートフィックス, 219
XML スキーマの検証, 219
XML スキーマ定義,
  高度な機能, 61
XML データ,
  データベースへエクスポート, 1350
  テキストファイルへエクスポート, 1347
XML データをデータベースへエクスポート, 1350
XML データをテキストファイルへエクスポート, 1347
XML テーブル 内の Authentic View,
  編集のためのアイコン, 562
XML テーブル内の Authentic View,
  使用方法, 558
XML テキスト, 1461
  コピー, 1150

(C) 2015-2021 Altova GmbH

XML テキストのコメントアウトとアンコメント, 1167
XML ドキュメント, 261
Authentic View 内で開く, 529
Authentic ビュー内での編集, 299
DTD から生成, 384
Oracle XML Db の参照, 1329
XInclude の挿入, 266
XML スキーマから生成(Enterprise ならびに Professional エディションのみ), 386
XPath 条件式の評価, 381
XQuery 実行の自動化, 306
XSLT スタイルシートへ割り当て, 1288
XSLT による変換, 306
XSLT 変換の自動化, 306
エンコーディング, 381
グリッドビュー内での編集, 268
スキーマの生成, 381
スキーマを割り当て(DTD を含む), 264
テキストのインポートとエクスポート, 381
テキストビュー内での編集, 84
デフォルトビュー, 262
と XQuery, 306
とテキストビュー内のコメント, 266
とテキストビュー内の空のライン, 266
とノードの XPath 式, 266
のためのテキストビュー編集機能, 266
の妥当性のチェック, 88
ファイルパスの挿入, 266
開く, 262
検索と置き換え, 381
検証, 264
自動検証, 262
新規作成, 79
整形式のチェック, 264
保存, 262
XML ドキュメントで選択されたノードの XPath,
クリップボードにコピー, 1154
XML ドキュメントで選択されたノードの XPointer,
クリップボードにコピー, 1154
XML ドキュメントの検証, 264
XML ドキュメントの整形式をチェック, 264
XML ドキュメント作成,
チュートリアル, 79
XML の差分,
ディレクトリの比較, 1427
ファイルの比較, 1430
XML パーサー,
について, 2094
XML ファイル,
DTD または XML スキーマから生成する, 1251
XML プロローグ,
グリッドビューにて子として追加, 1215
グリッドビューにて挿入, 1200
グリッドビューにて追加, 1208
グリッドビューにて変換, 1221
XML メニュー, 1198
XML 差分, 1423
ファイルの比較, 1423
XML 署名, 370, 574, 1298
検証, 375, 1236
作成, 372, 1234
xml:base, 228
xml:id, 228
xml:lang, 228
xml:space, 228
XMLData,
AppendChild, 1796
EraseAllChildren, 1797
EraseCurrentChild, 1798
GetChild, 1799
GetChildKind, 1800
GetCurrentChild, 1800
GetFirstChild, 1800
GetNextChild, 1801
HasChildren, 1803
HasChildrenKind, 1803
InsertChild, 1803
IsSameNode, 1804
Kind, 1805
MayHaveChildren, 1805
Name, 1805
Parent, 1805
TextValue, 1806
XMLSpy, 1127
plug-in registration, 1530
ヘルプ, 125
機能, 125
統合, 1905
XMLSpy API,
documentation, 1544
overview, 1545
XMLSpy Enterprise Edition,
ユーザーマニュアル, 28
XMLSpy Integration Package, 1011
XMLSPY plug-in, 1530
XMLSpy コマンドテーブル, 1932

XMLSpy でのプロジェクト,
の利点, 103
作成方法, 103
XMLSpy でのプロジェクト管理, 103
XMLSpy における SchemaAgent, 397
XMLSpy におけるデータベース, 846
XMLSpy を SchemaAgentサーバーから切断, 1273
XMLSPY を使用して編集, 1455
XMLSpy 統合,
のサンプル, 1916, 1917, 1918
XMLSpy 統合パッケージ, 1006
XMLSpy 内の XQuery Update, 456
XMLSpy 内のインライン XBRL, 766
XMLSpy 内のカタログ, 410
XMLSpyControl, 1954
C# を使用した統合, 1913
HTML を使用下統合, 1919
アプリケーションレベルでのサンプル 統合, 1916, 1917, 1918
オブジェクトレファレンス, 1951
ドキュメント, 1905
ドキュメントレベルでの統合のサンプルサンプル, 1913
XMLSpyControl ドキュメント, 1962
XMLSpyControl プレースホルダ, 1968
XMLSpyDocumentEditor,
MarkUpView, 1838
XMLSpyLib, 1544, 1545
Application, 1580
AuthenticDataTransfer, 1601
AuthenticRange, 1606
AuthenticSelection (obsolete), 1819
AuthenticView, 1634
CodeGeneratorDlg, 1652
DatabaseConnection, 1658
Dialogs, 1664
DocEditEvent (obsolete), 1806
DocEditView (obsolete), 1821
Document, 1668
Documents, 1698
DTDSchemaGeneratorDlg, 1702
ElementList, 1706
ElementListItem, 1707
ExportSettings, 1708
FileSelectionDlg, 1711
GenerateSampleXMLDlg, 1724
GridView, 1730
ProjectItem, 1748
SchemaDocumentationDlg, 1733
SpyProject, 1747
SpyProjectItems, 1751
TextImportExportSettings, 1753
XMLData, 1795
XMLSpyコマンド,
XMLSpyControl, 1951
XMLSpyControl 内, 1953
XMLグリッドビュー内のフォーミュラ, 286
の設定, 296
XML-準拠, 1458
XPath,
XML ドキュメント内でノードを生成する方法, 266
評価, 113
XPath 1.0,
XPath の評価, 1225
XPath 2.0,
XPath の評価, 1225
XPath ウィンドウ, 113
XPath のコピー, 1154
XPath の評価, 113
使用方法, 1225
XPath/XQuery ウィンドウ, 506, 507
XPath/XQuery デバッガー内のウォッチ式, 514
XPath/XQuery デバッガー内のデバッグポイント, 514
XPath/XQuery デバッガー内のトレースポイント, 514
XPath/XQuery デバッガー内のブレークポイント, 514
XPath/XQuery デバッガー内の変数, 514
XPath/XQuery 式,
JSON ドキュメントのための, 522
XPath/XQuery ウィンドウ, 524
XPath/XQuery ウィンドウ内での作成, 520
デバッグ, 514
評価, 509
評価とデバッグ, 506
XPath/XQuery 式のデバッグ, 514
XPath/XQueryデバッガー内のブレークポイント, 514
XPaths,
トレースポイントの設定, 486
XPath-Watch ウィンドウ,
XSLT/XQuery デバッガー, 478
XPointer,
XML ドキュメント内でノードを生成する方法, 266
XPointer のコピー, 1154
XQuery, 446, 452
DB サポート, 462
IBM DB2 に対する関数, 462
XML データベースの問い合わせ, 462
Xquery ドキュメントに変数をパスする, 1281

XQuery, 446, 452
インテリジェントな編集機能, 448
テキストビューにて編集, 442
ドキュメント検証, 452
ファイルを開く, 444
拡張関数, 2064
実行, 452
入力ヘルパー, 445
XQuery Update,
PUL, 456
プレビュー, 456
XQuery Update Facility, 456
XQuery Update オプション, 1482
XQuery Update 内のPUL, 456
XQuery オプション, 1482
XQuery デバッガー,
XSLT/XQuery デバッガーを参照, 467
XQuery ドキュメント,
実行時間の分析, 492, 1287
XQuery ファイル,
XMLSpy にてファイル拡張子をセット, 444
XQuery プロセッサー,
Altova 製品内で, 2094
XQuery プロファイラー,
結果のチャート, 503
XQuery 更新, 453
XQuery 実行, 453, 1285
XSD 1.0 と 1.1,
編集モード, 161
XSD モード, 161
XSD 検証, 219
xsi:type,
使用方法, 81
XSL,
XSLT を参照, 1288
XSL アウトライン, 433
XSL アウトラインウィンドウ, 434
XSL スピードオプティマイザー, 440, 1279
XSL スピードオプティマイザーオプション, 1481
XSL 概要ウィンドウ, 114
XSL 変換,
XSLT を参照, 99
XSL/XQuery メニュー, 1277
xsl:call-template, 434
XSL:FO,
と XSLT 変換, 431
xsl:param, 434
xsl:with-param, 434
XSLT, 1362
XMLSpy での変更, 101
XMLSpy における機能, 428
XMLSpy を使った変換, 431
テキストビューにおける自動補完, 429
ドキュメント, 429
とバッチ変換, 431
プロセッサ, 1479
拡張関数, 2064
検証, 429
入力ヘルパー, 429
XSLT スタイルシート,
XML ドキュメントの割り当て, 1287
XML ドキュメントへ割り当て, 1288
開く, 1288
実行時間の分析, 492, 1287
XSLT デバッガー,
Visual Studio .NET 内の XSLT デバッガー, 1009
ステップアウト, 1290
ステップイン, 1290
ステップオーバー, 1290
デバッガーを開始, 1289
デバッガーを再起動, 1289
デバッガーを停止, 1289
デバッグウィンドウ, 1293
デバッグセッションを終了, 1290
トレースポイントの挿入/削除, 1291
トレースポイントの有効化/無効化, 1292
ブレークポイント/トレースポイントダイアログボックス, 1292
ブレークポイントの挿入/削除, 1291
ブレークポイントの有効化/無効化, 1291
現在実行中のノードを表示, 1291
設定, 1293
XSLT テンプレート,
XSL アウトラインウィンドウにて, 434
XSLT と XQuery のためのJava 拡張関数,
Java 拡張関数を参照する, 2065
XSLT と XQuery のための拡張関数, 2064
XSLT ドキュメント,
XSL アウトラインを使った編集と管理, 433
と情報ウィンドウ, 437
XSLT ドキュメント構造, 114
XSLT パラメーター,
インターフェイスを使用してスタイルシートにパスする, 1281
XSLT プロセッサー,
Altova 製品内で, 2094
XSLT プロファイラー,

XSLT プロファイラー,
結果のチャート, 503
XSLT へ変換,
XSLT ファイルを割り当てる, 99
XSLT 関数,
XSL アウトラインウィンドウにて, 434
XSLT 変換, 1279
FO への変換, 1280
PDF への変換, 1280
XMLSpy 内で, 100
チュートリアル, 99
XSLT/XQuery デバッガー, 468
XPath-Watch ウィンドウ, 478
インターフェースの記述, 468
キーボードショートカット, 491
コーススタックウィンドウ, 478
コンテキストウィンドウ, 476
セッションの開始, 474
ツールバーアイコン, 469
テンプレートウィンドウ, 480
トレースウィンドウ, 481
トレースポイントウィンドウ, 486
ブレークポイントの使用方法, 483
メカニズムの記述, 468
メッセージウィンドウ, 479
機能と使用方法, 467
情報ウィンドウ, 475, 480
設定, 473
変数ウィンドウ, 477
XSLT/XQuery デバッグ,
使用するファイル, 467
XSLT/XQuery プロファイラー, 492
XSLT/XQuery 内のスクリプト,
拡張関数で参照する, 2064
XSLT/XQueryを使用した JSON 変換, 666
XULE options, 1488
XULE Window, 121
XULE ドキュメント,
XMLSpyプロジェクト, 821
検証, 821
自動補完, 821
自動補完のためのタクソノミ, 821
適合ファイル型, 821
編集サポート, 821
XULE の実行, 1409
XULE 実行,
オプション, 827
方法, 827
XULE 処理, 1409
XULEウィンドウ,
XBRLインターフェイスのインタラクティブなクエリ, 824

Z

ZIP ファイル, 257, 834
Archive View にて作成, 840
Archive View にて編集, 840
zip64mode,
enabling in the build.xml file, 1107
アイコン,
ツールバー/メニューへ追加, 1439
大きく表示, 1451
アクティブな構成,
グローバルリソース, 1434
アサーションメッセージ, 220
アップデートリストの保留 (PUL), 456
イメージ書式設定,
内の Authentic View, 575
インターネット, 1501
インターネットの使用,
Altova 製品内で, 2095
インテリジェントな編集, 1461
インデント,
テキストビューにて, 1157
インデントガイド, 1363
インポート,
MS Word ドキュメントを XML として, 1339
XML スキーマをベースにデータベースデータをインポート, 1343
データベースデータを XML として, 1334
データベース構造を XML スキーマとして, 1339
テキストファイルから XML へ, 1331
参照, 1396
インライン XBRL,
空白スペースの処理, 1485
インライン XBRL 内の空白スペース, 1485
ウィンドウ,
フロート、ドッキング、タブ, 108
プロジェクト, 1493, 1494
開く, 1495
左右に並べて表示, 1493
自動非表示, 108
重ねて表示, 1493
上下に並べて表示, 1493
情報, 1494

(C) 2015-2021 Altova GmbH

ウィンドウ,
入力ヘルパー, 1494
ウィンドウ メニュー, 1493
ウェブサービス,
接続する, 690
エイリアス,
グローバルリソース参照, 934
エクスプローラー, 1458
エリアチャートの機能, 344
ゲージチャートの機能, 344
ローソク足チャートの機能, 344
円グラフの機能, 344
折れ線グラフの機能, 344
棒グラフの機能, 344
エンコーディング,
デフォルト, 1460
ファイル, 1137
エンティティ,
Authentic View に挿入, 537
Authentic View 内に挿入, 553
Authentic View 内の定義, 553
XML スキーマをベースにする XML にて, 264
内の Authentic View の定義, 573
エンティティ入力ヘルパー,
Authentic View, 546
エンドユーザー使用許諾契約書, 2096, 2097
オーバーライド, 169
オープン コンテンツ モデル, 193
オブジェクトロケーター,
データベースクエリウィンドウにて, 1310
オプション,
3D チャート, 355
エリアチャートの機能, 344
ゲージチャートの機能, 344
チャート X-軸, 350
チャート Y-軸, 353
チャート Z-軸, 354
チャートグリッド, 350, 353, 354
チャートサイズ, 355
チャートのタイトル, 342
チャートの色, 348
チャートの背景, 342
チャートの凡例, 342
チャートフォント, 357
ローソク足チャートの機能, 344
円グラフの機能, 344
折れ線グラフの機能, 344
棒グラフの機能, 344
オプションの要素,
にする, 57
お気に入り, 1496
カスタマイズ, 122
コンテキストメニュー, 1446
コンテキストメニューのカスタマイズ, 1451
ツールバー/メニューコマンド, 1439
マクロ, 1448
メニュー, 1446
カスタムカタログ, 1227
カスタム辞書, 1414
カタログ, 409
Oasis XML, 1227
カタログとインテリジェントな編集, 411
カタログと環境変数, 413
カタログのカスタム化, 411
カタログメカニズムの概要, 409
カタログ内で使用される環境変数, 410
キーコード,
Altova ソフトウェアのための, 1498
キーボードショートカット, 1443
キーマップ, 1497
クエリ,
Authentic View内の DB 表示, 566
XQuery を参照, 1305
データベースクエリウィンドウ以下を参照, 1305
クエリデータベース ウィンドウ, 1305
クエリデータベースコマンド, 1305
クエリペイン,
データベースクエリウィンドウにて, 1313
グリッドビュー, 92, 149, 1359, 1361
データの入力, 83
テーブル, 151
テーブルビューに切り替える, 1223
とデータのインポート/エクスポート, 151
とテーブルビュー, 94
入力ヘルパー, 155
入力ヘルパーの使用, 92
編集, 149, 1224
要素と属性の付加, 92
グリッドビュー (JSON),
JSON グリッドビューを参照, 600
グリッドビュー (JSON) テーブル,
TSV または JSON としてコピー, 627
グリッドビュー (JSON) 設定, 627
グリッドビュー (XML), 268
XQuery 式, 285, 286
コンテキストメニュー, 293

グリッドビュー (XML), 268
チャートの作成, 290
テーブル表示アイコン, 281
ドキュメントコンテンツの編集, 275
ドキュメントの表示, 270
ドキュメント構造の編集, 273
ドラッグアンドドロップ, 278
ノードの型を編集する, 275
ノードの追加と削除, 273
メタデータファイル, 285, 286
機能, 270
検索と置換, 275
検証, 275
出力を計算するためのフォーミュラ, 285
内でのイメージ, 289
入力ヘルパー, 273
表示のフィルター, 286
グリッドビュー (XML) テーブル,
TSV または XML としてコピー, 296
グリッドビュー (XML) 設定, 296
グリッドビュー内のイメージ, 289
グリッドビュー内のチャート, 290
グリッドビュー内の行,
挿入, 1223
付加, 1223
グリッドフォント, 1472
グレーのバー, 1361
グローバル,
設定, 1454
グローバル リソース,
定義, 935
グローバル リソース XML ファイル, 935
グローバルコンポーネント,
グローバルコンポーネントの名前変更, 425
グローバルなコメント,
表示される行, 165
グローバルリソース,
アクティブな構成, 1434
ツールバーにて有効にする, 1440
データベースの型の定義, 942
ファイルの型の定義, 937
ファイル型とフォルダー型の使用, 944
フォルダーの型の定義, 941
構成の変更, 948
使用, 944, 947, 948
定義, 1433
グローバルリソースの,
構成, 935
グローバルリソースの構成, 948
グローバル属性, 169
グローバル属性グループ, 169
グローバル要素, 169
XML スキーマを使用する, 68
ゲージチャートの機能, 344
コーススタックウィンドウ,
XSLT/XQuery デバッガー, 478
コードページ, 1472
コード生成, 1255
コールグラフプロファイリング, 492
コールスタック (XPath/XQuery), 514
コピー,
XML テキストとして, 1150
構造化テキストとしての XML, 1150
コピーコマンド, 1149
コマンド, 1439
キーマップにて表示, 1497
コンテキストメニュー, 1446
ツールバー/メニューへ追加, 1439
メニューから削除, 1446
メニューのリセット, 1446
コマンドライン, 1502
コミュニケーション処理,
SOAP デバッガー, 698
コメント, 165
グリッドビューにて子として追加, 1215
グリッドビューにて挿入, 1200
グリッドビューにて追加, 1208
グリッドビューにて変換, 1221
コメントインとアウト,
テキストビュー内の XML ドキュメント, 266
コンテキストウィンドウ,
XSLT/XQuery デバッガー, 476
コンテキストメニュー,
カスタマイズ, 1451
コマンド, 1446
コンテンツ モデル ビュー,
インターフェイスの詳細, 178
コンポジッターとコンポーネント, 178
一般的な説明, 176
条件型のい割り当て, 189
図に表示されるオブジェクト, 178
編集, 184
コンテンツモデル,
ダイアグラムの保存, 1259
基本的なモデルの作成, 49
切り替え属性, 70

(C) 2015-2021 Altova GmbH

コンテンツモデルビュー, 46
設定, 1264
コンポーネント定義,
再利用する, 61
コンポーネント入力ヘルパー, 209, 215
コンポジッター,
シーケンスコンポジッター, 49
サーチ,
検索を参照, 1162
サーバー上での EDGAR 検証, 1410
サポートオプション, 125
サポートセンター, 1501
シーケンスコンポジッター,
使用する, 49
ショートカット, 259
ツールチップにて表示, 1451
割り当て/削除, 1443
ズーム機能,
スキーマデザインビューにて使用, 1267
スキーマ,
DB XML へ割り当て, 1322
IBM DB2 データベースにおける管理と割り当て, 1318
IBM DB2 にて管理, 1318
UML へ変換, 1250
WSDL スキーマを開く, 690
XML スキーマも参照, 1243
XML スキーマを参照, 46
カタログを介したルックアップ, 411
スキーマをベースに DB を作成, 1344
デザインビュー, 1359
ドキュメンテーション, 75
ドキュメント フォント, 1475
メモリ内, 1255
設定, 1455
スキーマ コンポーネント,
のコンテンツ モデル, 176
スキーマ ビュー,
コンポーネント入力ヘルパー, 209, 215
詳細入力ヘルパー, 213
入力ヘルパー, 209
戻る、進む, 230
スキーマ ビューにおけるアサーション, 195, 199
スキーマ ビューにおけるコメント, 169
スキーマ ビューにおけるコンポジッター, 178
スキーマ ビューにおける属性, 195, 196
スキーマ ビューにおける属性グループ, 196
スキーマ ビューにおける注釈, 169
スキーマ ビューにおける同一性制約, 195, 202
スキーマ ビューにおける入力ヘルパー, 209
スキーマ フォント, 1475
スキーマサブセット, 387, 1274, 1275
スキーマデザインビュー,
ズーム機能, 1267
ダイアグラムの表示, 1268
全てのグローバルを表示, 1268
スキーマデザインメニュー, 1256
スキーマとカタログ, 409
スキーマの修正,
コンテンツ モデル, 176
スキーマの中を検索,
ウィンドウ, 425
スキーマの中を検索, 415
結果, 425
範囲の設定, 422
スキーマの中を検索ウィンドウ, 117
スキーマの中を検索を参照,
スキーマの中を検索, 415
スキーマの内を検索,
検索コマンドの実行, 831
スキーマビュー, 159
ビューの構成, 55
スキーマビュー, 検索,
スキーマの中を検索, 415
スキーマビュー内での命令の処理, 169
スキーマビュー内の注釈, 169
スキーマルール, 391
スキーマへルールセットを追加, 391
定義, 392
スキーマ概要, 46
アイコン, 165
グローバルコメントのライン表示, 165
グローバルなコメント, 165
コンポーネントの並べ替え, 165
とコンテンツ モデル ビュー, 164, 165
編集, 165
スキーマ型の生成, 223
スキーマ内の検索, 415
スキーマの中を検索, 425
プロパティとプロパティの値による検索結果の絞り込み, 419
検索と名前変更, 425
検索ならびに置き換えコマンドの実行, 423
検索をコンポーネントに絞り込む, 418
検索語, 417
置き換え語, 417
スクリプト, 1488

(C) 2015-2021 Altova GmbH

スクリプトエディター,
開始, 1422
スクリプト言語, 1488
スタートグループ,
追加(コンテキストメニュー), 1451
スタイル, 1472
スタイルシート処理命令, 1287, 1288
ステータスバー, 120
ステップアウト,
XSLT デバッガー, 1290
ステップイン,
XSLT デバッガー, 1290
ステップオーバー,
XSLT デバッガー, 1290
スプラッシュ画面, 1466
スペルチェッカー,
カスタム辞書, 1414
スペルチェックのオプション, 1418
スマート制約, 223
セルの最大幅, 1466
ソースコントロール,
チェックアウトを元に戻す, 1177
プロバイダーの変更, 1184
ソース管理, 1489
サポートされているプロバイダー, 1171
ステータスを最新の状態に更新, 1184
ソース管理に追加, 1177
チェックアウト, 1175
ファイルの削除, 1178
フォルダーの取得, 1174
プロジェクトを開く, 1172
プロパティ, 1183
共有, 1178
差分の表示, 1181
最新バージョンの取得, 1174
有効化、無効化, 1173
履歴の表示, 1180
ソース管理マネジャー, 1184
ソフトウェアのライセンスの認証, 1498
ソフトウェア製品ライセンス, 2097
ダイアグラムの表示, 1268
タクソノミ,
インポート, 748, 1396
ターゲット名前空間, 747
ターゲット名前空間におけるタクソノミ, 1395
とリレーションシップ, 756, 757, 760
とリンクベース, 750
とリンクロール, 756
と新規タクソノミウィザード, 741
ファイル内におけるタクソノミ, 750
ファイル内部にあるタクソノミ, 739
作成ステップ, 738
新たに作成, 741
名前空間, 747, 1394
要素の追加, 752
タクソノミのインポート, 1396
タクソノミ内のリレーションシップ, 756, 757, 760
タクソノミ内のリンクベース, 750
タブのサイズ,
および整形, 1363
設定, 1363
タブ文字, 1455
チェック,
スペルチェッカー, 1414
チャート,
3D 設定, 355
XSLT/XQuery プロファイラーにて, 503
X-軸, 350
X-軸選択, 320
Y-軸, 353
Y-軸選択, 325
Z-軸, 354
エクスポート, 357
エリアチャートの機能, 344
オーバーレイ, 331
グリッドライン, 350, 353, 354
ゲージチャートの機能, 344
サイズ, 355
サンプル(ローソク足チャート), 366
サンプル(基本), 358
サンプル(詳細), 360
ソース XPath, 317
タイトル, 342
チャートデータ, 329
チャート設定, 332
フォント, 357
ローソク足チャートの機能, 344
円グラフの機能, 344
外観, 334
概要, 310
系列色, 348
再ロード, 310
色の定義, 348
色の範囲, 348
折れ線グラフの機能, 344
塗りつぶし, 348

(C) 2015-2021 Altova GmbH

チャート,
背景色, 342
複数タブで使用, 310
棒グラフの機能, 344
凡例の削除, 342
凡例の追加, 342
目盛り間隔, 355
余白, 355
チャートウィンドウ, 118
チャートデータテーブル,
作成方法, 313
チャート関数,
サンプル, 2062
チャートデータ構造, 2057
リスト, 2053
チュートリアル,
WSDL, 684
ツール,
外部アプリケーションを参照, 1442
ツールチップ,
ショートカットを表示, 1451
表示, 1451
ツールバー, 120
コマンドを追加, 1439
ツールバーならびにメニューコマンドのリセット, 1440
マクロの追加, 1448
新規作成, 1440
大きなアイコンで表示, 1451
有効化/無効化, 1440
ツールメニュー, 1413
ディレクトリ,
2つのディレクトリを比較, 1427
ディレクトリの比較, 977, 979, 1427
データのインポート/エクスポート,
とグリッドビュー, 151
データベース,
Authentic Viewにて編集, 1295
DB を参照, 565
Oracle XML Db, 1326
XML データをエクスポート, 1350
XMLSpy 内でのサポート, 933
スキーマをベースに DB を作成, 1344
データを XML としてインポート, 1334
とグローバルリソース, 947
レコードの編集, 1316
構造を XML スキーマにインポート, 1339
データベース クエリ,
メッセージペイン, 1316
結果の編集, 1316
データベース/テーブルビュー,
使用方法, 94
データベースクエリ,
DB クエリウィンドウ内のブラウザーペイン, 1310
DB へ接続して問い合わせ, 1307
クエリの作成, 1315
データベースドライバー,
概要, 849
データベースレコードの編集, 1316
データベース接続,
ウィザードの開始, 848
グローバルリソースからの再使用, 876
設定, 847
テーブル,
Authentic View 内, 532
グリッドビューのテーブル, 151
自動的に生成, 1458
動的な(SPS) テーブルの編集, 541
テーブル レイアウトプレビュー, 817
フォーミュラの作成, 818
テーブル 内の Authentic View,
SPS (静的と動的) テーブルの使用, 557
XML テーブルの編集のためのアイコン, 562
テーブルコマンド,
グリッドビュー内, 1222
テーブルパラメーター, 814
テーブルビュー, 1461
グリッドビューに切り替える, 1223
行の挿入, 1223
行の付加, 1223
使用方法, 94
列の並べ替え, 1223, 1224
テーブル内の Authentic View,
XML テーブルの使用, 558
テキスト,
Authentic View 内の書式設定, 553
Authentic View 内の編集, 553
グリッドビューにて子として追加, 1215
グリッドビューにて挿入, 1200
グリッドビューにて追加, 1208
グリッドビューにて変換, 1221
ドキュメント内を検索, 1159
検索と置き換え, 1162
整形, 1157
テキストの範囲,
内の Authentic View, 550
テキストビュー, 130, 1358

テキストビュー, 130, 1358
XML ドキュメントのための特別な編集機能, 266
XML ドキュメント内のコメント, 266
インテリジェントな編集機能, 135
インデント, 131, 132
キーボードショートカット, 148
スキーマ フォント, 1475
ソース折りたたみ, 132
テキストのフォーマット, 131
と XML ドキュメント内の空のライン, 266
フォントプロパティ, 131
ブックマーク, 132
ワードラップ, 131
行番号, 132
自動補完の有効化/無効化, 1461
内の分割機能, 146
入力ヘルパー, 145
編集, 84
テキストビューにおけるソースの折りたたみ, 132
テキストビューにおけるブックマーク, 132
テキストビューにおける行番号, 132
テキストビューにおける入力ヘルパー, 145
テキストビューにおける編集, 135
テキストビューのズーム, 132
テキストビュー設定ダイアログ, 1363
テキストビュー内でのインデント, 131
テキストビュー内でのフォーマット, 131
テキストビュー内でのフォント, 131
テキストビュー内でのワードラップ, 131
テキストビュー内の分割ビュー, 146
テキストファイル,
XML データからエクスポート, 1347
XML としてインポート, 1331
テクニカルサポート, 1501
デバッガー,
SOAP に対してデバッグ, 1387
SOAP に対する設定, 1389
SOAP のための, 696
XSLT デバッガーを開始, 1289
XSLT デバッガーを再起動, 1289
XSLT デバッガーを停止, 1289
ステップアウト, 1290
ステップイン, 1290
ステップオーバー, 1290
セッションを終了, 1290
デバッグウィンドウ, 1293
トレースポイントの挿入/削除, 1291
トレースポイントの有効化/無効化, 1292
ブレークポイント/トレースポイントダイアログボックス, 1292
ブレークポイントの挿入/削除, 1291
ブレークポイントの有効化/無効化, 1291
現在実行中のノードを表示, 1291
設定, 1293
デバッガーウィンドウ,
配置, 481
デフォルト,
エンコーディング, 1460
メニュー, 1446
デフォルト オープン コンテンツ モデル, 193
デフォルトエディター, 1458
デフォルトビュー,
メインウィンドウにおける設定, 1458
テンプレート,
Authentic View における XML ドキュメントのテンプレート, 1294
テンプレート XML ファイル,
Authentic View 内で開く, 529
テンプレートウィンドウ,
XSLT/XQuery デバッガー, 480
テンプレートファイル,
新規ドキュメントに対して, 1128
ドキュメンテーション,
WSDL ファイルのドキュメンテーション, 1375
XBRL タクソノミのドキュメンテーション, 1398
XML スキーマファイルから生成, 1259
スキーマ, 75
ドキュメント,
Oracle XML Db の参照, 1329
スペルチェッカー, 1414
ドキュメントレベル,
XMLSpy の統合のサンプル, 1913
ドッキングウィンドウ, 108
ドック可能なウィンドウ, 1493, 1494
トピック,
目次にて参照, 1496
トレースウィンドウ, 486
XSLT/XQuery デバッガー, 481
トレースポイント,
XSLT/XQuery デバッガー, 486
ダイアログボックス, 1292
トレースポイントの有効化 - XSLT デバッガー, 1292
ナビゲーション,
スキーマデザインでのショートカット, 73
ノード,
現在実行中のノードを表示, 1291
パーサー,

(C) 2015-2021 Altova GmbH

パーサー,
Altova 製品に内蔵の, 2094
バックアップ, 128
パラメーター,
DB クエリ内, 566
インターフェイスを使用してスタイルシートにパスする, 1281
バリデーター,
Altova 製品内で, 2094
ビッグエンディアン, 1460
ビュー,
スキーマデザインビュー, 1359
テキストビュー, 1358
ファイルへ移動, 1363
ブラウザー ビュー, 1360
行/文字へ移動, 1362
最適な幅, 1362
縮退, 1361
縮退ー, 1361
展開, 1361
ビューの変更,
Authentic View, 541
ビューメニュー, 1358
ビルダーモード, 520
ファイル, 1455
2つを比較, 1423
エンコーディング, 1137
ソース管理に追加, 1177
タブ, 1455
デフォルトエンコーディング, 1460
メールで送信, 1143
印刷オプション, 1144
開く, 1132
開くオプション, 1455
最後に使用, 1146
新たに作成, 1128
比較オプション, 1430
閉じる, 1138
保存, 1138
ファイル DSN,
セットアップ, 865
ファイル コマンド内の検索, 1164
ファイル メニュー, 1128
ファイルタイプ, 1458
ファイルの中を検索ウィンドウ, 115
ファイルの比較, 977, 978
オプション, 1430
ファイルの保存,
エンコーディング, 1137
ファイルパス,
XML ドキュメントの挿入, 266
ファイルへ移動, 1363
ファイル拡張子,
XQuery ファイル, 444
カスタマイズ, 1227
ファイル比較の設定, 1430
フィルターとフォーミュラのための JSON メタデータ, 617
フィルターのための XML メタデータファイル, 285
フォーミュラ,
テーブル レイアウトプレビュー内でのフォーミュラの作成, 818
フォーミュラのための XML メタデータファイル, 286
フォント, 1475
スキーマ, 1475
スキーマ ドキュメント, 1475
フォントサイズ,
ブラウザービューにて, 1366
ブックマーク, 1496
ナビゲーション, 1166
挿入と削除, 1166
ブックマークの余白, 1363
ブラウザー, 1466
ビュー, 1360
ブラウザー ビュー, 1365
ブラウザー メニュー, 1365
ブラウザービュー,
フォントサイズ, 1366
ページの読み込みを中止, 1366
最新の状態に更新, 1366
進む, 1365
別のウィンドウ, 1366
戻る, 1365
ブラウザーペイン,
データベースクエリウィンドウにて, 1310
プラットフォーム,
Altova 製品のための, 2094
ブレークポイント,
SOAP デバッガーを使用する, 707
XSLT/XQuery デバッガーで使用する, 483
ダイアログボックス, 1292
ブレークポイントダイアログ, 469
ブレークポイントの有効化 - XSLT デバッガー, 1291
プログラマーのレファレンス, 1503
プログラム設定, 1454
プロジェクト, 1186
URL の追加, 1185
アクティブなファイルを追加, 1185, 1186

プロジェクト, 1186
グローバルリソースの追加, 1185
ソース管理に追加, 1177
ファイルの追加, 1185
フォルダーの追加, 1186
プロパティ, 951, 1194
を使ったバッチ処理, 955
開く, 1170
外部ウェブフォルダーの追加, 1189
外部フォルダーの追加, 1186
概要, 950, 1168
関連するファイルを追加, 1186
再ロード, 1171
最近使用されたプロジェクト, 1197
作成と編集の方法, 951
使用する, 955
使用する利点, 955
新規作成, 1170
閉じる, 1171
保存, 951, 1171
名前をつける, 951
プロジェクト ウィンドウ, 110
プロジェクトウィンドウ, 1493, 1494
表示/非表示の切り替え, 1494
プロジェクトメニュー, 1168
プロファイラー, 492, 1287
プロファイリング, 492
コールグラフ, 492
呼び出し回数, 492
ヘルプ,
キーマップ, 1497
目次, 1496
ヘルプ メニュー, 1496
ヘルプシステム, 1496
ホットキー, 1443
マークアップ,
内の Authentic View, 541, 544
マークアップ(Authentic View にて),
隠す, 1301
小さな/大きな/混合マークアップを表示, 1301
マークアップの非表示, 541
マークアップを隠す, 544
マークアップを隠す(Authentic View にて), 1301
マクロ,
アプリケーションマクロの起動, 1423
メニュー/ツールバーへ追加, 1448
編集ボタン, 1451
メインウィンドウ, 109
メインウィンドウにおけるドキュメント, 109
メインカタログ, 1227
メタデータファイル, 285, 286, 617
メッセージウィンドウ,
XSLT/XQuery デバッガー, 479
メニュー, 1439
Authentic, 1294
DTD/スキーマ, 1243
JSON, 1239
SOAP, 1380
WSDL, 1367
XML, 1198
XSL/XQuery, 1277
ウィンドウ, 1493
カスタマイズ, 1446
コマンドの追加/削除, 1439
コマンドを削除, 1446
スキーマデザイン, 1256
デフォルト/XMLSpy, 1446
ビュー, 1358
プロジェクト, 1168
ヘルプ, 1496
マクロの追加, 1448
ルーツ, 1413
変換, 1331
編集, 1148
メニュー ブラウザー, 1365
メニューバー, 120
メモリ,
スキーマ情報の保管, 1255
メモリ要件, 2094
モデルグループ, 169
やり直しコマンド, 1149
ユーザー DSN,
セットアップ, 865
ユーザーインターフェイス詳細, 108
ユーザーマニュアル, 28
ユーザーリファレンス, 1127
ライセンス, 2097
Altova ソフトウェアのための, 1498
情報, 2096
ライセンス計測,
Altova 製品にて, 2096
リセット,
ショートカット, 1443
ツールバーならびにメニューコマンド, 1440
メニューコマンド, 1446
リッチエディット, 1301

(C) 2015-2021 Altova GmbH

リトルエンディアン, 1460
リンク,
Authentic View 内の, 553
ルール,
スキーマ検証(スキーマルールを参照), 391
レジストリ,
設定, 1454
ローソク足チャートの機能, 344
ワードラップ,
有効化/無効化, 1362
位置,
行, 1362
文字, 1362
移動ウィンドウ, 108
印刷,
Authentic View から, 538
印刷オプション, 1144
印刷設定, 1146
隠す, 1493, 1494
右へ移動コマンド, 1224
円グラフ, 325
円グラフの機能, 344
解析,
XSLT, 1479
改行, 1455
改行マーカー, 1363
開く,
ファイル, 1132
開くオプション,
ファイル, 1455
開始,
XSLT デバッガーを開始, 1289
外部 JSON コンテンツ,
JSON グリッドビューにコピーする, 610
JSON グリッドビューにドラッグアンドドロップする, 610
外部 XML コンテンツ,
XML グリッドビューにコピーする, 278
XML グリッドビューにドラッグアンドドロップする, 278
外部 XSL プロセッサ, 1479
外部アプリケーション,
ファイルから開く, 1442
外部キー,
SQLite 内での無効化, 875
外部ソースからの JSON テキスト,
ドキュメントに素早く追加, 664
外部ソースからの XML テキスト,
ドキュメントに素早く追加, 304
外部で解析されるエンティティ, 1458
拡張グリッドビュー, 1359
グリッドビューを参照, 83
拡張スキーマ検証, 391
拡張検証, 406
SchemaAgent にて実行, 1273
割り当て,
ショートカットをコマンドへ割り当て, 1443
環境変数, 413
基底型,
修正, 222
技術データ, 2094
空のライン,
テキストビュー内の XML ドキュメント, 266
空白,
削除, 1158
空白の削除, 1158
空白マーカー, 1363
空要素, 1458
繰り返される要素, 1461
型,
XML ドキュメントでの拡張, 81
検索,
ドキュメント内のテキスト, 1159
と置き換え(ドキュメント内のテキストにて), 1162
正規表現の使用, 140
検証, 122, 1227
DTD を XML ドキュメントへ割り当て, 1243, 1244
SchemaAgent により関連するスキーマの検証, 406
WSDL ファイル, 689, 1232
XML ドキュメント, 88
スキーマルールを使った拡張, 391
検証の設定, 1455
元に戻すコマンド, 1149
現在実行中のノードを表示,
XSLT デバッガー, 1291
呼び出し回数プロファイリング, 492
構造化されたテキスト, 1461
構文による色分け, 446, 1458, 1466
XQuery, 446
行,
移動, 1362
削除(Authentic View にて), 1302
上/下へ移動, 1303
挿入(Authentic View にて), 1302
追加(Authentic View にて), 1302
複製(Authentic View にて), 1302
行/文字へ移動, 1362
行番号の余白, 1363

混合マークアップ(Authentic View にて), 1301
左へ移動コマンド, 1224
再ロード, 1455
  ファイルの変更, 1137
再起動,
  XSLT デバッガーを再起動, 1289
再定義, 169
最後に使用されたファイル,
  リスト, 1146
最適な幅, 1362, 1466
作成,
  スキーマをベースにした DB, 1344
削除,
  コンテキストメニューのコマンドを削除, 1446
  ショートカット, 1443
  ツールバー, 1440
  ツールバーからアイコンを削除, 1439
  ツールバーからコマンドを削除, 1439
  トレースポイント - XSLT デバッガー, 1291
  ブレークポイント - XSLT デバッガー, 1291
  行(Authentic View にて), 1302
削除コマンド, 1149
参照,
  Oracle XML Db, 1329
子の追加コマンド,
  グリッドビューにて, 1213
視覚的な書式設定,
  内の Authentic View, 575
試用期間,
  Altova ソフトウェア製品の試用, 2096
自動バックアップの設定, 1455
自動検証, 1458
自動検証の無効化, 1458
自動的なバックアップ, 128
自動非表示ウィンドウ, 108
自動補完,
  テキストビューにて有効化/無効化, 1461
辞書,
  スペルチェッカー, 1414
  既存のものを修正, 1414
  新たに追加, 1414
式ビルダー, 520
終了,
  デバッガーセッションを終了, 1290
重ねて表示,
  ウィンドウ, 1493
縮退,
  未選択を縮退, 1361
出現,
  回数, 49
出力ウィンドウ,
  表示/非表示の切り替え, 1494
出力フォーマット, 1455
処理命令,
  グリッドビューにて子として追加, 1216
  グリッドビューにて挿入, 1201
  グリッドビューにて追加, 1209
  グリッドビューにて変換, 1221
署名,
  XML 署名を参照, 370, 1234
小さなマークアップ, 544
小さなマークアップ (Authentic View), 1301
小さなマークアップの表示, 541
章, 1496
証明書, 377
証明書ストア, 377
証明書の公開鍵, 377
証明書の秘密鍵, 377
詳細入力ヘルパー, 49, 213
上/下へ移動,
  行(Authentic View にて), 1303
情報 ウィンドウ, 1494
情報ウィンドウ, 112
  CSS ドキュメントに対して, 582
  HTML ドキュメントに対して, 580
  XSLT/XQuery デバッガー, 480
  配置, 481
情報ウィンドウ XSLT タブ,
  XSL アウトラインも参照, 437
  と XSLT ドキュメント, 437
  と ZIP フォルダーの作成, 437
  とプロジェクト, 437
  の記述, 437
条件型の割り当て, 189
条件型の代替, 189
色, 1472
新規タクソノミの作成ウィザード, 741
新規の XML ドキュメントのため,
  作成, 79
新規ファイル,
  作成, 1128
新規機能, 29, 30
進む,
  スキーマ ビューにて, 230
整形,
  テキストビューにて, 1157

整形式チェック,
XML ドキュメントのため, 88
整形式のチェック, 1225
正規表現, 1164
検索と置換の条件式, 140
検索文字列にて, 1159
生成,
スキーマをベースに DB 構造を生成, 1344
静的な (SPS) テーブル 内の Authentic View,
使用方法, 557
切り取りコマンド, 1149
切り替え, 1493, 1494
折りたたみ余白, 1363
折り返し,
ワードラップの有効化/無効化, 1362
折れ線グラフの機能, 344
設定, 122, 1454
3D 設定, 355
XSLT デバッガー, 1293
エリアチャートの機能, 344
ゲージチャートの機能, 344
スクリプト, 1488
チャート X-軸, 350
チャート Y-軸, 353
チャート Z-軸, 354
チャートグリッド, 350, 353, 354
チャートサイズ, 355
チャートのタイトル, 342
チャートの色, 348
チャートの背景, 342
チャートの凡例, 342
チャートフォント, 357
の概要, 44
ローソク足チャートの機能, 344
円グラフの機能, 344
折れ線グラフの機能, 344
棒グラフの機能, 344
設定ビュー,
コンテンツモデルビューのダイアログ, 1264
選択されたノードに対しての XPath, 540
全てのグローバルを表示, 1268
全て選択コマンド, 1158
挿入, 1302
トレースポイント - XSLT デバッガー, 1291
ブレークポイント - XSLT デバッガー, 1291
属性, 70, 169
グリッドビューにて子として追加, 1214
グリッドビューにて挿入, 1199
グリッドビューにて追加, 1207
グリッドビューにて変換, 1221
コンテンツモデルビューに切り替え, 70
スキーマ定義にて, 70
属性のプレビュー, 1466
属性値,
Authentic View 内に挿入, 536
属性入力ヘルパー,
Authentic View, 546
大きなマークアップ, 544
大きなマークアップ(Authentic View にて), 1301
大きなマークアップの表示, 541
単純型,
アサーション, 215
グローバル, 169
サンプル値, 215
スキーマ定義にて, 61
パターン, 215
ファセット, 215
匿名, 169
名前を持つ, 169
列挙, 215
単純型のアサーション, 215
単純型のサンプル値, 215
単純型のパターン, 215
単純型のファセット, 215
単純型の派生, 213
単純型の列挙, 215
置き換え, 115
置換,
正規表現の使用, 140
置換え,
テキスト, 1159
ドキュメント内のテキスト, 1162
複数のファイル内のテキスト, 1164
著作権に関する情報, 2096
追加コマンド,
グリッドビューにて, 1206
停止,
XSLT デバッガーを停止, 1289
定義,
3D 設定, 355
X-軸設定, 350
Y-軸設定, 353
Z-軸設定, 354
エリアチャートの機能, 344
グリッドライン, 350, 353, 354
ゲージチャートの機能, 344

定義,
チャートサイズ, 355
チャートのタイトル, 342
チャートの色, 348
チャートフォント, 357
チャート内のフォント, 357
ローソク足チャートの機能, 344
円グラフの機能, 344
折れ線グラフの機能, 344
棒グラフの機能, 344
定義概要グリッド, 630
適用, 1454
展開,
全て展開, 1361
貼り付け,
XML, 550
XML として, 553
テキスト, 550
テキストとして, 553
貼付けコマンド, 1149
電子メール,
ファイルを送信, 1143
統合,
アプリケーション内の XMLSpy, 1905
動的な (SPS) テーブル 内の Authentic View,
使用方法, 557
動的なテーブル,
編集, 541
同一性制約,
コンテンツモデルビューに切り替え, 70
内の Authentic View テーブル,
使用方法, 557
日付,
手動での変更, 572
日付の選択,
内の Authentic View の使用, 571
入力ヘルパー, 113, 1494
XQuery, 445
グリッドビュー内, 92
更新, 1233
詳細, 49
表示/非表示の切り替え, 1494
入力ヘルパー (テキストビュー、Authentic ビュー), 301
入力ヘルパーの更新コマンド, 1233
派生型,
基底型の切り替え, 222
背景情報, 2094
配置,
左右に並べて, 1493
上下に並べて, 1493
配布,
Altova ソフトウェア製品, 2096
Altova ソフトウェア製品の配布, 2096
比較,
ディレクトリの比較, 979
ファイルとディレクトリの比較, 977
ファイルの比較, 978
非 XML ファイル, 1458
表示, 1493, 1494
拡張グリッドビュー, 1359
表示方法, 1466
評価キー,
Altova ソフトウェアのための, 1498
評価機関,
Altova ソフトウェア製品, 2096
複合型, 61
グローバル, 169
スキーマ定義にて, 61
定義を拡張する, 61
匿名, 169
名前を持つ, 169
複合型マークアップ, 544
複合型マークアップの表示, 541
複数ユーザー, 1455
文字,
位置, 1362
文字セット,
エンコーディング, 1460
文法, 1458
変換,
MS Word データから XML へ, 1339
Oracle XML Db, 1326
XSLT 変換を参照, 1280
スキーマから DB 構造へ, 1344
データベーススキーマを XML スキーマへ, 1339
データベースデータを XML として, 1334
テキストファイルから XML へ, 1331
変換 メニュー, 1331
変換コマンド,
グリッドビューにて, 1220
変更コマンドでの検証, 1233
変更を検知, 1455
変数ウィンドウ,
XSLT/XQuery デバッガー, 477
編集,
マクロボタン, 1451

編集ビュー, 126
編集メニュー, 1148
法的な情報, 2096
棒グラフの機能, 344
未選択, 1361
無効化,
トレースポイント - XSLT デバッガー, 1292
ブレークポイント - XSLT デバッガー, 1291
名前なし要素リレーションシップ,
MS SQL Server スキーマ設定, 1271
名前空間,
WSDL ドキュメントにて, 684
XBRL におけるタクソノミ, 747
スキーマデザインビューにお, 1256
スキーマ内の, 48
名前空間プレフィックス,
グリッドビューにて追加, 1233
名前付きスキーマのリレーションシップ,
MS SQL Server スキーマ設定, 1271
名前付きテンプレート, 434
名前付きテンプレートの呼び出し, 434
目次, 1496
戻る,
スキーマ ビューにて, 230
要素, 57
オプションにする, 57
グリッドビューにて子として追加, 1214
グリッドビューにて挿入, 1199
グリッドビューにて追加, 1207
グリッドビューにて変換, 1221
コンテンツの制限, 57
要素の型,
XML ドキュメントで指定する, 81
要素の中へコマンド, 1224
要素入力ヘルパー,
Authentic View, 546
履歴, 230
隣り合わせ, 1466
列挙,
XMLSpyControl, 1971
属性を定義する, 70